UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

PLAID INC.
Petitioner

v.

SIX TREES CAPITAL, LLC
Patent Owner According to PTO Assignment Records

and

SECURE AUTHENTICATION TECHNOLOGIES LLC
Patent Owner Identified in Patent Complaint

Case No. IPR2026-00157

DECLARATION OF ALEX C. SNOEREN, PH.D.
REGARDING U.S. PATENT NO. 11,315,090
CLAIMS 1-10

TABLE OF CONTENTS

Page

I. BACKGROUND 1
II. LEGAL PRINCIPLES 7
III. PERSON OF ORDINARY SKILL IN THE ART 12
IV. OVERVIEW OF THE '090 PATENT 13
A. The '090 Specification 13
B. Summary of the '090 Prosecution History 16
V. CLAIM CONSTRUCTION 21
VI. THE CHALLENGED CLAIMS ARE UNPATENTABLE 22
A. Overview of Prior Art 22
B. Ground 1: Claims 1, 4-6, and 9-10 are anticipated by Hsu 29
1. Overview of Hsu (Ex-1004) 29
2. Claim 1 32
3. Claim 4 45
4. Claim 5 46
5. Claim 6 47
6. Claim 9 49
7. Claim 10 49
C. Ground 2: Claims 1, 4-6, and 9-10 are anticipated by Hazlehurst 49
1. Overview of Hazlehurst (Ex-1005) 49
2. Claim 1 53
3. Claim 4 66
4. Claim 5 68
5. Claim 6 69
6. Claim 9 71
7. Claim 10 71

D. Ground 3: Claims 1, 4-6, and 9-10 are obvious over Emerson in view of Hsu 71
1. Overview of Emerson (Ex-1006) 71
2. Claim 1 75
3. Claim 4 91
4. Claim 5 92
5. Claim 6 93
6. Claim 9 98
7. Claim 10 99
E. Grounds 4, 5, and 6: Claims 2, 3, 7, and 8 are obvious over Hsu, Hazlehurst, or Emerson/Hsu in view of Peotta 99
1. Overview of Peotta (Ex-1007) 99
2. Claim 2 101
3. Claim 3 107
4. Claim 7 108
5. Claim 8 108
F. Grounds 7, 8, and 9: Claims 2, 3, 7, and 8 are obvious over Hsu, Hazlehurst, or Emerson/Hsu in view of 2FA Q&A 109
1. Overview of 2FA Q&A (Ex-1009) 109
2. Claim 2 111
3. Claim 3 116
4. Claim 7 117
5. Claim 8 117
VII. ENABLEMENT AND SECONDARY CONSIDERATIONS OF NON-OBVIOUSNESS 117
VIII. AVAILABILITY FOR CROSS-EXAMINATION 118
IX. RIGHT TO SUPPLEMENT 118
X. JURAT 118

I declare as follows:

1. My name is Alex C. Snoeren.

2. I have been retained by Plaid Inc. and Gibson, Dunn & Crutcher LLP in this case as an expert in the relevant art.

3. I have been asked to provide my opinions and views on the materials I have reviewed in this case related to U.S. Patent No. 11,315,090 (the "'090 patent", Ex-1001), and the scientific and technical knowledge regarding the same subject matter as the '090 patent around the priority date of the '090 patent, January 28, 2014.

4. My opinions and underlying reasoning for the opinions are set forth below.

I. BACKGROUND

5. My qualifications for forming the opinion set forth in this Declaration are summarized here and include my educational background, career history, publications, and other relevant qualifications. My full curriculum vitae is attached as Exhibit A to this declaration and includes my detailed employment background, professional experience, and list of publications.

6. Overall, I have over twenty-nine (29) years of experience in the field of computer science and, specifically, networking and network communications, operating systems, mobile computing, and security.

7. I am currently the Peter Banks Endowed Chair in the Jacobs School of Engineering and Professor in the Department of Computer Science and Engineering at the University of California, San Diego (UC San Diego), where I am a member of the Systems and Networking Research Group, the Security and Cryptography Research Group, and the Center for Networked Systems. I have served on the faculty of UC San Diego for more than twenty years, starting in 2002 and continuing to the present day. My research spans the areas of computer networking, distributed systems, operating systems, security, and mobile and wireless networking. While at UC San Diego, I have published in and made contributions to the areas of datacenter networking, network security, mobile and wireless networking, and fault localization, among others. I have been a Visiting Faculty Researcher at Google, on a full-time basis during the 2022-23 academic year and in a one-day-a-week capacity during the 2023-24 and 2024-25 academic years. During the 2011-12 academic year, I held a visiting appointment as an Associate Professor in the College of Computing at the Georgia Institute of Technology, where I conducted research focused on security, privacy, and censorship on the Internet.

8. I received a Ph.D. in Electrical Engineering and Computer Science from the Massachusetts Institute of Technology (MIT) in 2003. My dissertation, awarded George M. Sprowls Honorable Mention from the MIT EECS department,

devised a new architecture called *Migrate* to allow mobile devices such as laptop computers and personal digital assistants ("PDAs") to maintain connectivity to the Internet while changing physical location and/or form of network connectivity (e.g., switching between wired Ethernet and cellular data connections). The Migrate technology developed in my dissertation played a key role in fulfilling the vision of MIT's Project Oxygen, which included streaming live audio and video over hand-held devices despite rapidly changing network conditions. In the late 1990s, also at MIT, I devised a scheme that allowed mobile devices to leverage multiple cellular connections simultaneously to obtain sufficient bandwidth to support live video calls as part of the DARPA Communicator project, which sought to outfit the warfighter with wireless multimedia-capable devices on the battlefield.

9. From 1999 to 2002, I worked at Bolt, Beranek, and Newman (BBN) Technologies—the company that developed the first Internet routers—as a research scientist, where I helped implement a protocol to dramatically increase the flexibility of Internet routing to support improved performance for the rapidly growing set of applications on the Internet, including streaming multimedia. While at BBN, I also led the development of a system capable of tracing a single Internet packet back to its origin.

10. Prior to MIT, I attended the Georgia Institute of Technology, where I received B.S. degrees in Computer Science (Highest Honors, 1996) and Applied Mathematics (Highest Honors, 1997) and an M.S. degree in Computer Science (1997). I was honored as the 1996 Outstanding Undergraduate from the College of Computing, graduating with a 4.0 grade point average. That same year, I was named a United States finalist in the Rhodes Scholar competition. My Ph.D. studies were supported by a National Defense Science and Engineering Graduate (NDSEG) Fellowship. I was also awarded—but declined—a National Science Foundation Graduate Fellowship. During my undergraduate years at Georgia Tech, I was the recipient of a Georgia Tech Presidential Scholarship as well as National Academy of Space, Science & Technology, National Merit, and National Science Scholarships.

11. I have authored twenty-five (25) journal or magazine articles and one hundred and twenty-five (125) conference and workshop publications. I am also an inventor on six (6) awarded patents relating to computer networking and security. I have served as the dissertation advisor for twenty (20) Ph.D. students, five (5) of which have gone on to tenure-track faculty positions in top US Computer Science departments. I am the recipient of several professional honors and awards. These include the 2009 Alfred P. Sloan Fellowship, 2004 National Science Foundation (NSF) CAREER Award, and a 2017 Facebook Faculty Research Award. I have

authored several award-winning papers, including the Jay Lepreau Best Paper at USENIX OSDI 2008 and best papers at ACM SIGCOMM in 2001, 2007, 2018, and 2023. I served on the Defense Advanced Research Projects Agency (DARPA) Defense Science Study Group (DSSG) from 2012-2013 and have been a member of the JASON defense science advisory group since 2019 and the DARPA Information Science and Technology (ISAT) Study Group since 2025.

12. I am a Fellow of the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE). I have participated in the Internet Engineering Task Force (IETF) standardization process on multiple occasions, presenting at meetings of the IETF including the End-to-End Research group, Birds-of-a-Feather (BoF) sessions, and co-authoring multiple Internet Drafts.

13. I have reviewed the specification, claims, and file history of the ’090 patent (Ex-1001, Ex-1002). I have also reviewed the following references, all of which I understand to be prior art to the ’090 patent:

- U.S. Patent Appl. No. 2008/0115198 to Hsu et al. (“Hsu”, Ex-1004);
- U.S. Patent No. US 8,261,334 to Hazlehurst et al. (“Hazlehurst”, Ex-1005);
- U.S. Patent Appl. No. 2009/0198615 to Emerson et al. (“Emerson”, Ex-1006);

- Laerte Peotta et al., "A Formal Classification of Internet Banking Attacks and Vulnerabilities", *International Journal of Computer Science & Information Technology* (IJCSIT), Vol. 3, No. 1, Feb. 2011 ("Peotta", Ex-1007);
- "Could My Bank's Two-Factor Authentication be Hacked?", *IT Security Stack Exchange*, May 29, 2013 ("2FA Q&A", Ex-1009);
- Irv Englander, "The Architecture of Computer Hardware, System Software, and Network", 4th Ed., 2009 ("Englander", Ex-1010);
- Declaration of Mary Bolin, Ph.D. (Ex-1011);
- Declaration of Mina Ching (Ex-1012);
- Ross Anderson, "Security Engineering", 1st Ed., 2001, Chapter 19 ("Anderson I", Exhibit B);
- Ross Anderson, "Security Engineering", 2nd Ed., 2008, Chapters 3 and 10 ("Anderson II", Exhibit C);
- Robert Morris & Ken Thompson, "Password Security: A Case History", *Communications of the ACM*, Vol. 22, No. 11, Nov. 1979 ("Morris & Thompson", Exhibit D);
- Bruce Schneier, "Applied Cryptography", 2nd Ed., 1996, Chapter 12 ("Schneier", Exhibit E);

- U.S. Patent Appl. No. 2013/0276080 to Nakonetzki et al. ("Nakonetzki", Exhibit F).

14. I have also reviewed the exhibits and references cited in this Declaration, including:

- Google Scholar search results, "Internet banking authentication authorization", custom range until 2013 (Exhibit G);
- Pascal-Emmanuel Gobry, "Google Is Indexing Stack Overflow At 10 Times Per Second", *Business Insider*, Mar. 30, 2011 (Exhibit H);
- Google search results, "multi-factor authentication phone number internet banking", custom range until Dec. 31, 2013 (Exhibit I);
- Adrianne Jeffries, "Forget Quora, New York's Stack Overflow Is Killing It", *BetaBeat*, Jan. 25, 2011 (Exhibit J);
- Jon Swartz, "Q&A websites like Quora and Stack Exchange take off", USATODAY, Jan. 24, 2011 (Exhibit K).

II. LEGAL PRINCIPLES

15. I am not an attorney and offer no legal opinions, but in the course of my work I have had experience studying and analyzing patents and patent claims from the perspective of a person skilled in the art.

16. For the purposes of my analysis and conclusions expressed in this Declaration, I have applied the legal principles described in the following paragraphs.

17. Petitioner's counsel has informed me that for the prior art to inherently disclose a claimed limitation, the prior art need not expressly disclose the limitation, so long as the claimed limitation necessarily flows from a disclosure in the prior art. I also understand that it is acceptable to examine evidence outside the prior art reference (extrinsic evidence) in determining whether a feature, while not expressly discussed in the reference, is necessarily present in that reference.

18. Petitioner's counsel has informed me that a patent claim can be considered to have been obvious to a person having ordinary skill in the art ("POSITA") at the time the application was filed. I am informed that this means that, even if all of the requirements of a claim are not found in a single prior art reference, the claim is not patentable if the differences between the subject matter in the prior art and the subject matter in the claim would have been obvious to a POSITA at the time of the invention.

19. I have been informed by Petitioner's counsel that a determination of whether a claim would have been obvious should be based upon several factors, including, among others:

- The level of ordinary skill in the art at the time the application was filed;

- The scope and content of the prior art; and
- What differences, if any, existed between the claimed invention and the prior art.

20. I have been informed by Petitioner's counsel that a single reference can render a patent claim obvious if any differences between that reference and the claims would have been obvious to a POSITA. Alternatively, I understand that the teachings of two or more references may be combined in the same way as disclosed in the claims, if such a combination would have been obvious to one having ordinary skill in the art. In determining whether a combination based on either a single reference or multiple references would have been obvious, it is appropriate to consider, among other factors:

- Whether the teachings of the prior art references disclose known concepts combined in familiar ways, and when combined, would yield predictable results;
- Whether a person having ordinary skill in the art could implement a predictable variation, and would see the benefit of doing so;
- Whether the claimed elements represent one of a limited number of known design choices, and would have a reasonable expectation of success by those skilled in the art;

- Whether a person having ordinary skill would have recognized a reason to combine known elements in the manner described in the claim;
- Whether there is some teaching or suggestion in the prior art to make the modification or combination of elements claimed in the patent; and
- Whether the innovation applies a known technique that had been used to improve a similar device or method in a similar way.

21. I have been informed by Petitioner's counsel that one of ordinary skill in the art has ordinary creativity and is not an automaton. Petitioners' counsel has also informed me that in considering obviousness, obviousness may not be determined using the benefit of hindsight, including hindsight derived from the patent being considered.

22. I further understand that a hypothetical POSITA is assumed to know and to have all relevant prior art in the field of endeavor covered by the patent at issue. I have been informed that there are at least two criteria for determining whether prior art is analogous and thus can be considered prior art: (1) whether the art is from the same field of endeavor, regardless of the problem addressed; and (2) if the reference is not within the field of the patentee's endeavor, whether the reference still is reasonably pertinent to the particular problem with which the patentee is involved. I have also been informed that the field of endeavor of a patent is not limited to the specific point of novelty, the narrowest possible

conception of the field, or the particular focus within a given field. I have also been informed that a reference is reasonably pertinent if, even though it may be in a different field from that of the patentee's endeavor, it is one which, because of the matter with which it deals, logically would have commended itself to a patentee's attention in considering their problem.

23. I further understand that, in making a determination as to whether the claimed invention would have been obvious to a POSITA, the U.S. Patent Trial and Appeal Board may consider certain objective factors if they are present, such as: commercial success of products practicing the claimed invention; long-felt but unsolved need; teaching away; unexpected results; copying; and praise by others in the field. These factors are generally referred to as "secondary considerations" or "objective indicia" of nonobviousness. I understand, however, that for such objective evidence to be relevant to the obviousness of a claim, there must be a causal relationship (called a "nexus") between the claim and the evidence, and that this nexus must be based on a novel element of the claim rather than something in the prior art. I also understand that even when they are present, secondary considerations may be unable to overcome primary evidence of obviousness (such as motivation to combine with predictable results) that is sufficiently strong.

III. PERSON OF ORDINARY SKILL IN THE ART

24. I have been informed and understand that, in the context of an invalidity analysis, a POSITA is not a specific real individual, but a hypothetical person who looks to prior art at the time of the invention. I further understand that the factors that may be considered in determining the level of ordinary skill include: (1) the level of education and experience of persons working in the field; (2) the types of problems encountered in the field; (3) the teachings of the prior art regarding solutions to such problems; (4) the sophistication of the technology at the time of the alleged invention; and (5) the rapidity of innovation. I understand that these factors need not all be considered for the analysis, and that one or more of these factors may control.

25. I was asked to provide my opinion on the level of one of ordinary skill in the art with respect to the alleged invention of the ’090 patent no earlier than January 28, 2014. In my opinion, a POSITA would have had a bachelor’s degree in computer science, electrical engineering, computer engineering, or similar technical field. A POSITA would also have had approximately one to two years of experience working in the analysis, design, development, and/or implementation of authentication protocols for Web applications. Additional experience might substitute for less education and *vice versa*.

26. As of January 28, 2014, I met, and in fact exceeded, the qualifications of a person of ordinary skill in the art. To be clear, all of my opinions in this declaration are from the perspective of one of ordinary skill in the art as I have defined it here during the relevant timeframe.

IV. OVERVIEW OF THE '090 PATENT

A. The '090 Specification

27. The '090 patent relates to "systems, product programs and methods of using the same for optimizing the distribution of financial assets among financial accounts via a fully-automated approach that does not require user intervention on an ongoing basis." Ex-1001, 1:18-22. According to the patent, prior art such as automated cash sweep systems have sought to allocate cash across multiple accounts to provide increased FDIC insurance protection, but not necessarily to optimize interest yield. *Id*., 2:21-25. Furthermore, in prior art cash sweep systems, the accounts are not directly held by the user and therefore cannot be directly accessed by the user without coordination with the cash sweep administrator. *Id*., 2:60-63.

28. To address these purported shortcomings, the patent describes a "method for permitting a **third-party entity to gain access, in a secure manner, to an online account held by an account holder** that is **secured by a multi-**

factor authentication protocol". *Id.*, claim 1 (emphasis added[1]). The financial management system detailed in the '090 patent can "act as proxy for the user and respond to any multi-factor authentication that may be required for account access in a way that does not require user intervention". *Id.*, 18:54-58.

29. FIG. 1 shows a logical configuration during operation:

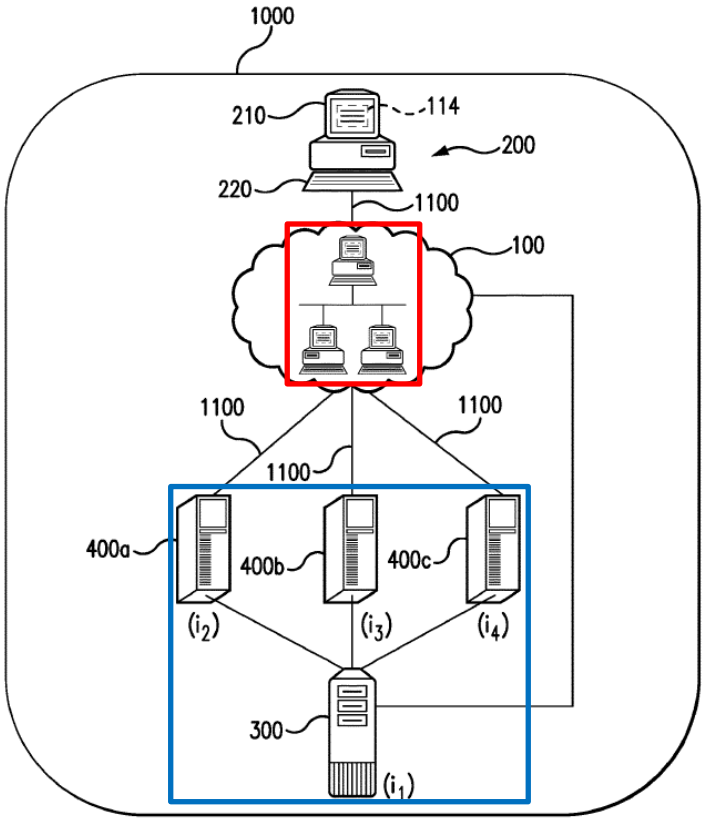


FIG. 1

'090 Patent (Ex-1001), FIG. 1 (Annotated)

[1] These four elements are color-coded throughout the Declaration for the ease of visual comparison with prior art.

30. In short, a financial institution that manages one or more of **accounts 300, 400a, 400b, 400c** may require a response from user to be input into a **multi-factor authentication request**. *Id.*, 18:24-30 (emphasis added). **Financial management network 100** is "configured to **communicate directly**", via e-mail or SMS, "**with one or more of [the] accounts … as a proxy for the user … to receive … unique authentication codes**", without intervention by the user on client device **200**. *Id.*, 18:34-52 (emphasis added). The "**financial management network** may comprise … a credential data store **120**" (see FIG. 3 below) where it stores credentials "capable of assessing any of [the] accounts" that it has collected. *Id.*, 12:19-21, 13:9-18 (emphasis added). "To enhance security, the **credential data store 120 is formed as an encrypted** key/value database to protect the users' [credentials]". *Id.*, 20:15-19 (emphasis added).

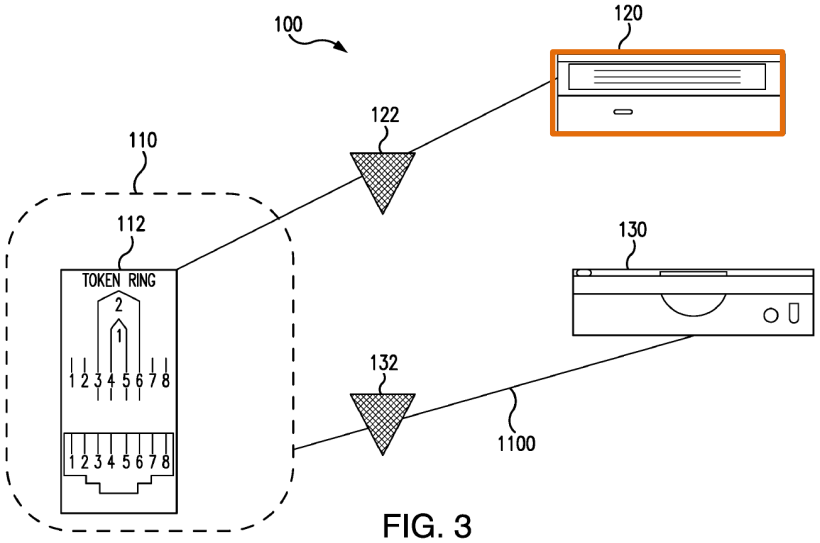


FIG. 3

'090 Patent (Ex-1001), FIG. 3 (Annotated)

B. Summary of the ’090 Prosecution History

31. The ’090 patent (Ex-1001) issued from U.S. Patent Appl. No. 16/850,150, filed on April 16, 2020. Ex-1001, 1. It purports to claim priority to Provisional Appl. No. 61/932,495, filed on January 28, 2014. *Id*., 1; Ex-1002, 96. Prosecution involved two rejections before the ’090 patent issued.

32. The patent applicants initially sought claims directed to a method of “accessing … the online account using account holder’s credentials; receiving a multi-factor authentication request … transmitting the request to the account holder and receiving the response thereto from the account holder [and] transmitting the account holder’s response to the online account to satisfy the multi-factor authentication request.” *Id*., 70 (Claims dated 4/16/2020, claim 1). The patent applicants claim that the method further includes “storing … the account holder’s endpoints for the online account”, such as an email address or a phone number, “in a database” and “utilizing said … endpoints to … gain access to the online account … without requiring a response by the account holder to any subsequent multi-factor authentication requests.” *Id*. (Claims dated 4/16/2020, claims 2 and 3).

33. These claims were rejected as directed to abstract ideas and as anticipated by US 8,495,720 to Counterman. *See* Ex-1002, 318-30 (Non-Final Rejection dated 6/11/2021). To distinguish Counterman, the applicants amended

the claims to further require the third-party entity to collect the account holder's credentials and to store the credentials in a data store. *Id*., 370. (Amendments to the Claims dated 9/10/2021):

> A method for permitting a third-party entity to gain access to an online account held by an account holder that is secured by a multi-factor authentication protocol, comprising the following steps performed by the third-party entity:
>
> collecting the account holder's credentials;
>
> storing the account holder's credentials using one or more processors, wherein the credentials are stored in a data store;
>
> verifying the account holder's credentials by [[(1)]] accessing, via one or more processors, the online account using the account holder's credentials;
>
> [[(2)]] receiving a multi-factor authentication request from the online account, the multi-factor authentication request indicating that a response is required from the account holder before further access to the online account is required;
>
> [[(3)]] ~~transmitting~~ prompting the ~~request to the~~ account holder for a response and receiving the response thereto from the account holder; and

[[(4)]] transmitting the account holder's response to the online account to satisfy the multi-factor authentication request.[[;]]

34. The Examiner rejected all the amended claims again as patent ineligible, because the claims "simply recite accessing an account, receiving and sending data as performed by generic hardware." *Id*., 415 (Final Rejection dated 9/29/2021). In response, the applicants amended independent claims 1 and 6 to require an additional step, arguing that encryption cannot be achieved by a human. *Id*., 457-59. (Amendments to the Claims 11/29/2021):

A method for permitting a third-party entity to gain access, in a secure manner, to an online account held by an account holder that is secured by a multi-factor authentication protocol, comprising the following steps performed by the third-party entity:

…

encrypting the account holder's credentials using one or more processors;

storing the account holder's encrypted credentials using one or more processors, wherein the encrypted credentials are stored in a data store;

….

35. The Examiner advised again that "the pending claims recite simple steps of receiving and sending information" and "encrypting and verifying

received data" which "is done with generic computer equipment." *Id*., 476 (Advisory Action dated 12/09/2021).

36. Before appeal, the applicants alleged that the claims "can only be implemented using specialized computer hardware" to encrypt login credentials. *Id*., 489 (Pre-Appeal Brief Request for Review dated 1/30/2022).

37. The Examiner issued a Notice of Allowance on March 11, 2022, cancelling claims 2 and 7 and amending claims 1, 3–6, and 8–10 by adding the "endpoint" elements originally recited in claims 2 and 7 to claims 1 and 6, pursuant to the applicants' authorization in a February 4, 2022 e-mail. *Id*., 498-501 (Notice of Allowance dated 3/11/2022). The amended claim 1 is reproduced below:

> A method for permitting a third-party entity to gain access, in a secure manner, to an online account held by an account holder that is secured by a multi-factor authentication protocol, comprising the following steps performed by the third-party entity:
>
> collecting the account holder's credentials;
>
> encrypting the account holder's credentials using one or more processors;
>
> storing the account holder's encrypted credentials using one or more processors, wherein the encrypted credentials are stored in a data store;

verifying the account holder's credentials by accessing, via one or more processors, the online account using the account holder's credentials;

receiving a multi-factor authentication request from the online account, the multi-factor authentication request indicating that a response is required from the account holder before further access to the online account is required;

prompting the account holder for a response and receiving the response thereto from the account holder; [[and]]

transmitting the account holder's response to the online account to satisfy the multi-factor authentication request;

storing one or more of the account holder's endpoints for the online account in a data store associated with the online account; and

utilizing said one or more endpoints to permit the third-party to gain access to the online account, or information pertaining to the online account, without requiring a response by the account holder to any subsequent multi-factor authentication requests.

38. Based on the above amendments, the Examiner concluded that the claims are patent-eligible and that Counterman did not teach each element of the claims. *Id*., 501-03.

39. However, the Examiner issued a Corrected Notice of Allowance on March 25, 2022 allowing currently challenged claims 1–10, as presented prior to the Examiner's amendment, with no explanation about the above amendments. *Id*., 543 (Corrected Notice of Allowance dated 3/25/2022).

40. In sum, the applicants did not dispute that the prior art Counterman taught the use of essentially every claim element, except the steps of collecting, storing, and encrypting credentials, and utilizing endpoints to gain access to online accounts without user intervention.

V. CLAIM CONSTRUCTION

41. I understand that, for purposes of my analysis in this *inter partes* review proceeding, the terms appearing in the patent claims should be interpreted according to their "ordinary and customary meaning." In determining the ordinary and customary meaning, I understand that the words of a claim are first given their plain meaning that those words would have had to a person of ordinary skill in the art at the time of the alleged invention. I also understand that the structure of the claims, the specification, and file history may be used to better construe a claim. Moreover, I understand that even treatises and dictionaries may be used, albeit under limited circumstances, to determine the meaning attributed by a person of ordinary skill in the art to a claim term.

42. As detailed in my analysis below, the challenged claims are obvious under any reasonable interpretation of the claim language.

43. For purposes of this declaration, I've been asked to assume that the time of the alleged invention is January 28, 2014 (the priority date of the '090 patent).

VI. THE CHALLENGED CLAIMS ARE UNPATENTABLE

44. In my analysis of the '090 patent and relevant art, I interpreted the challenged claim language according to their "ordinary and customary meaning."

45. Based on my reading of the '090 patent's specification and the challenged claims, it is my conclusion that the prior art teaches each limitation of the challenged claims under any reasonable interpretation of the claim language.

46. The challenged claims recite and claim features that were obvious in view of the art prior to the earliest priority date.

A. Overview of Prior Art

47. The idea of using multi-factor authentication to enhance security of an online account was known long before the '090 patent. For example, Anderson II teaches that: "A much more visible use of challenge-response is in **two-factor authentication**. Many organizations issue their staff with password generators to let them log on to corporate computer systems .... When you want to log in to a machine on the network, **you call up a logon screen and are presented with a**

random challenge of maybe seven digits. You key this into your password generator, together with a PIN of maybe four digits. The device encrypts these eleven digits using a secret key shared with the corporate security server, and displays the first seven digits of the result. You enter these seven digits as your password .... If you had a password generator with the right secret key, and you entered the PIN right, and you typed in the result correctly, then the corporate computer system lets you in." Exhibit C, at 71 (emphasis added).

48. In particular, the financial industry had been using multi-factor authentication for years before the '090 patent, including sending authentication codes through SMS text messages (i.e., "endpoints"). *See* Exhibit C, at 344 ("A recent development has been the 'Verified by VISA' program under which merchants can refer online credit-card transactions directly to the issuing bank, which can then authenticate the cardholder using its preferred method .... At the top of the scale are banks that use **two-channel authentication**: when you buy online **you get a text message saying something like 'If you really want to pay Amazon.com $76.23, enter the code 4697 in your browser now'**.") (emphasis added).

49. The use of multi-factor authentication has created obstacles for automated systems seeking to access online accounts on behalf of users. In response, techniques and methods of using a third-party entity to gain access to

online accounts secured by multi-factor authentication were developed prior to the '090 patent. For example, the Examiner found that this technique was disclosed in Counterman (U.S. Patent No. 8,495,720). Ex-1002, 329-30. Indeed, this idea was merely a reiteration of adding a third-party intermediary to pass along multi-factor authentication requests between the online accounts and the user. Multiple other prior art references, including Hsu (Ex-1004), Hazlehurst (Ex-1005), and Emerson (Ex-1006), all expressly teach the same techniques of using a third-party entity to process authentication requests, just like those described in the '090 patent. *See, e.g.*, Ex-1004, [0023] ("**an application** executing on an electronic device may request and receive **multi-factor authentication** information one or more times from the user. For example, the application may include a financial application, such as Quicken™, TurboTax™, or other software **capable of receiving financial-related data**, bank statements, and/or investment records") (emphasis added); Ex-1005, Abstract, 3:33-34 ("**the Pin Vault software** cooperates with an Internet service to provide, from the Pin Vault, data as needed by the Internet service to log onto and **authenticate as the user at the one or more web** sites storing information of interest to the user, to gather information on behalf of the user," and the software "can perform **two-factor authentication** on behalf of a user") (emphasis added); Ex-1006, [0053], [0060] ("**The ODS agent 114** is a programming module that provides other network components with programming

interfaces to **on-line banking authentication** and fund transfer services through a specific financial institution"; "[O]nce a user provides his financial institution identification information, **the ODS agent 114 conducts a user-free (i.e., user's further input is not needed) communication session with the financial institution 106**"; "**ODS agent 114** ... may **cache (device) tokens set by on-line banking to pass multi-factor authentication** when applicable") (emphasis added).

50. Likewise, there was nothing new about collecting, storing, and encrypting a user's login credentials to access the user's online account. In fact, storing login credentials in encrypted form has been standard operational practice since at least the 1970s, when it was included in the earliest versions of the UNIX operating systems at Bell Labs. *See* Exhibit D, at 594-95 ("The obvious solution is to arrange that the passwords not appear in the system at all, and it is not difficult to decide that this can be done by **encrypting each user's password, putting only the encrypted form in the password file**, and throwing away his original password (the one that he typed in).") (emphasis added). Moreover, it has been mandated in many sectors including the financial industry for decades. As one example, Anderson I states that in 2001 VISA required all its merchants to encrypt stored and transmitted data. *See* Exhibit B, at 397 ("Credit card numbers are indeed available on the Net, but usually because someone hacked the computer of a

merchant who disobeyed the standard bank prohibition against retaining customer credit card numbers after being paid. (As this book was going to press, VISA announced that, starting in 2001, all its merchants will have to obey 10 new security rules; for example, they must install a firewall, keep security patches up to date, **encrypt stored and transmitted data**, and regularly update antivirus software.)") (emphasis added). According to Schneier, the financial industry had developed various standards based on Data Encryption Standard (DES) to encrypt client credentials such as passwords years before the '090 patent. *See* Exhibit E, at 374-75 ("ANSI's Financial Institution Retail Security Working Group developed a **standard for the management and security of PINs** (ANSI X9.8) and another DES-based standard for the authentication of retail financial messages (ANSI X9.19). The group has a draft standard for secure key distribution (ANSI X9.24).") (emphasis added). There were many commercial processors designed to implement these encryption standards. *See* Exhibit E, at 387, 390-91 ("Table 12.9 gives the specifications for some commercial DES chips"; "Table 12.10 gives actual results and estimates for various Intel and Motorola microprocessors").

51. Hsu and Hazlehurst also expressly disclosed the method of collecting, storing, and encrypting user credentials for access to an online account. Ex-1004, [0039] ("The multi-factor authentication information and/or the retrieved information may be a sensitive nature. As a consequence, in some embodiments

stored authentication information and/or stored retrieved information **are encrypted**.") (emphasis added); Ex-1005, 4:48-67 ("the client-side PVA application **112** contains an **encrypted user pin vault (UPV) file 113** that is adapted to store at minimum all of the user's subscribed-to Web site locations and appropriate authentication credentials for accessing those sites .... PVA **114** has the appropriate **encryption/decryption software (SW)** to decrypt and use the user authentication credentials from UPV **113**.") (emphasis added).

52. Similarly, there also was nothing new about using an "endpoint" such as a phone number to receive multi-factor authentication codes in order to access online account. As discussed above, banks had been using this technique to verify credit card holders years before the '090 patent. *See* Exhibit D, at 344 ("banks that use **two-channel authentication**: when you buy online **you get a text message saying something like 'If you really want to pay Amazon.com $76.23, enter the code 4697 in your browser now'**.") (emphasis added).

53. Multiple prior art references expressly taught the technique of storing endpoints at a third-party entity for bypassing subsequent multi-factor authentication requests. For example, Nakonetzki (U.S. Patent Appl. No. 2013/0276080, Exhibit F) discloses that an application can store an "address identifier" (e.g., MSISDN or Mobile Station International Subscriber Directory Number, the technical term for a fully specified phone number including country

and area codes) and use the identifier to gain access to a server without the user responding to subsequent multi-factor authentication requests, as the application can automatically initiate communication with the server using the one-time password received via SMS message. *See* Exhibit F, [0058] ("The **address identifier** can be stored in the communication terminal **1**, for example in the identification module **5**. Subsequently, that means with the address identifier being present, the application **3** establishes an initiation request for initiating a session with the service **4** .... Subsequently, a request specific key, in particular a **one-time-password (OTP)** is transmitted to the communication terminal **1** from the service **4** .... It is particularly preferred that the transmission is thus performed in an **SMS message** .... The SMS is sent from the service server **2** or the service **4**, respectively, to the MSISDN which was contained in the initiation request. After receipt of this message, which contains the request specific key, in particular the OTP, at the communication terminal **1**, **this key is automatically read out at the communication terminal 1 and is provided to the application 3**.") (emphasis added). As explained below, examples like Peotta and 2FA Q&A expressly taught that a third party (i.e., attacker) may use an endpoint (i.e., phone number) to bypass multi-factor authentication requests from banks. *See* Ex-1007, 5 (noting that "[t]he attacker may alter the **cellular phone number** to which the authorization methods are sent" is a vulnerability of the "Short Message Service" used by "Internet

Banking Systems") (emphasis added); Ex-1009, 1 (Question: "When I attempt to log in to my bank, an SMS code is sent to my phone .... How could it be exploited?"; Answer: "Social engineer your bank to change your **mobile phone** details to a phone [the attackers] control.") (emphasis added).

54. In short, there is nothing inventive in the claims of the '090 patent. Hsu, Hazlehurst, Emerson, Peotta, and 2FA Q&A are specific examples of a wealth of literature that teaches the very technique for gaining access to online accounts secured by multi-factor authentication that the '090 patent later relied on to get the claims allowed. As discussed in further detail below, these references demonstrate that the challenged claims are not patentable.

B. Ground 1: Claims 1, 4-6, and 9-10 are anticipated by Hsu

1. Overview of Hsu (Ex-1004)

55. Hsu was filed on October 31, 2006 and published on May 15, 2008, which I understand makes it prior art under AIA 35 U.S.C. 102(a)(1).

56. Hsu is directed to "techniques for collecting and providing authentication information." Ex-1004, [0001]. Hsu explains that authentication is a "widely used procedure[]" that "**enable[s] a user to access an application or system (by confirming the user's identity) and to verify the authority of the user to perform certain operations or tasks**." *Id.*, [0002]. As an example, Hsu discloses that a user may provide information, such as a username and password, to

confirm the user's identity. *Id.* Hsu further explains that "[a]s security threats continue to grow, many applications and systems are significantly increasing such protection requirements", and as a result, many applications and systems utilize **multiple authentication factors to perform authentication (also referred to as multi-factor authentication)**." *Id.*, [0003]. Such multi-factor authentication may include "something the user knows (for example, a password), something the user has (for example, a token), and/or something the user is (for example, a biometric feature)." *Id.*

57. Hsu describes a "**computer system 100**" that includes computers and servers that are networked together, where one or more users may provide multi-factor authentication information to a program, such as a financial program, that executes on computer **110**. *Id.*, [0027].

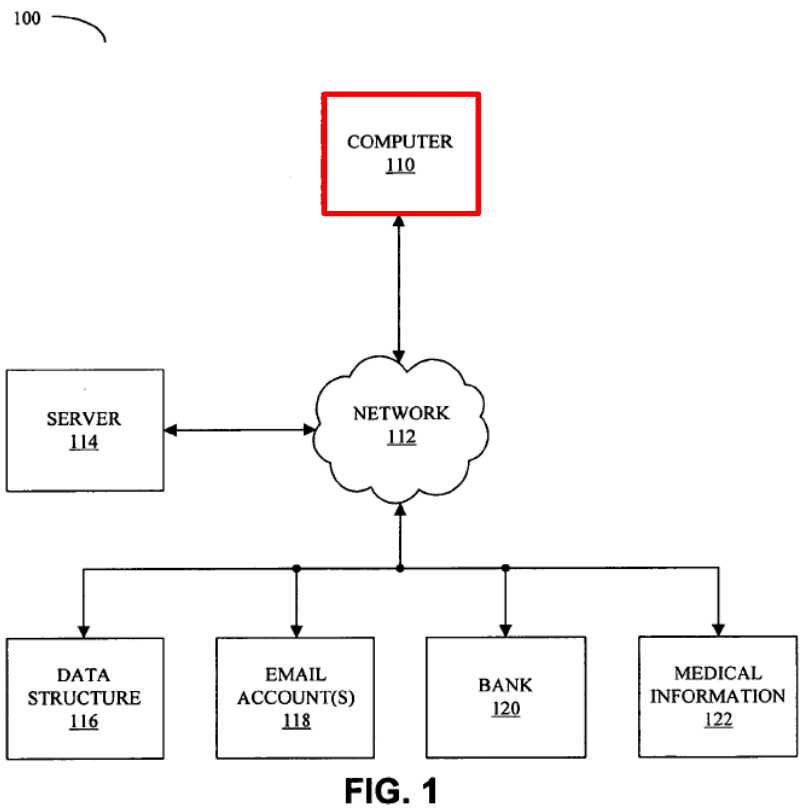


FIG. 1

Hsu (Ex-1004), FIG. 1 (Annotated)

58. As shown in FIG. 2, Hsu further discloses a "**computer system 200**" that includes "one or more processors **210**, a communication interface **212**, a user interface **214**, and one or more signal lines **222** coupling these components together." *Id.*, [0040]. "[T]he user interface **214** may include a display **216**, a keyboard **218**, and/or a pointer **220**, such as a mouse." *Id.*

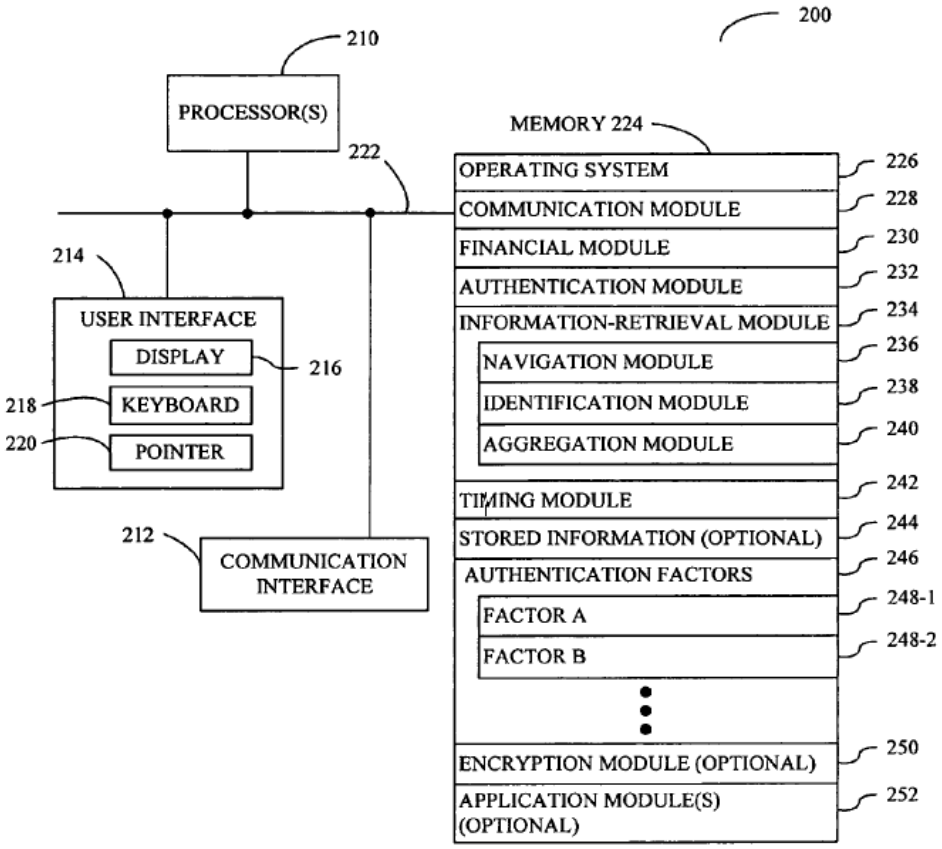


FIG. 2

Hsu (Ex-1004), FIG. 2

2. Claim 1

a) *[1pre] A method for permitting a **third-party entity** to **gain access, in a secure manner, to an online account held by an account holder** that is **secured by a multi-factor authentication protocol**, comprising the following steps performed by the third-party entity:*

59. To the extent this preamble term is limiting, Hsu discloses it.

60. Hsu discloses a method for permitting a third party to gain access to an online account that is secured by a multi-factor authentication protocol. Ex-

1004, [0023]. For instance, Hsu discloses a "**computer system**" that "may be used to **retrieve information, such as financial information for a user** (for example, banking information), information associated with multiple email accounts for the user, and/or medical information for the user." *Id.* (emphasis added).

61. The computer system in Hsu is usable by a third party , such as a financial program or financial application, to gain access to the user's online account. *Id.*, [0031] ("**the financial program** may **access one or more documents (such as one or more websites or web pages on one or more hosts) and may retrieve stored information (such as financial information) for the user**.") (emphasis added); *see also id.*, [0023] ("For example, the application may include a **financial application**, such as Quicken™, TurboTax™, or other software capable of receiving financial-related data, bank statements, and/or investment records.") (emphasis added); *id.*, [0002] ("the user may provide information, such as a username, a password, or a pin number during these procedures to **confirm the user['s] identi[t]y** (authorization) and/or **the user's right to transfer funds from a bank account** (authorization)") (emphasis added).

62. The user in Hsu is an account holder. As the following commands show, the financial program (i.e., Customer Central or CC) requests the user to answer a question (i.e., "In what city were you born?") for the financial program to "[d]iscover" the user's "[a]ccount[]" (i.e.,

"CC**DiscoverAccounts**InteractiveResponse") from the bank 120. *Id*., [0035] (emphasis added).

```
<?xml version="1.0" encoding="UTF-8"?>
<cc:CCWSResponse xmlns:cc="http://www.intuit.com/CustomerCen-
tral">
<status>
<code>ok</code>
<string>call successful</string>
</status>
<body>
<ccresp:CCDiscoverAccountsInteractiveResponse
   xmlns:ccresp="http://www.intuit.com/CustomerCentral/Responses">
<session>
<cccaptureIpAddress>172.23.29.76</cccaptureIpAddress>
<cccapturePort>9909</cccapturePort>
<ccscrapeIpAddress>172.23.29.76</ccscrapeIpAddress>
<ccscrapePort>9979</ccscrapePort>
<ccscriptInstanceId>-208666287</ccscriptInstanceId>
</session>

<questions>
<question>
<text>In what city were you born? (Enter full name of city only)</text>
</question>
</questions>
</ccresp:CCDiscoverAccountsInteractiveResponse>
</body>
</cc:CCWSResponse>.
```

Hsu (Ex-1004), [0035] (Annotated)

63. Moreover, Hsu indicates that the "**stored authentication information and/or stored retrieved information are encrypted**" and that "such **information may be encrypted when it is communicated over the network 112**", i.e., in a secure manner. *Id.,* [0039] (emphasis added).

64. The online account is secured by a MFA protocol, and therefore the user must "provide **multi-factor authentication** information to a program, such as a financial program," so that the financial program can "assemble (i.e., retrieve) information for the user". *Id*., [0027], [0038] (emphasis added). That is, Hsu discloses "[u]sing the **multi-factor authentication**, [to enable] the **financial program** [to] **access one or more documents (such as one or more websites or web pages on one or more hosts) and may retrieve stored information (such as financial information) for the user**." *Id*., [0031].

b) *[1a] collecting the account holder's credentials;*

65. Hsu discloses this limitation. Hsu discloses that the system "may request and receive multi-factor authentication information one or more times from the user." Ex-1004, [0023]. This multi-factor authentication information includes user's credentials, "such as a social security number, one or more usernames, one or more passwords, one or more pins, one or more telephone numbers, one or more addresses, and/or additional personal information." *Id.*, [0029]; *see also id.*, [0002] ("Authentication … enable a user to access an application or system (by confirming the user's identity)"). As explained above with respect to element [1pre], the user in Hsu is an account holder. *Id*., [0035] ("CCDiscoverAccountsInteractiveResponse").

66. Further, Hsu provides in FIG. 1 a block diagram illustrating a "computer system 100 that includes a number of computers and servers that are networked together in accordance with an embodiment of the present invention." *Id.*, [0027].

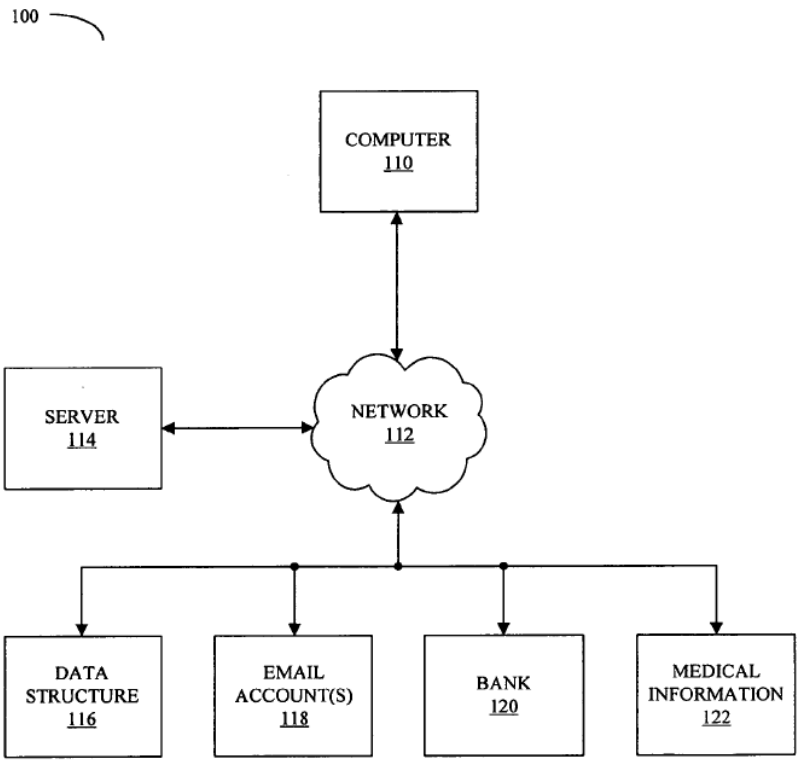


FIG. 1

Hsu (Ex-1004), FIG. 1

67. Hsu discloses that account holders/users "may provide multi-factor authentication information to a program, such as a financial program, that executes on computer **110**." *Id.*, [0027]. The "authentication information for one or more users **510** of the financial program" may be stored in a data structure **500**. *Id.*, [0051] ("This data structure may include authentication information for one or

more users **510** of the financial program. For example, for user **510-1**, the authentication information may include a user name **512-1**, a password **514-1**, personal information **516-1**, and/or an RSA token **518-1**.").

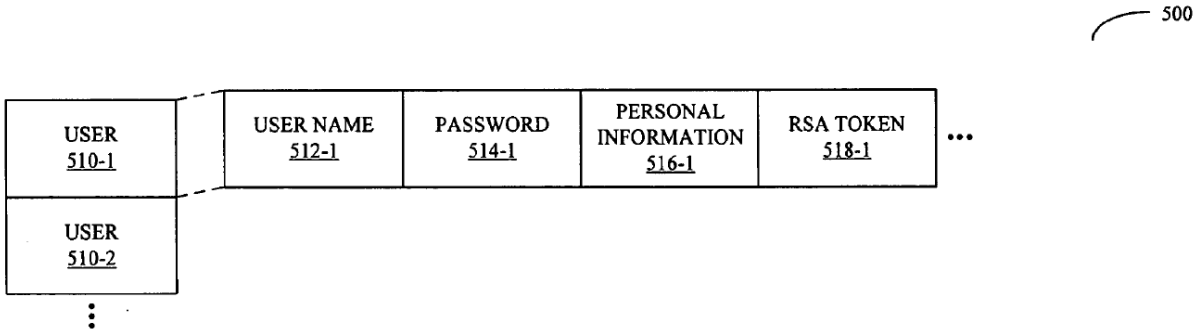


FIG. 5

Hsu (Ex-1004), FIG. 5

c) *[1b] encrypting the account holder's credentials using one or more processors;*

68. Hsu discloses this limitation. For instance, Hsu explains that the multi-factor authentication information and/or the retrieved information "may be a sensitive nature" and as a result, the "**stored authentication information** and/or stored retrieved information **are encrypted**." Ex-1004, [0039] (emphasis added). Further, Hsu discloses an "**encryption module** (or a set of instructions) **250**" (*Id.* [0043] (emphasis added)) that execute on "**one or more processors 210**". *See Id.* [0040] and Fig. 2. Thus, the encrypting occurs using one or more processors as claimed.

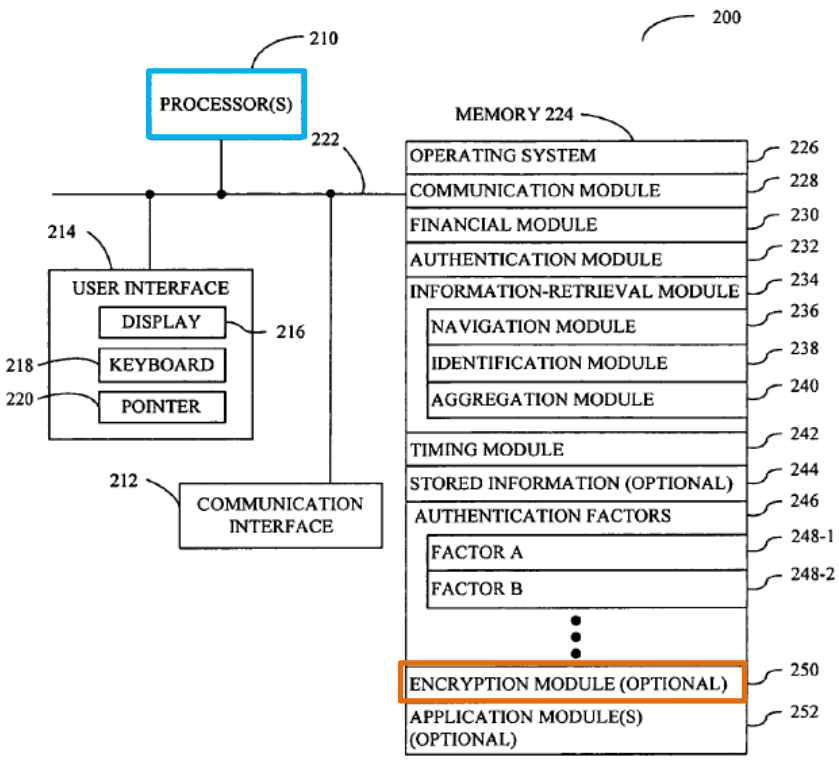


FIG. 2

Hsu (Ex-1004), FIG. 2 (Annotated)

d) *[1c] storing the account holder's encrypted credentials using one or more processors, wherein the encrypted credentials are stored in a data store;*

69. Hsu discloses this limitation. Specifically, Hsu discloses that the system "**stores** the information and/or the **multiple authentication factors** on the first host." Ex-1004, [0008] (emphasis added). Further, the "stored authentication information and/or stored retrieved information are encrypted." *Id*., [0039].

In addition, Hsu discloses that the multi-factor authentication information "may be initially **stored** locally on the computer **110** or remotely, for example, on the server **114**, **in a data structure 116**," i.e., a data store. *Id*., [0031] (emphasis

added); *see also id.*, [0026] ("the information and/or multi-factor authentication information may be stored locally (for example, on a local computer) and/or remotely (for example, on a computer or server that is accessed via a network)."); *id.*, [0029] ("Such static information may be stored locally (i.e., on the computer **110**) and/or remotely (for example, on the server **114**)."); [0041] ("In addition, the system may optionally store the multiple authentication factors and/or the information (**430**)."). As shown in Figure 5, "this data structure 500 may include authentication information for one or more users **510** of the financial program," such as "user name," "password," and "personal information." *Id.*, [0051].

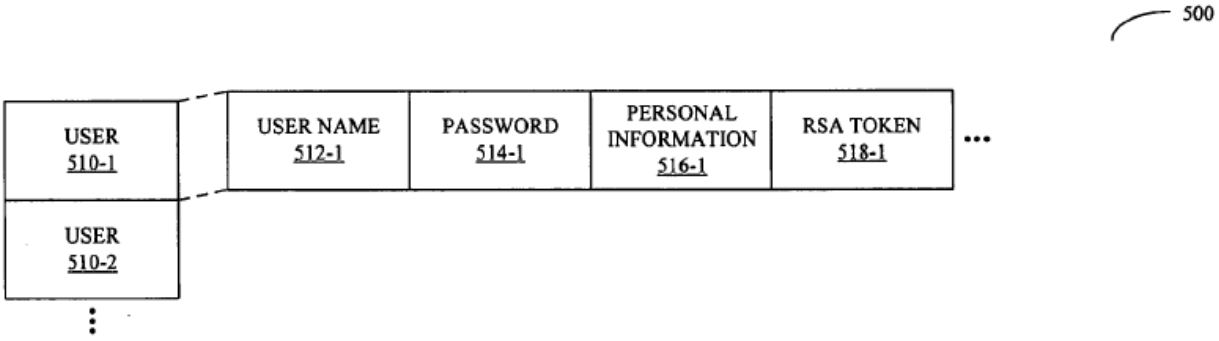


FIG. 5

Hsu (Ex-1004), FIG. 5

70. The storage process is performed using "**one or more processors 210**" on the computer. *Id.*, [0040] ("The computer system **200** includes one or more processors **210**"); *see also id.*, [0041] ("Memory **224** may store an operating system **226** that includes procedures (or a set of instructions) for handling various basic system services for performing hardware dependent tasks."); *id.*, [0044]

("Instructions in the various modules in the memory **224** may be implemented in a … programming language …. The programming language may be … executed by the one or more processing units 210.").

e) *[1d] verifying the account holder's credentials by accessing, via one or more processors, the online account using the account holder's credentials;*

71. Hsu discloses this limitation. As explained above, Hsu "uses the multiple authentication factors to access [a] document". *Id*., [0006]. Said document resides "on a second host" (*id.*), i.e., "remotely, for example on the server 114" (*id*., [0031]) which is shown to connect "via network 112". *See id.*, [0028] and Fig. 1. A POSITA would understand that using authentication factors to access a document on a remote host requires accessing the online account with which both the authentication factors and document are associated. *See id*., [0006] ("These authentication factors are associated with a document on a second host, and include authentication information that enables access to the document"). For example, the system may request the user to answer a challenge question in order to "[d]iscover" the user's "[a]ccount[]" and then to "request[] information from the bank 120." Ex-1004, [0035]. As explained above, the user in Hsu is an account holder. *Id*. ("CCDiscoverAccountsInteractiveResponse").

72. Hsu discloses that the account holder/user provides multi-factor authentication information, which includes the account holder/user's usernames,

passwords, telephone numbers, pins, or additional information, to "enable [the] user to access an application or system (by confirming the user's identity) and to verify the authority of the user to perform certain operations or tasks." *Id.*, [0002], [0027]-[0029]. Hsu further explains that "[u]sing the multi-factor authentication information, the financial program may access one or more documents (such as one or more websites or web pages on one or more hosts) and may retrieve stored information (such as financial information) for the user." *Id.*, [0031]. A POSITA would have understood that the account holder/user's documents and information could only be accessed after the account holder/user's credentials (e.g., usernames and passwords) were verified.

73. The online account is accessed via "one or more processors 210". *Id*., [0040] ("The computer system 200 includes one or more processors 210"); *see also id.*, [0041] ("The memory 224 may also store procedures (or a set of instructions) in a communication module 228. The communication procedures may be used for communicating with one or more computers and/or servers, including computers and/or servers that are remotely located with respect to the computer system 200."); *id.*, [0044] ("Instructions in the various modules in the memory 224 may be implemented in a … programming language …. The programming language may be … executed by the one or more processing units 210.").

f) *[1e] receiving a multi-factor authentication request from the online account, the multi-factor authentication request indicating that a response is required from the account holder before further access to the online account is granted;*

74. Hsu discloses this limitation. As an example, Hsu discloses that after the third party (e.g., financial program in Hsu) requests information from the online account (e.g., the bank in Hsu), the online account requires additional information (e.g., a multi-factor authentication request) for further access to the information from the online account. Ex-1004, [0035]; *see also id.*, [0024] ("The application may utilize such multi-factor information to access a document (such as a website or web page) that is resident on a server computer."). As shown below, "CCDiscoverAccountsInteractiveResponse" means that the financial program (i.e., Customer Central or CC) requests the account holder/user to answer a question (city of birth) in order to "[d]iscover" (i.e., access) the account holder/user's "account[]" from the bank. *Id.*, [0035]. Hsu provides the following commands to demonstrate this request:

<?xml version="1.0" encoding="UTF-8"?>
<cc:CCWSResponse xmlns:cc="http://www.intuit.com/CustomerCen-
tral">
<status>
<code>ok</code>
<string>call successful</string>
</status>
<body>
<ccresp:CCDiscoverAccountsInteractiveResponse
xmlns:ccresp="http://www.intuit.com/CustomerCentral/Responses">
<session>
<cccaptureIpAddress>172.23.29.76</cccaptureIpAddress>
<cccapturePort>9909</cccapturePort>
<ccscrapeIpAddress>172.23.29.76</ccscrapeIpAddress>
<ccscrapePort>9979</ccscrapePort>
<ccscriptInstanceId>-208666287</ccscriptInstanceId>
</session>
<questions>
<question>
<text>In what city were you born? (Enter full name of city only)</text>
</question>
</questions>
</ccresp:CCDiscoverAccountsInteractiveResponse>
</body>
</cc:CCWSResponse>.

Hsu (Ex-1004), [0035] (Annotated)

75. As seen in the red box above, the bank requires a response from the user to confirm which city the user was born, which Hsu explains is a type of multi-factor authentication (e.g., "something the user knows"). *Id.*, [0003], [0035]; *see also id.*, [0036] ("In another illustrative example, the bank **120** (such as ING bank) requires authentication information. In this example, the financial program may either request this authentication information from the user or may retrieve the answer from storage. Then, the financial program responds.").

g) *[1f] prompting the account holder for a response and receiving the response thereto from the account holder; and*

76. Hsu discloses this limitation. Specifically, Hsu discloses that the answer to the requested authentication information (i.e., "In what city were you born?") could be prompted/requested from the account holder/user with the response being provided thereto. Ex-1004, [0035] ("The financial program may either request the authentication information (city of birth) from the user or may retrieve the answer (Palo Alto) from storage.").

h) *[1g] transmitting the account holder's response to the online account to satisfy the multi-factor authentication request.*

77. Hsu discloses this limitation. Hsu discloses that after the account holder/user provides the response (i.e., "Palo Alto") to the online account's request (i.e., "In what city were you born?"), the information is then transmitted using the following command:

<?xml version="1.0" encoding="utf-8" ?> <cc:CCWSRequest
xmlns:cc="http://www.intuit.com/CustomerCentral">
<authentication><tp_
partner_id>3</tp_partner_id><userId>
ezQwQTgzNkIxLTdGRkItNDJBM
C05RDc5LUJBOTc3MTcyMEY0NX0=</userId><password>X</
password></authentication><body><ccreq:
CCDiscoverAccountsInteractiveRequestxmlns:ccreq="http://
www.intuit.com/CustomerCentral/Requests"><session>
<cccaptureIpAddress>172.23.29.76</cccaptureIpAddress>
<cccapturePort>9909</cccapturePort><ccscrapeIpAddress>
172.23.29.76
</ccscrapeIpAddress><ccscrapePort>9979</ccscrapePort>
<ccscriptInstanceId>208666287<ccscriptInstanceId></
session><answers
>
<answer>PaloAlto</answer></answers>
</ccreq:CCDiscoverAccountsInteractiveRequest></body>
</cc:CCWSRequest>AccountsInteractiveRequest></body>
</cc:CCWSRequest>.

Hsu (Ex-1004), [0035] (Annotated)

3. Claim 4

a) *[4] The method of claim 1, wherein the account holder's credentials are collected via a user interface.*

78. Hsu discloses this limitation. Hsu discloses the use of a user interface to collect the account holder/user's credentials. Ex-1004, [0040] ("The computer system **200** includes one or more processors **210**, a communication interface **212**, ***a* user interface 214**, and one or more signal lines **222** coupling these components together. … Moreover, the **user interface 214** may include a display **216**, a keyboard **218**, and/or a pointer **220**, such as a mouse") (emphasis added). The user interface in the Hsu system "receives the multiple authentication factors from the user", which include the account holder/user's credentials. *Id*., [0047]; *see also id*.,

[0029] ("The multi-factor authentication information provided by the user may include ... one or more usernames, one or more passwords, ... and/or additional personal information).

4. Claim 5

a) *[5] The method of claim 1, wherein the account holder's credentials are collected from a data store.*

79. Hsu discloses this limitation. Hsu discloses that the user's credentials are stored, and thereby retrieved/collected, from a "**data structure 116**", i.e., a data store. Ex-1004, [0031] ("The **information to be retrieved** may be initially stored locally on the computer **110** or remotely, for example, on the server **114**, in **a data structure 116**") (emphasis added); *see also id*., [0035] ("The financial program may either request the authentication information (city of birth) from the user or may **retrieve the answer** (Palo Alto) **from storage**.") (emphasis added); [0036] ("In another illustrative example, the bank 120 (such as ING bank) requires authentication information. In this example, the financial program may either request this authentication information from the user or may **retrieve the answer from storage**. Then, the financial program responds.") (emphasis added).

5. Claim 6

a) *[6pre] A method for permitting a third-party entity to gain access, in a secure manner, to an online account held by an account holder that is secured by a multi-factor authentication protocol, comprising the following steps performed by the third-party entity:*

80. As discussed above for limitation [1pre], to the extent this preamble term is limiting, Hsu discloses it.

b) *[6a] collecting the account holder's credentials;*

81. As discussed above for limitation [1a], Hsu discloses this limitation.

c) *[6b] encrypting the account holder's credentials using one or more processors;*

82. As discussed above for limitation [1b], Hsu discloses this limitation.

d) *[6c] temporarily storing the account holder's encrypted credentials using one or more processors, wherein the encrypted credentials are held in memory;*

83. As discussed above for limitation [1c], Hsu teaches this limitation.

84. In addition, Hsu discloses temporarily storing the encrypted credentials in memory. Ex-1004, [0041] ("Memory 224 in the computer system 200 may include **volatile memory and/or non-volatile memory**. More specifically, memory 224 may include ROM, RAM, EPROM, EEPROM, FLASH, one or more smart cards, one or more magnetic disc storage devices, and/or one or more optical storage devices.") (emphasis added); *see also id*., [0043] ("memory 224 includes optional stored information 244"); [0049] ("the system may

optionally **store the multiple authentication factors** and/or the information") (emphasis added); [0039] ("The multi-factor authentication information and/or the retrieved information may be a sensitive nature. As a consequence, in some embodiments stored authentication information and/or stored retrieved information are **encrypted**.") (emphasis added). Volatile memory, such as random-access memory (RAM), stores the account holder's credentials only temporarily. Ex-1010, at 209 ("Nonvolatile memory retains its values when power is removed. Volatile memory loses its contents when power is removed. ... RAM is volatile.").

e) *[6d] verifying the account holder's credentials by accessing, via one or more processors, the online account using the account holder's credentials;*

85. As discussed above for limitation [1d], Hsu discloses this limitation.

f) *[6e] receiving a multi-factor authentication request from the online account, the multi-factor authentication request indicating that a response is required from the account holder before further access to the online account is granted;*

86. As discussed above for limitation [1e], Hsu discloses this limitation.

g) *[6f] prompting the account holder for a response and receiving the response thereto from the account holder; and*

87. As discussed above for limitation [1f], Hsu discloses this limitation.

h) *[6g] transmitting the account holder's response to the online account to satisfy the multi-factor authentication request.*

88. As discussed above for limitation [1g], Hsu discloses this limitation.

6. Claim 9

a) *[9] The method of claim 6, wherein the account holder's credentials are collected via a user interface.*

89. As discussed above for claim 4, Hsu discloses this limitation.

7. Claim 10

a) *[10] The method of claim 6, wherein the account holder's credentials are collected from a data store.*

90. As discussed above for claim 5, Hsu discloses this limitation.

C. Ground 2: Claims 1, 4-6, and 9-10 are anticipated by Hazlehurst

1. Overview of Hazlehurst (Ex-1005)

91. Hazlehurst was filed on April 25, 2008 and published on October 29, 2009, which I understand makes it prior art under AIA 35 U.S.C. 102(a)(1).

92. Hazlehurst is directed to "systems and methods for gathering summary data and authorizing Web transactions." Ex-1005, 1:14-16. Hazlehurst explains that "it is known that proxy servers exist and that a proxy service may perform tasks on behalf of a subscribing user". *Id.*, 1:19-21. As an example, Hazlehurst discloses that a user can "pre-provide[] the login data to the service in association with any site that requires the information", enabling "automatic logins". *Id.*, 1:41-48. But "recently, security regimes have been developed to elevate the level of security provided over the traditional username/personal identification number or password combination login credentials", such as "Web token generators" and "CAPTCHA". *Id.*, 1:53-2:20. These developments "present

a challenge" for automated services because the existing technology "has no way of inputting or utilizing the unique user token data without the user's direct input." *Id.*, 2:21-29.

93. To address this challenge, Hazlehurst teaches a system that "leverage[s] Two-factor authentication" "when it requires at least two of the authentication form factors". *Id.*, 2:39-41. These "universally recognized factors" include: "(1) Something a person knows, such as a password, a PIN or an out-of-wallet-response. (2) Something a person has, such as a mobile phone, a credit card or a hardware security token. (3) Something a person is, such as represented by a fingerprint, by a retinal scan, or by some other biometric." *Id.*, 2:30-38.

94. Hazlehurst provides in Fig. 1 a block diagram illustrating "a system that can perform **two-factor authentication** on behalf of a user." *Id.*, 3:33-34 (emphasis added). The system includes a service provider **103** that provides transaction services for users, a **portal server (PS) 105** through which transaction services are accessible to the users, a user domain 107 that subscribes to services offered by service provider **103**, and a personal device that is used by the user to access **PS 105** for services. *Id.*, 3:65-4:11. According to input from user, **PS 105** may **log in on behalf of the user and access data from Web sites maintained in servers 104 or 109** and even perform certain transactions at those sites. *Id.*, 4:37-42.

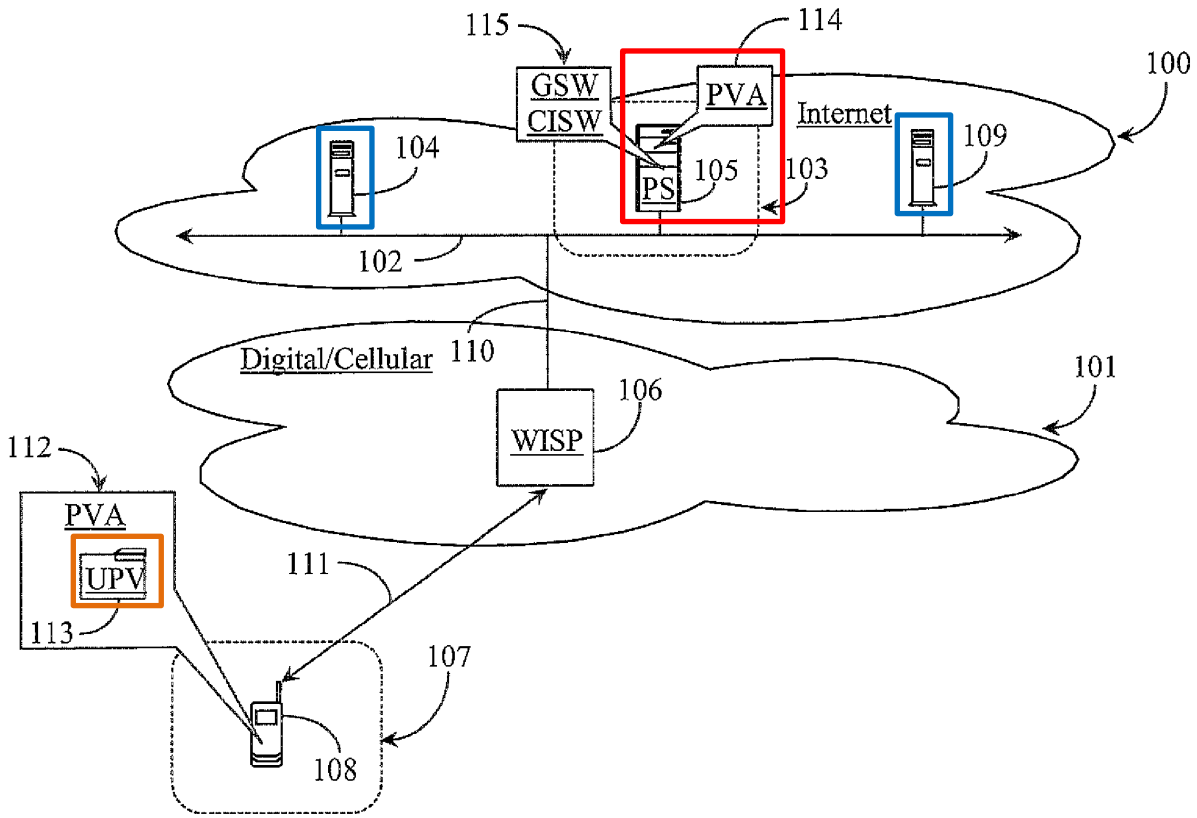


Fig. 1

Hazlehurst (Ex-1005), Fig. 1 (Annotated)

95. Hazlehurst further discloses the use of a software application termed "**Pin Vault Application (PVA)**" on both client side and server side. *Id.*, 4:43-46. "[T]he client-side PVA application 112 contains an **encrypted user pin vault (UPV) file 113**" that stores the user's authentication credentials. *Id.*, 4:48-52 and Fig. 2. "[W]hen **PS 105** is performing … services for the user, it accesses **UPV 113** … **to obtain the required data in order to access the appropriate Web sites**". *Id.*, 4:61-65. In the case that **PS 105** receives a **two-factor authentication**

request such as CAPTCHA, it submits the request to device **108** for the user to type in the required characters. *Id.*, 5:38-47.

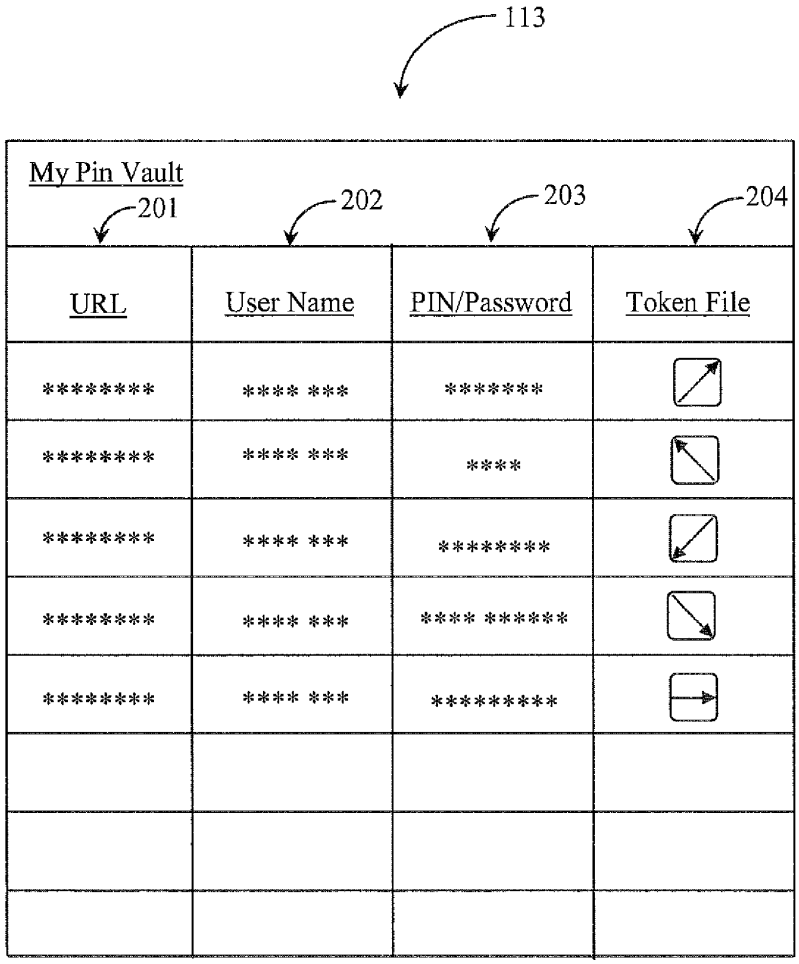


Fig.2

Hazlehurst (Ex-1005), Fig. 2

2. Claim 1

a) *[1pre] A method for permitting a **third-party entity** to **gain access, in a secure manner, to an online account held by an account holder** that is **secured by a multi-factor authentication protocol**, comprising the following steps performed by the third-party entity,*

96. To the extent this preamble term is limiting, Hazlehurst discloses it.

97. Hazlehurst discloses a method for permitting a "**Pin Vault software**" (i.e., third party entity) to gain access to a "**Web-based account**" (i.e., online account) that requires account holder/user's credentials for login (i.e., secured by a multi-factor authentication protocol). Ex-1005, Abstract (emphasis added), 1:59-61 ("**a Web-based account** or service may **require the user to log in and provide, in addition to normal user credentials, a token that changes in value, say every 60 seconds**.") (emphasis added). For instance, Hazlehurst discloses a system that "has a Pin Vault software" (i.e., PVA) where "**the Pin Vault software** cooperates with an Internet service to provide, from the Pin Vault, data as needed by the Internet service **to log onto and authenticate as the user at the one or more web sites** storing information of interest to the user, to gather information on behalf of the user." *Id.*, Abstract (emphasis added).

98. The user in Hazlehurst is an account holder, because the user "subscribes to services offered by service provider" (*id.*, 4:7-9), and later provides "user credentials" to log in the "Web-based account or service". *Id.*, 1:59-61.

99. Hazlehurst's accessing method is done in a secure manner, because the "authentication credentials" stored in the Pin Vault are "**encrypted**" and the Pin Vault application has an "**encryption/decryption software (SW)** to decrypt and use the user authentication credentials" from the Pin Vault to access the Web-based account. *Id*., 4:48-67 (emphasis added).

100. Hazlehurst also discloses that this system "can perform **two-factor authentication** on behalf of a user." *Id.*, 3:33-34 (emphasis added). Examples of multi-factor authentication used to secure online accounts include "**time sensitive tokens**" and "**CAPTCHA**". *Id.*, 5:27-47 (emphasis added). In an embodiment where "**one or more sites use time sensitive tokens as part of the two-factor authentication**, **PS 105** may obtain the correct token file from device **108** at the same time the standard authentication credentials are obtained at the time of login to the site." *Id.*, 5:30-34 (emphasis added). In another embodiment where "**user's sites require submission of CAPTCHA data**, those CAPTCHA displays, including the appropriate data entry boxes, may be captured from those sites and then submitted to device **108** at which time the user operating the device may type in **the appropriate characters required by the site as part of the two-factor authentication**." *Id.*, 5:38-43 (emphasis added).

b) *[1a] collecting the account holder's credentials;*

101. Hazlehurst discloses this limitation. Hazlehurst discloses that the Pin Vault Application (PVA) must "obtain" the account holder/user's credentials from the user. *Id.*, 8:42-44 ("the user has the **required credentials** stored locally on his or her device and **the system must obtain them from the user**") (emphasis added); *see also id*., 4:48-52 ("the client-side PVA application **112** contains an encrypted user pin vault (UPV) file **113** that is adapted to store at minimum all of the **user's** subscribed-to Web site locations and appropriate **authentication credentials** for accessing those sites") (emphasis added); *id.*, 1:42-45 ("[A]utomatic logins are accomplished for a subscriber at Internet destinations by use of **pre-stored** configuration information and **user authentication credentials**.") (emphasis added). A portal server **105** can also collect the account holder/user's credentials submitted by the account holder/user. *Id.*, 5:15-18 ("In applications ... without PVA **114** or **112**, the user submits the required Web site access and authentication data to PS **105** for storage.").

102. The collected MFA factors (i.e., account holder's credentials) may include: "(1) Something a person knows, such as a password, a PIN or an out-of-wallet-response. (2) Something a person has, such as a mobile phone, a credit card or a hardware security token. (3) Something a person is, such as represented by a fingerprint, by a retinal scan, or by some other biometric." *Id*., 2:30-38 and Fig. 2.

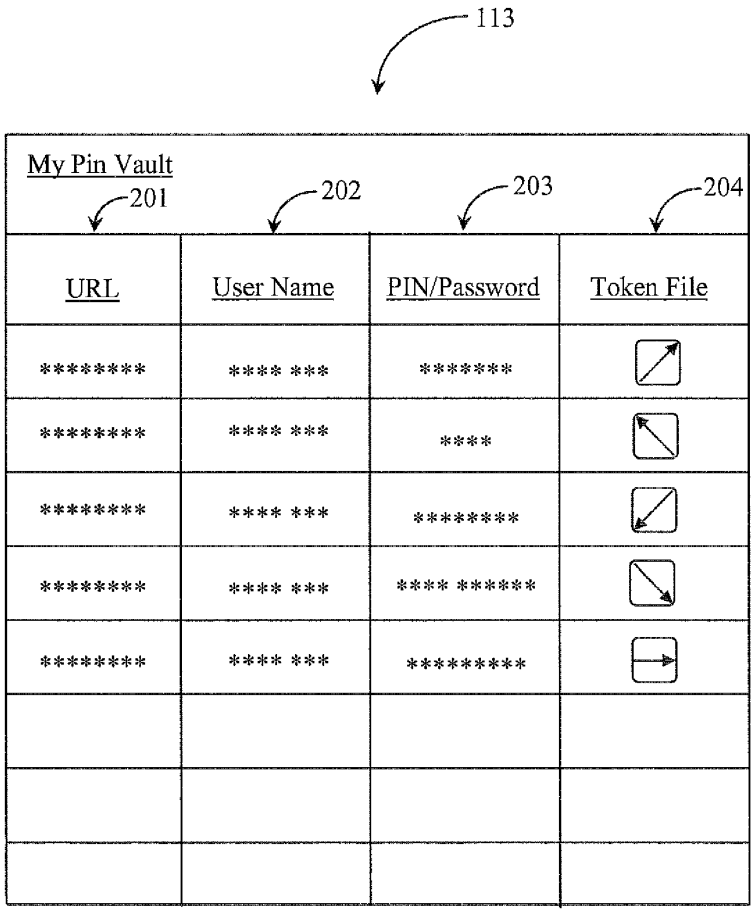


Fig.2

Hazlehurst (Ex-1005), Fig. 2

c) *[1b] encrypting the account holder's credentials using one or more processors;*

103. Hazlehurst discloses this limitation. As discussed above, Hazlehurst discloses that "the client-side PVA application **112** contains **an encrypted user pin vault (UPV) file 113** that is adapted to store at minimum all of the user's subscribed-to Web site locations and appropriate **authentication credentials** for accessing those sites." *Id.*, 4:48-52 (emphasis added); *see also id.*, 4:58-59

("Encrypted file **113** is made accessible to PVA **114** running on PS **105**."). "PVA **114** has the appropriate encryption/decryption software (SW) to decrypt and use the user authentication credentials from UPV **113**." *Id.*, 4:65-67. "The entire process is automated at server side and no human operator has access to the decrypted data of the user." *Id.*, 4:67-5:2.

104. The account holder/user's credentials are encrypted by PVA **112** on device **108**. *Id*., 5:20–22 ("Provision of PVA **112** on device **108** enables the user to keep the sensitive information stored locally in encrypted format."); *see also id*., 4:19–23 ("device **108** is a personal handheld device ... Device **108** may also be a PDA, a laptop computer, or a desktop computer"). A POSITA would understand that device **108** "enables" the storage of information in encrypted format means the encryption is an "other operation[]" that is executed by one or more processors on device **108**. Ex-1010, 13 ("Calculations and other operations in your program are performed by a central processing unit (CPU) inside the computer."); *id.*, 18 ("Most modern computer systems provide more than one CPU (or 'core') within the computer system."). In addition, a POSITA would understand that encryption software "running on PS **105**" (Ex-1005, 4:58-59) means it is executed by one or more processors on portal server **105**.

d) *[1c] storing the account holder's encrypted credentials using one or more processors, wherein the encrypted credentials are stored in a data store;*

105. Hazlehurst discloses this limitation. Specifically, Hazlehurst discloses "a data gathering system having an Internet-capable appliance associate with a user, a Pin Vault software instance executing from a digital memory media coupled to the appliance, and **a Pin Vault file stored in a data repository** coupled to the appliance, **the Pin Vault file comprising data for authenticating the user to one or more web sites** storing information of interest to the user", meaning that the account holder/user's credentials are stored in a data store. Ex-1005, 2:53-59 (emphasis added); *see also id*., 5:15-18 ("In applications ... without PVA **114** or **112**, the user submits the required Web site access and authentication data to PS **105** for storage."). The Pin Vault file that contains the account holder/user's credentials is "**encrypted**". *Id*., 4:48-50 ("the client-side PVA application **112** contains an **encrypted** user pin vault (UPV) file **113**") (emphasis added).

106. It would have been obvious to a POSITA that this storage process would be completed using one or more processors on "device **108**", e.g., cell phone, PDA, PC, and laptop computers. *Id*., 4:52-56 ("UPV file **113** may be a small file downloadable to a cell phone, PDA, or other device …. Likewise, the UPV file may be stored locally on any PC or laptop computer."); 5:20-22 ("Provision of PVA **112** on device **108** enables the user to keep the sensitive

information stored locally in encrypted format."). Storage is an "other operation[]" that is performed by one or more processors on the device. Ex-1010, 13 ("Calculations and other operations in your program are performed by a central processing unit (CPU) inside the computer."); *id.*, 18 ("Most modern computer systems provide more than one CPU (or 'core') within the computer system.").

e) *[1d] verifying the account holder's credentials by accessing, via one or more processors, the online account using the account holder's credentials;*

107. Hazlehurst discloses this limitation. In one embodiment, Hazlehurst describes a system where "**PS 105** may navigate to various user-subscribed Web sites and ... may **log in representing the user** and access specific data from those sites and even perform certain transactions at those sites, authorized by the user." Ex-1005, 4:37-42 (emphasis added); *see also id.*, 5:27-30 ("PS **105** aided by PVA **114** may access device **108** as needed and interrogate UPV **113** to obtain required site location information and authentication data needed to access the sites."). This login process involves "**authenticating as the user using authentication data**" (i.e., verifying the account holder's credentials … using account holder's credentials). *Id*., 3:25-26; 9:16–20 (In an embodiment where CAPTCHA is required, "one [software] transaction is used to submit the CAPTCHA data and the username password/PIN credential to the gatherer", and then "the gatherer agent uses the credentials and the CAPTCHA data to authenticate the user."). As

discussed above, the "Web-based account" being accessed in Hazlehurst is an online account that requires account holder/user credentials. *Id*., Abstract, 1:59-61

108. A POSITA would understand that this verification/access process is completed by the portal server **105** using one or more processors. *Id*., 5:27-30 ("PS **105** aided by PVA **114** may access device **108** as needed and interrogate UPV **113** to obtain required site location information and authentication data needed to access the sites."). Verification/Access is an "other operation[]" performed by one or more processors on the server. Ex-1010, 51 ("In a client-server configuration, a program on a client computer accepts services and resources from a complementary program on a server computer."); *id*., 13 ("Calculations and other operations in your program are performed by a central processing unit (CPU) inside the computer."); *id*., 18 ("Most modern computer systems provide more than one CPU (or 'core') within the computer system.").

f) *[1e] receiving a multi-factor authentication request from the online account, the multi-factor authentication request indicating that a response is required from the account holder before further access to the online account is granted;*

109. Hazlehurst discloses this limitation. Hazlehurst discloses that the system receives a MFA request in the form of a "token" or "CAPTCHA" from the "Web-based account". Ex-1005, 1:59-62 ("a **Web-based account** or service may **require the user to log in and provide, in addition to normal user credentials,**

a token that changes in value, say every 60 seconds") (emphasis added); *id.*, 5:38-43 ("In the case that any of the **user's sites require submission of CAPTCHA data**, those CAPTCHA displays, including the appropriate data entry boxes, may be captured from those sites and then submitted to device **108** at which time the user operating the device may type in the appropriate characters required by the site as part of the two-factor authentication.") (emphasis added).

110. Hazlehurst provides in Figure 5 "a process flow chart illustrating a two-factor authentication sequence by proxy leveraging CAPTCHA input." Ex-1005, 3:44-46.

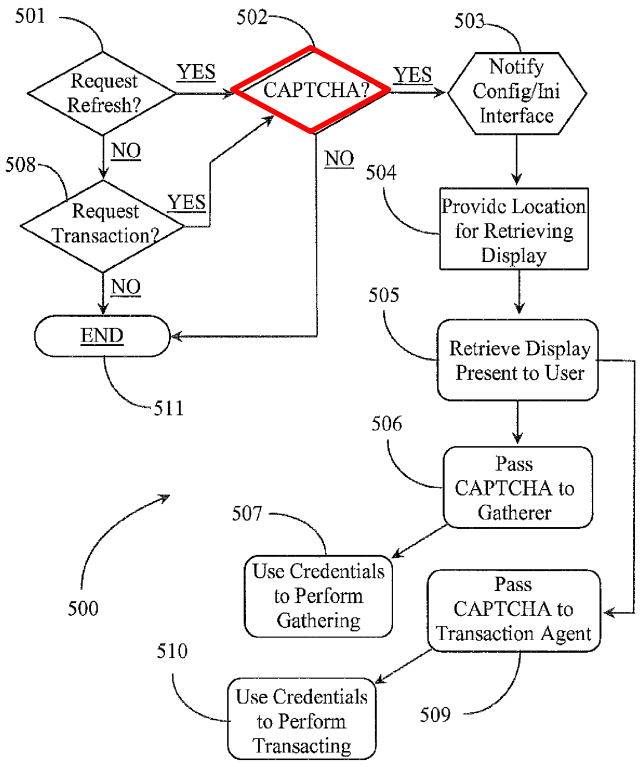


Fig. 5

Hazlehurst (Ex-1005), Fig. 5 (Annotated)

111. "The process may begin with a refresh request in step **501** or with a transaction request in step **508**." *Id.*, 8:55-57. "If in step **501**, the user requests a refresh then **in step 502, the system determines if there is a CAPTCHA requirement as part of a two-factor authentication**." *Id.*, 8:57-59 (emphasis added). "If in step **502**, the system determines that a CAPTCHA requirement exists to login, then at step **503**, the system notifies a configuration and initiation interface that a CAPTCHA interaction is required, and in step **504** provides the interface with the address location for receiving the display. In the meantime, the gathering SW remains in a hold state waiting for information." *Id.*, 8:63-9:2.

112. The account holder/user's response to the MFA request is required before further access to the online account is granted, because the system "need[] to obtain the required data in order to access the appropriate Web sites and services" (*id.*, 4:61-65), and if "the credentials fails then the system may automatically reattempt the sequence". *Id*., 8:28–31.

g) *[1f] prompting the account holder for a response and receiving the response thereto from the account holder; and*

113. Hazlehurst discloses this limitation. As discussed above for limitation [1e], Hazlehurst discloses that if the system receives a CAPTCHA request from a Web site, then at step **503** it "notifies a configuration and initiation interface that a

CAPTCHA interaction is required, and in step **504** provides the interface with the address location for receiving the display." *Id*., 8:63-9:1.

114. Hazlehurst further discloses that "[i]n step **505**, the configuration and initiation interface retrieves the CAPTCHA display required by the site and **presents the display to the device of the user along with the appropriate data entry box or field for the user to type in the required characters**", i.e., prompting the account holder for a response. *Id.*, 9:2-6 (emphasis added). "In the case that any of the user's sites require submission of CAPTCHA data, those CAPTCHA displays, including the appropriate data entry boxes, may be **captured from those sites and then submitted to device 108** at which time the user operating the device may type in the appropriate characters required by the site as part of the two-factor authentication", i.e., receiving the response thereto from the account holder. *Id*., 5:38-43 (emphasis added). After the user types in the required CAPTCHA data and "submit[s] [it] at the time of login attempt", the system "may retrieve the user login credentials and accept the user-submitted CAPTCHA data in the same transaction." *Id.*, 5:44-47.

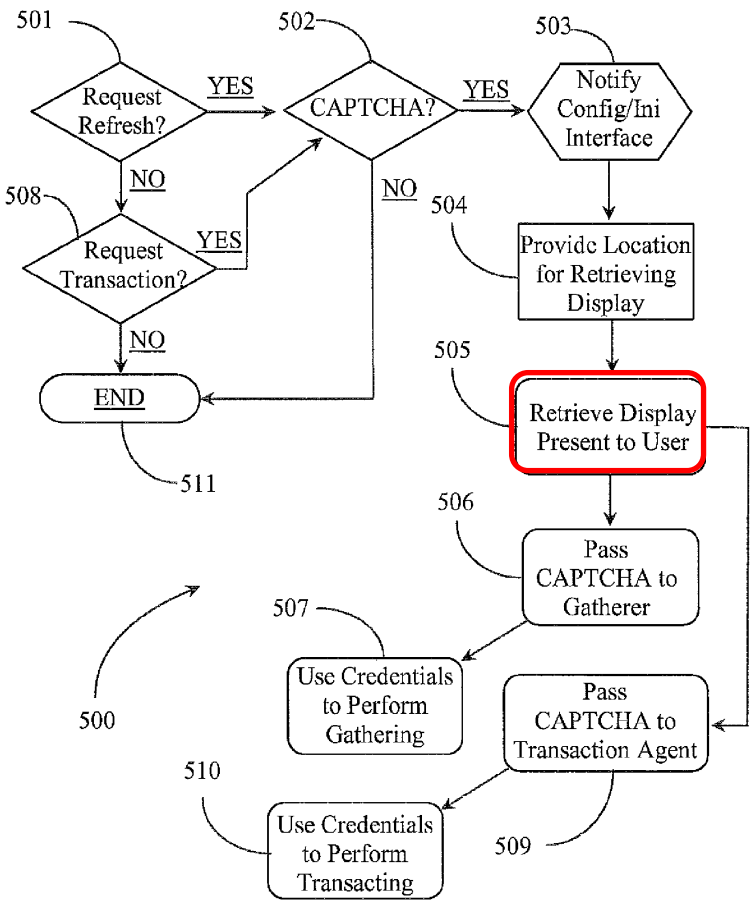


Fig. 5

Hazlehurst (Ex-1005), Fig. 5 (Annotated)

h) *[1g] transmitting the account holder's response to the online account to satisfy the multi-factor authentication request.*

115. Hazlehurst discloses this limitation. As discussed above for limitation [1f], Hazlehurst discloses a system that can present the CAPTCHA displays from a "Web-based account", along with data entry boxes, to the user's device and receives a response from the user. *Id.*, 9:2-6; *see also id.*, 1:59-62, 2:12-14 ("[A] Web-based account or service may require the user to log in and provide, in

addition to normal user credentials, a token that changes in value, say every 60 seconds" and "may be configured to initiate the CAPTCHA whenever a request for services is received."). After the CAPTCHA data is received, "in step **506** the configuration and initiation interface passes the user-submitted CAPTCHA data to the gatherer agent." *Id.*, 9:7-10. "In one embodiment, the system may interrogate the [user pin vault] via the server-side PVA instance and the user credentials are passed to the gatherer from the PVA application while the CAPTCHA data arrives at the gatherer via the configuration and initiation interface. In another embodiment, one SW transaction is used to submit the CAPTCHA data and the username password/PIN credentials to the gatherer. In any case, **the gatherer agent uses the credentials and the CAPTCHA data to authenticate the user in step 507**." *Id.*, 9:12-20 (emphasis added); *see also id*., 5:3–7 ("PS 105 in addition to PVA 114, has a gatherer software (GSW) agent application 115 … that may spawn multiple individual instances for performing … site login authentication"). Step **507** means that the third-party entity (i.e., "the gatherer agent") transmits the account holder's credentials (i.e., "credentials" and "CAPTCHA data") to satisfy the multi-factor authentication request.

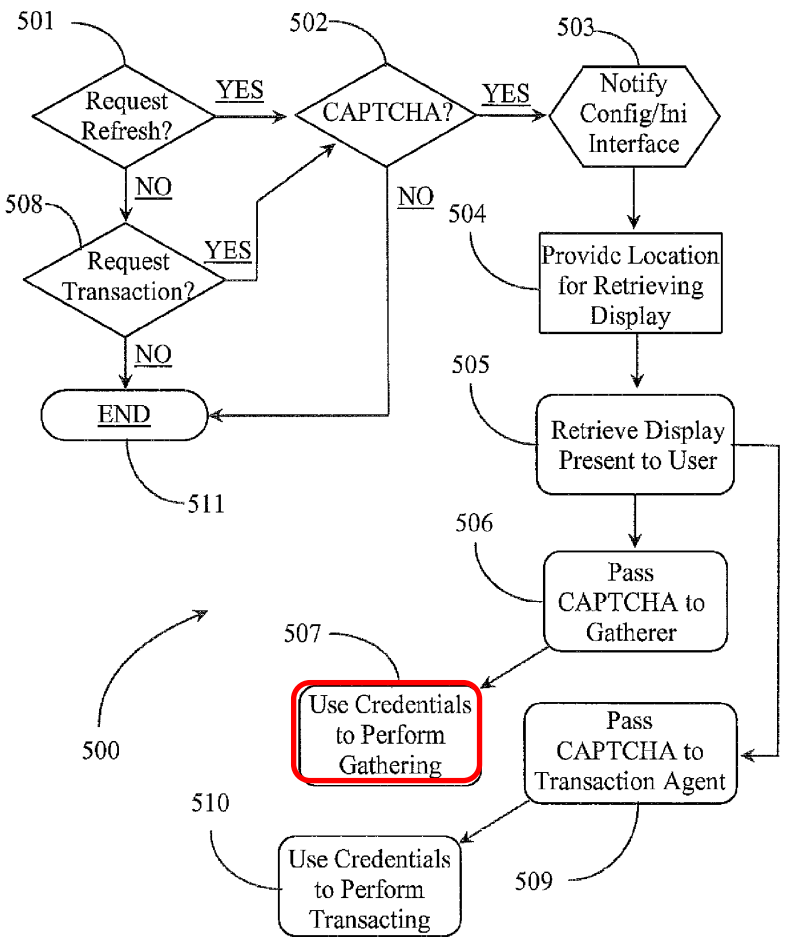


Fig. 5

Hazlehurst (Ex-1005), Fig. 5 (Annotated)

3. Claim 4

a) *[4] The method of claim 1, wherein the account holder's credentials are collected via a user interface.*

116. Hazlehurst discloses this limitation. Specifically, Hazlehurst discloses a "pin vault user interface (UI) **300**" that "may be an integrated part of PVA," the Pin Vault Application. Ex-1005, 6:64-67. Hazelhurst further explains that "PVA **112** contains an encrypted user pin vault (UPV) file **113** that is adapted to store at

minimum all of the user's … authentication credentials" (*Id.*, 4:48-52) and explains that "**Table 113 may** or may not **be displayable** and may vary widely in displayed format if it is displayable." Ex-1005, 6:14-19 (emphasis added). A POSITA would understand that, when displayable, the pin vault table would be used by the UI to collect the user's credentials.

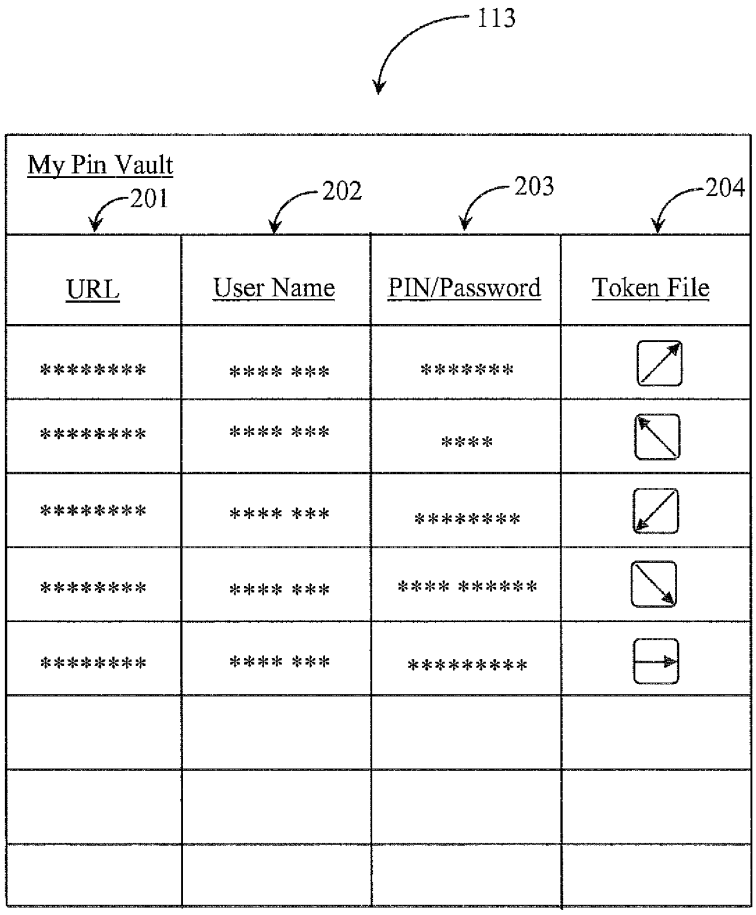


Fig.2

Hazlehurst (Ex-1005), Fig. 2

117. As Fig. 2 shows, "**the table is divided into several columns and rows adapted to organize the client information and credentials**. A column **201** headed URL lists the Web site navigation addresses to client sites. ... A column **202** headed User Name lists the appropriate user names that the client has set up for the appropriate Web site addresses listed in column 201. ... A column **203** titled PIN/Password lists the appropriate PIN numbers and/or passwords that the client has associated with the appropriate usernames in column **202** for the appropriate addresses listed in column **201**." *Id.*, 6:20-32. Therefore, Fig. 2 shows how the Pin Vault Application (PVA) User Interface (UI) collects the account holder/user's credentials, including "User Name[s]", "PIN/Password[s]", and "Token File[s]". *Id*.

4. <u>Claim 5</u>

a) *[5] The method of claim 1, wherein the account holder's credentials are collected from a data store.*

118. Hazlehurst discloses this limitation. As discussed above, Hazlehurst discloses that a "Pin Vault file" that contains the account holder/user's credentials is stored in a "data repository", i.e., a data store. Ex-1005, 2:56-59 ("**a Pin Vault file stored in a data repository** coupled to the appliance, the Pin Vault file comprising data for authenticating the user to one or more web sites storing information of interest to the user."). Further, Hazlehurst discloses that PS 105 collects the account holder/user's credentials from the data repository by

"**access[ing] UPV 113**" (i.e., the Pin Vault file) on the user's device when needed to "obtain the required data in order to access the appropriate Web sites and services." *Id*., 4:61-65.

5. <u>Claim 6</u>

a) *[6pre] A method for permitting a third-party entity to gain access, in a secure manner, to an online account held by an account holder that is secured by a multi-factor authentication protocol, comprising the following steps performed by the third-party entity:*

119. As discussed above for limitation [1pre], to the extent this preamble term is limiting, Hazlehurst discloses it.

b) *[6a] collecting the account holder's credentials;*

120. As discussed above for limitation [1a], Hazlehurst discloses this limitation.

c) *[6b] encrypting the account holder's credentials using one or more processors;*

121. As discussed above for limitation [1b], Hazlehurst discloses this limitation.

d) *[6c] temporarily storing the account holder's encrypted credentials using one or more processors, wherein the encrypted credentials are held in memory;*

122. Hazlehurst discloses this limitation. For instance, Hazlehurst discloses that PS 105 (*e.g.*, third-party entity) "temporar[ily] stor[es]" the account holder/user's UPV data. Ex-1005, 5:63-65 ("[A] user may also elect to transmit

some or all of their UPV data to PS 105 for permanent or ***temporary storage***."). As discussed above for limitation [1c], the UPV data is temporarily stored using one or more processors on the portal server 105.

123. Hazlehurst further discloses an embodiment where "PS 105 does not retain any user authentication information locally." *Id.*, 4:60-61. Instead, when PS 105 is performing services for the user, "it accesses UPV 113 from the user's device when needed to obtain the required data in order to access the appropriate Web sites and services." *Id.*, 4:61-65. PS 105 then temporarily holds the UPV 113 in its memory for the "encryption/decryption software (SW) to decrypt and use the user authentication credentials from UPV 113." *Id.*, 4:65-67.

e) *[6d] verifying the account holder's credentials by accessing, via one or more processors, the online account using the account holder's credentials;*

124. As discussed above for limitation [1d], Hazlehurst discloses this limitation.

f) *[6e] receiving a multi-factor authentication request from the online account, the multi-factor authentication request indicating that a response is required from the account holder before further access to the online account is granted;*

125. As discussed above for limitation [1e], Hazlehurst discloses this limitation.

g) *[6f] prompting the account holder for a response and receiving the response thereto from the account holder; and*

126. As discussed above for limitation [1f], Hazlehurst discloses this limitation.

h) *[6g] transmitting the account holder's response to the online account to satisfy the multi-factor authentication request.*

127. As discussed above for limitation [1g], Hazlehurst discloses this limitation.

6. Claim 9

a) *[9] The method of claim 6, wherein the account holder's credentials are collected via a user interface.*

128. As discussed above for claim 4, Hazlehurst discloses this limitation.

7. Claim 10

a) *[10] The method of claim 6, wherein the account holder's credentials are collected from a data store.*

129. As discussed above for claim 5, Hazlehurst discloses this limitation.

D. Ground 3: Claims 1, 4-6, and 9-10 are obvious over Emerson in view of Hsu

1. Overview of Emerson (Ex-1006)

130. Emerson was filed on February 1, 2008 and published on August 6, 2009, which I understand makes it prior art under AIA 35 U.S.C. 102(a)(1).

131. Emerson is directed to "a method for conducting a secure transaction over a network", specifically, a method that "facilitates electronic fund transfers,

initiated at a merchant site, from a consumer's financial institution to the merchant." Ex-1006, [0001], [0010]. The method can be implemented through a system illustrated in FIG. 1, which includes a merchant server **104**, a financial institution server **106**, an online debit system (ODS) server **108** running an **ODS agent 114**, and the consumer **110**, all connected over the Internet **112**. *Id.*, [0035].

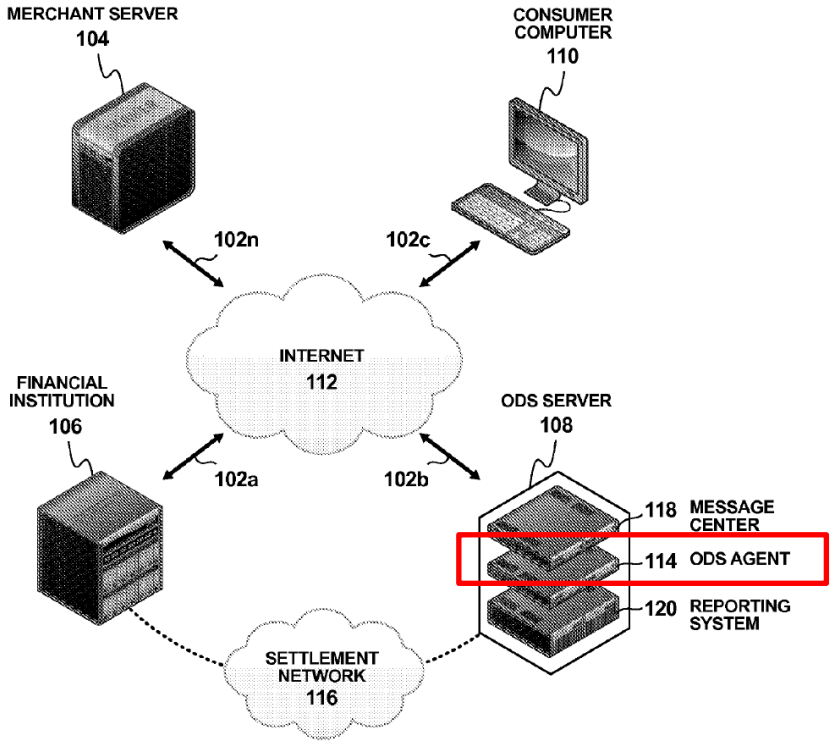
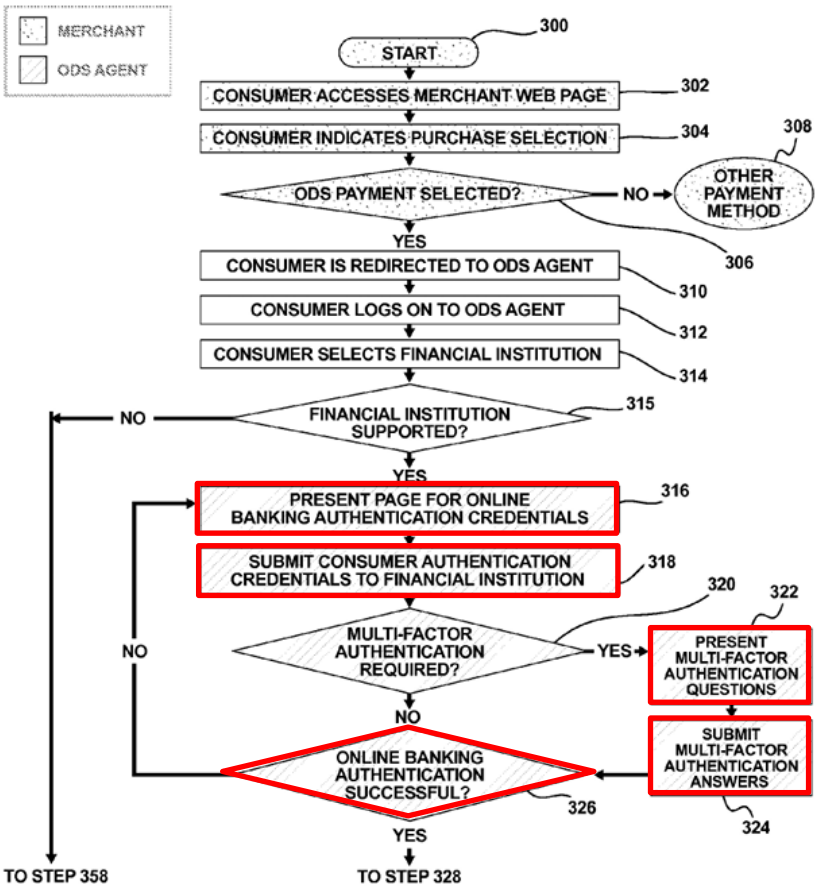


FIG. 1

Emerson (Ex-1006), FIG. 1 (Annotated)

132. Emerson further explains that "[m]ost bank nowadays require multi-factor authentication" which involves "usual authentication credentials such as log-in and password" and "(random) Q and A (question and answer) tests from a set of questions that are preset by a consumer for on-line banking." *Id.*, [0059]. As an example, Emerson discloses the following steps in FIG 3a:



300

FIG. 3a

Emerson (Ex-1006), FIG. 3a (Annotated)

133. In step **302**, "a consumer uses a consumer computer **100** to access a merchant's web page through a network, such as the internet **112**." *Id*., [0049]. In step **304**, "the consumer uses the consumer computer **110** to indicate to the merchant server **104** a purchase selection." *Id.*, [0050]. In step **306**, "the merchant server **104** provides the consumer with a choice of payment options supported by the merchant .... [O]ne of these options is an on-line debit process using the ODS server **108** and ODS agent **114**." *Id.*, [0051]. In step **310**, "[i]f the on-line debit process is selected by the consumer as the preferred payment method, ... the consumer is redirected to the ODS server **108** to complete the transaction." *Id.* In step **312**, "the consumer logs onto the ODS agent **114**." *Id.*, [0065]. In step **314**, "the consumer selects one of the financial institutions (if more than one is offered) from the list of available financial institutions **502**". *Id.*, [0068]. In step **315**, "[t]he ODS agent **114** ... determines whether the selected financial institution is supported by the ODS agent **114**". *Id*.

134. In step **316**, "**the ODS agent 114** determines the information required by the selected financial institution for **gaining access to the consumer's account at that institution** and then presents a financial-institution log-in information page, such as page 600, to the consumer. The page 600 requests the necessary authentication credentials from the consumer." *Id.*, [0069] (emphasis added). In step **318**, "the information ... is submitted to the specified financial institution". *Id*.

Plaid Ex1003
Page 77 of 372

(emphasis added). In step **320**, "**the ODS agent 114** determines whether or not the financial institution **106** requires **'multi-factor' authentication.**" *Id.*, [0070]. In step **322**, "**the ODS agent 114** presents the **multi-factor authentication** to the consumer." *Id.*, [0071] (emphasis added). In step **324**, "[t]he consumer presents the answer and ... **the ODS agent 114** submits the consumer's answers to the financial institution 106." *Id*. (emphasis added). In step **326**, "**the ODS agent 114** interprets the response from the financial institution and determines whether the authentication credentials are valid and accepted by the financial institution." *Id*., [0072] (emphasis added). If the authentication credentials are accepted, in step **328**, "**the ODS agent 114 acts on behalf of the consumer and maintains a session with financial institution 106** in order to interact and respond to actions and messages from the financial institution 106." *Id*., [0073].

2. Claim 1

a) *[1pre] A method for permitting a **third-party entity** to **gain access, in a secure manner, to an online account held by an account holder** that is **secured by a multi-factor authentication protocol**, comprising the following steps performed by the third-party entity,*

135. To the extent this preamble term is limiting, Emerson in view of Hsu teaches it.

136. Specifically, Emerson discloses a a third-party entity termed "**ODS agent 114**" which "acts independently based on a pre-determined set of steps and

interactions necessary to complete a bill pay or other payment on behalf of the consumer" and is "bank independent," *i.e.*, "no 'system-level' or dependent integration with any bank or any specific financial system." Ex-1006, [0075], [0090]. The ODS agent provides "**on-line banking authentication and fund transfer services through a specific financial institution**." *Id.*, [0053]. For on-line banking authentication, "**the ODS agent 114** determines the information required by the selected financial institution for **gaining access to the consumer's account at the institution**." *Id.*, [0069].

137. Further, Emerson discloses that "**[m]ulti-factor authentication** is a relatively-new procedure for ensuring the person accessing the account has permission to do so", but "[m]ost banks nowadays require **multi-factor authentication** when an unusual usage pattern is detected (for example when a user logs in from a new device for the first time)." Ex-1006, [0059], [0070] (emphasis added). In an embodiment, Emerson discloses that the "**ODS agent 114** ... implements on-line banking authentication" through "cach[ing] (device) tokens set by on-line banking to pass **multi-factor authentication** when applicable." *Id.*, [0060] (emphasis added). In another embodiment, "the **ODS agent 114** determines whether or not the financial institution **106** requires **'multi-factor' authentication**." *Id.*, [0070] (emphasis added). If the answer is yes, "the **ODS agent 114** presents the **multi-factor authentication** to the consumer", and

"submits the consumer's answer to the financial institution **106**." *Id.*, [0071] (emphasis added).

138. Hsu discloses that the access is gained in a secure manner, as the "**stored authentication information** and/or stored retrieved information **are encrypted**" and "may be **encrypted** when it is communicated over the network **112**." Ex-1004, [0039] (emphasis added).

(i) Motivation to Combine Emerson and Hsu

139. A POSITA would have been motivated to combine Emerson and the encryption technique disclosed in Hsu, and would have had a reasonable expectation of success in doing so.

140. Emerson and Hsu are extraordinarily similar. Fundamentally, they identify the same problem—the use of multi-factor authentication in authenticating online account holders makes it difficult for third-party automated systems to securely and unattended access the accounts, as MFA typically requires direct user input. Ex-1004, [0004]; Ex-1006, [0075]. And both disclose similar methods to address this problem—using a third-party system that collects the account holder's credentials and responses to multi-factor authentication, enabling automated access to the accounts. Ex-1004, [0023]; Ex-1006, FIG. 3a. Therefore, a POSITA looking to implement the account access method in Emerson would have looked at other similar methods, such as Hsu, to enhance the method.

141. Specifically, Emerson and Hsu both recognize that the information being processed by the system (such as authentication credentials and financial information) is "sensitive" in nature, Ex-1004, [0039]; Ex-1006, [0032], and that "security" of transactions is a major issue in online banking. Ex-1004, [0003], [0005]; Ex-1006, [0032]. Emerson also emphasizes repeatedly that the method is for conducting a "secure" transaction over a network. Ex-1006, Abstract, [0010], [0014], [0015], [0031], [0047], [0053], claims 1, 8, 14, 18. Thus, a POSITA seeking to implement the method disclosed in Emerson would have been motivated to look for available methods, like in Hsu, to find a way to further enhance its security. Hsu teaches a way to enhance security of the online banking process through encryption. Ex-1004, [0039]. Hsu further discloses two complimentary ways to use encryption: (1) encrypting the information and storing it locally on a computer or remotely on a server, *id.*, [0026], [0029], [0031], [0039], [0043] and (2) encrypting the information when it is communicated over the network. *Id.*, [0039]. A POSITA would have understood that encrypting sensitive information would have been beneficial because it can avoid exposing the information to malicious actors on the Internet. Therefore, a POSITA would have been motivated to add the encryption step, as taught by Hsu, to the verification process in Emerson to address the well-known security threat of using unencrypted information.

142. Furthermore, a POSITA would have had a reasonable expectation of success in incorporating the encryption step of Hsu into the method disclosed in Emerson. Emerson's system already has a "processor" operable to receive input from the user and communicate with financial institutions. Ex-1006, [0014]. This processor can be easily programmed by a POSITA to encrypt the account holder's credentials or other information when communicated with financial institutions, as taught by Hsu. Ex-1004, [0039].

b) *[1a] collecting the account holder's credentials;*

143. Emerson in view of Hsu teaches this limitation. As shown in FIG. 6, Emerson discloses that "**the ODS agent 114** determines the information required by the selected financial institution for gaining access to the consumer's account at that institution and then **presents a financial-institution log-in information page, such as page 600, to the consumer**." Ex-1006, [0069] (emphasis added).

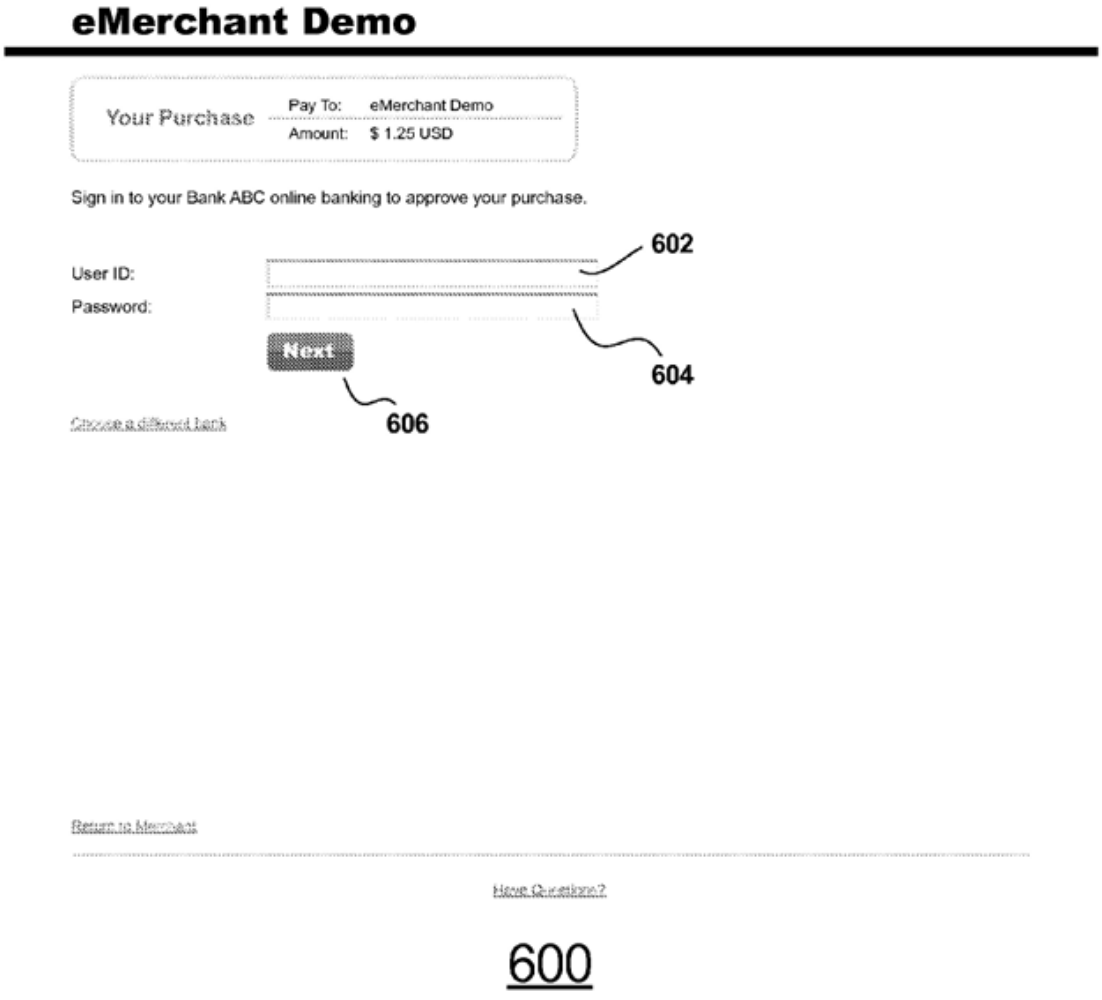


FIG. 6

Emerson (Ex-1006), FIG. 6

144. Emerson further teaches that "**[t]he page 600 requests the necessary authentication credentials from the consumer**. In one embodiment, the page requests, in a first field **602**, the consumer's identifying information and, in a second field **604**, a password." *Id.*, [0069] (emphasis added). "After [the consumer] hit[s] a 'next,' or other similar button **606**, **the information contained in the**

fields 602 and 604 is submitted to the specified financial institution in step **318**." *Id*. (emphasis added).

c) *[1b] encrypting the account holder's credentials using one or more processors;*

145. Emerson in view of Hsu teaches this limitation. Hsu teaches the encryption of the account holder's credentials. Ex-1004, [0039] ("The multi-factor **authentication information** and/or the retrieved information may be a sensitive nature. As a consequence, in some embodiments stored information and/or stored retrieved information **are encrypted**. In addition, such information may be **encrypted** when it is communicated over the network **112**.") (emphasis added). The information is encrypted by the "**encryption module**" using "**one or more processing units 210**". *Id*., [0043]–[0044] ("memory **224** includes ... optional **encryption module** (or a set of instructions) **250**. ... Instructions in the various modules in the memory 224 may be implemented in a high-level procedural language, an object-oriented programming language, and/or in an assembly or machine language. The programming language may be complied or interpreted, i.e., configurable or configured to be **executed by the one or more processing units 210**.") (emphasis added).

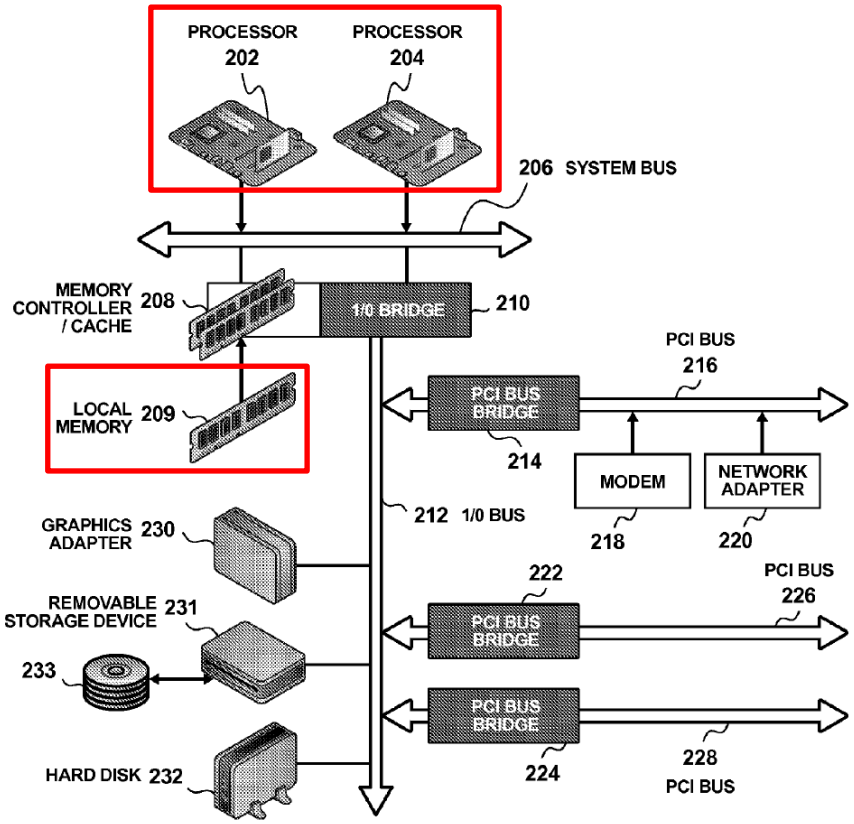
146. As discussed in detail above, it would have been obvious to a POSITA to apply the encryption step disclosed in Hsu to encrypt the collected user ID **602** and password **604** in Emerson. Ex-1006, [0069]. A POSITA would have

been motivated to do so because both Emerson and Hsu recognized that the information collected is "sensitive" (Ex-1004, [0039]; Ex-1006, [0031] ("[T]he consumer only provides sensitive information to a single entity that is consistent throughout all transactions, regardless of the various merchant selected.")), and Emerson states repeatedly that it provides a method for conducting "secure" transactions. Ex-1006, [0010], [0014], [0015], [0031], [0047], claims 1, 8, 14, and 18.

d) *[1c] storing the account holder's encrypted credentials using one or more processors, wherein the encrypted credentials are stored in a data store;*

147. Emerson in view of Hsu teaches this limitation. Emerson discloses that the ODS agent **114** may "**cache (device) tokens** set by on-line banking to pass multi-factor authentication" and "**stor[e] it in a database for subsequent usage**." *Id*., [0060] (emphasis added). Furthermore, Emerson discloses that the account holder/consumer's personal information is stored at the "ODS server **108**" using one or more "processors **202** or **204**", in order to compare it with the information submitted by merchant. Ex-1006, [0066] ("[T]he consumer's personal information can be provided to the ODS server **108** by the merchant server **104**. In one embodiment, as a security measure, the ODS agent **114** asks for the same information and compares it to that submitted by the merchant server **104**. If a difference in the two sets of information is detected, the transaction may be

declined."). Emerson explains that the ODS server **108** "may be a symmetric multiprocessor (SMP) system including a plurality of **processors 202 and 204**" and "[t]he **processor 202 or 204** in conjunction with memory controller **208** controls what data is stored in memory **209**." *Id.*, [0040] (emphasis added).



200

FIG. 2

Emerson (Ex-1006), FIG. 2 (Annotated)

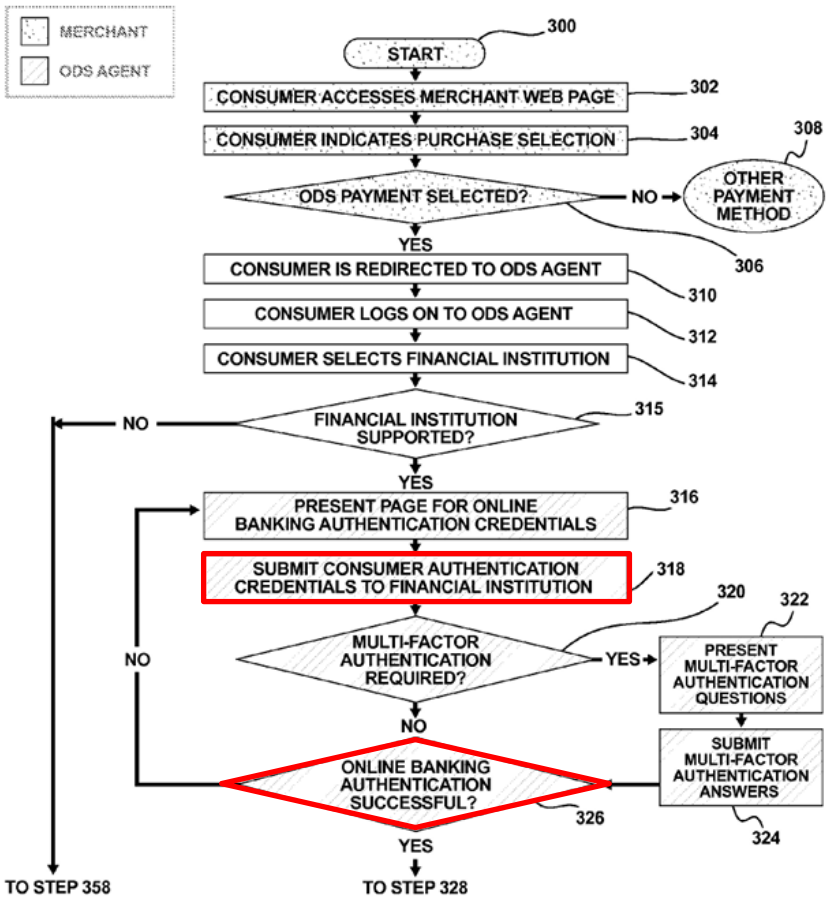
148. As discussed above in detail, Hsu also discloses storing the account holder's encrypted credentials in a data store. Ex-1004, [0031] ("The information

to be retrieved may be initially stored locally on the computer **110** or remotely, for example, on the server **114**, in a data structure **116**"). It would have been obvious to a POSITA to store the collected information (e.g., user ID **602** and password **604** in Emerson) at the ODS server **108**, because Emerson seeks to overcome the lack of "built in identity verification" by using the ODS agent **114** to "automatically perform[] substantially all steps for electronic communication with a financial institution to gain access to an account." Ex-1006, [0011]. A POSITA would have had a reasonable expectation of success in incorporating Hsu's stored credentials technique in the Emerson system, because Emerson discloses a similar technique where a type of authentication credentials (*e.g.*, token) is stored in a database for subsequent usage. *Id.*, [0060].

e) *[1d] verifying the account holder's credentials by accessing, via one or more processors, the online account using the account holder's credentials;*

149. Emerson in view of Hsu teaches this limitation. As discussed above for limitation [1a], Emerson discloses that the ODS agent **114** collects the account holder's credentials, such as user ID **602** and password **604**, through page **600**. Emerson further discloses that after the credentials are collected, "**the information contained in the fields 602 and 604 is submitted to the specified financial institution in step 318**", i.e., verifying the account holder's credentials. *Id.*, [0069] (emphasis added). If the financial institution accepts the authentication credentials

(step **326**), "**the ODS agent 114 acts on behalf of the consumer and maintains a session with the financial institution 106** in order to interact and respond to actions and messages from the financial institution **106**", i.e., accessing the online account using the account holder's credentials *Id*., [0072]-[0073] (emphasis added).



300

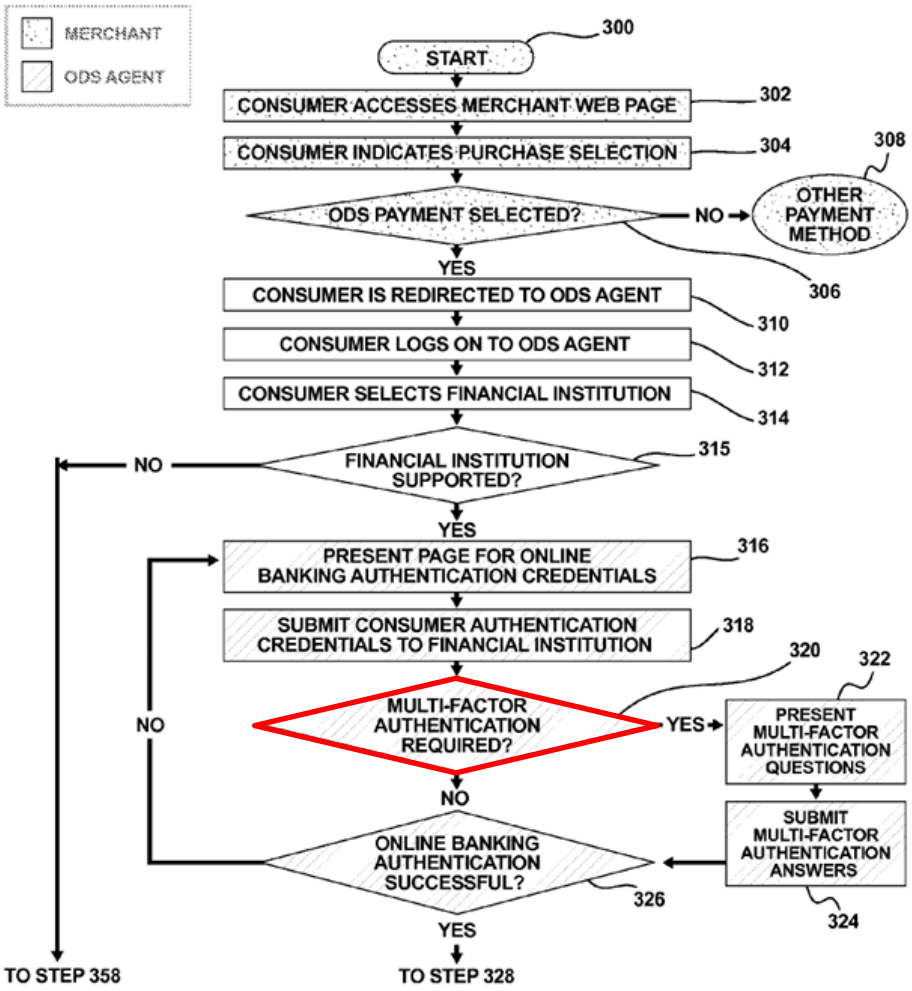
FIG. 3a

Emerson (Ex-1006), FIG. 3a (Annotated)

150. A POSITA would understand that this verification/access process is completed using one or more processors disclosed in Emerson. *Id.*, [0044] ("[T]he computer programs, when executed, enable the processor **202** and/or **204** to perform the features of the ODS agent **114**.").

f) *[1e] receiving a multi-factor authentication request from the online account, the multi-factor authentication request indicating that a response is required from the account holder before further access to the online account is granted;*

151. Emerson in view of Hsu teaches this limitation. Emerson discloses that "**[i]n step 320, the ODS agent 114 determines whether or not the financial institution 106 requires 'multi-factor' authentication**." *Id.*, [0070]. That passage explains that a response to the MFA is required from the account holder before access to the online account is granted. *Id*.; *see also id*., [0072] ("If the customer is unable to enter the correct authentication credentials the transaction will not proceed."). As an example, "a user selects a particular graphic at some point when setting up the account." *Id.*, [0070]. "**During the log-in, the consumer is given a choice of graphics and, only upon making the correct selection of graphics, is the granted access to the account**." *Id*. Another example involves prompting challenge questions that are preset by the consumer. *Id.*, [0059], [0070].



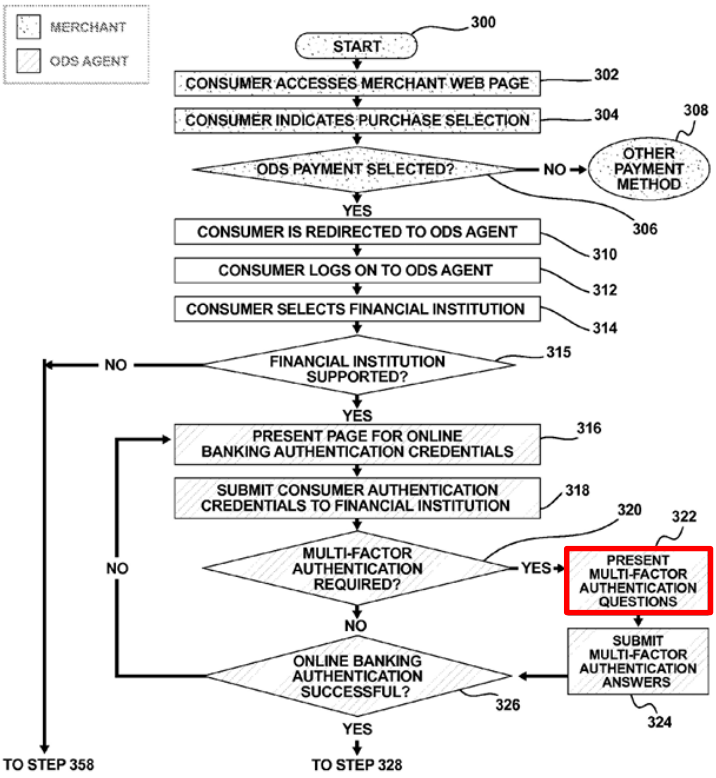
300

FIG. 3a

Emerson (Ex-1006), FIG. 3a (Annotated)

g) *[1f] prompting the account holder for a response and receiving the response thereto from the account holder; and*

152. Emerson in view of Hsu teaches this limitation. Emerson discloses that if multi-factor authentication is required in step **320**, "the process moves to **step 322** where **the ODS agent 114 presents the multi-factor authentication to the consumer**" and then **"[t]he consumer presents the answer**" to the ODS agent. *Id.*, [0071].



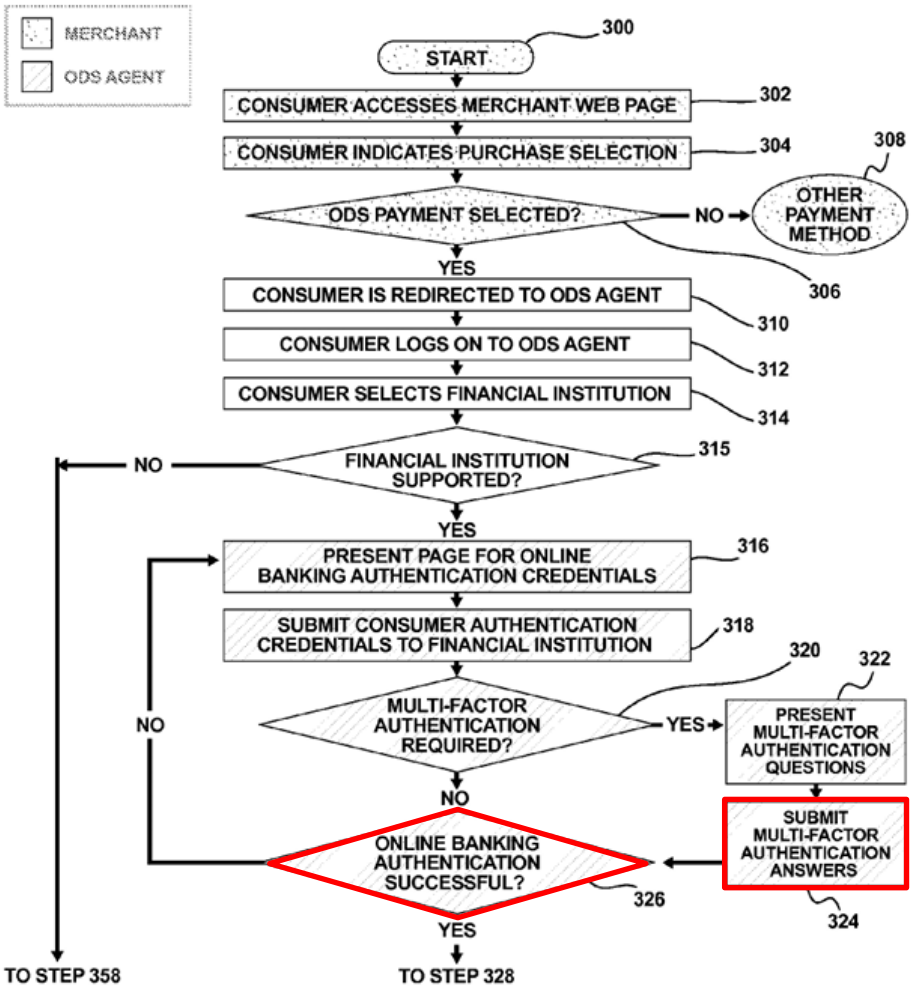
300

FIG. 3a

Emerson (Ex-1006), FIG. 3a (Annotated)

h) *[1g] transmitting the account holder's response to the online account to satisfy the multi-factor authentication request.*

153. Emerson in view of Hsu teaches this limitation. Emerson discloses that after receiving the answer to the multi-factor authentication request from the account holder/consumer, "**in step 324, the ODS agent 114 submits the consumer's answer to the financial institution 106**." *Id.*, [0071]. "**In step 326, the ODS agent 114** interprets the response from the financial institution and **determines whether the authentication credentials are valid and accepted by the financial institution**." *Id*. "If the credentials are not accepted by the financial institution the customer returns to Step **316** to correct the authentication credentials .... If the authentication credentials are accepted by the financial institution the process proceeds to Step **328**." *Id.*, [0072]. "In step **328**, the ODS agent **114** acts on behalf of the consumer and maintains a session with the financial institution **106** in order to interact and respond to actions and messages from the financial institution **106**." *Id.*, [0073].



300

FIG. 3a

Emerson (Ex-1006), FIG. 3a (Annotated)

3. Claim 4

a) *[4] The method of claim 1, wherein the account holder's credentials are collected via a user interface.*

154. As discussed above for limitation [1a], Emerson in view of Hsu teaches this limitation. Specifically, FIG. 6 in Emerson presents a user interface termed "**page 600**" that is used to collect the account holder's credentials, such as user ID **602** and password **604**. *Id.*, FIG. 6; *see also id.*, [0069] ("ODS agent **114** ... presents a financial-institution log-in information page, such as page **600**, to the consumer.").

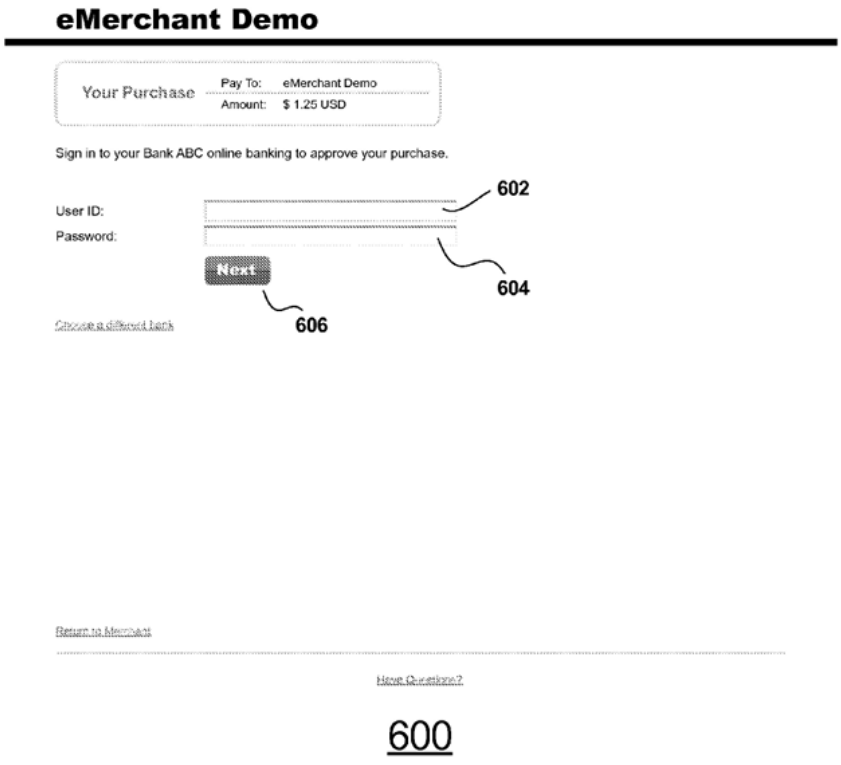


FIG. 6

Emerson (Ex-1006), FIG. 6

4. Claim 5

a) *[5] The method of claim 1, wherein the account holder's credentials are collected from a data store.*

155. Emerson in view of Hsu teaches this limitation. As discussed above for limitation [1c], Emerson discloses using the account holder's credentials stored in a data store for authentication. Ex-1006, [0060] ("ODS agent **114** ... implements on-line banking authentication. It may **cache (device) tokens set by on-line banking to pass multi-factor authentication** when applicable. The caching mechanism may be implemented by keeping track of the token for a specific on-line banking log-in and **storing it in a database for subsequent usage**.") (emphasis added).

156. Hsu also discloses retrieving/collecting the account holder's credentials in a data store, as discussed above for claim [5] in ground 1. Ex-1004, [0035] ("The financial program may either request the authentication information (city of birth) from the user or may **retrieve the answer** (Palo Alto) **from storage**.") (emphasis added). It would have been obvious to a POSITA to store the collected information (e.g., user ID **602** and password **604** in Emerson) at the ODS server **108** and later collect this information from the ODS server **108** for automatic login, because Emerson seeks to overcome the problem of not having "built in identity verification" by using the ODS agent **114** to "automatically perform[] substantially all steps for electronic communication with a financial

institution to gain access to an account." Ex-1006, [0011]. A POSITA would have had a reasonable expectation of success in incorporating Hsu's stored credentials technique in the Emerson system, because Emerson discloses a similar technique where a type of authentication credentials (e.g., token) is stored in a database for subsequent usage. *Id.*, [0060]

5. Claim 6

a) *[6pre] A method for permitting a third-party entity to gain access, in a secure manner, to an online account held by an account holder that is secured by a multi-factor authentication protocol, comprising the following steps performed by the third-party entity:*

157. As discussed above for limitation [1pre], to the extent this preamble term is limiting, Emerson in view of Hsu teaches it.

b) *[6a] collecting the account holder's credentials;*

158. As discussed above for limitation [1a], Emerson in view of Hsu teaches this limitation.

c) *[6b] encrypting the account holder's credentials using one or more processors;*

159. As discussed above for limitation [1b], Emerson in view of Hsu teaches this limitation.

d) *[6c] temporarily storing the account holder's encrypted credentials using one or more processors, wherein the encrypted credentials are held in memory;*

160. Emerson in view of Hsu teaches this limitation. Emerson discloses that the ODS agent **114** may "**cache (device) tokens** set by on-line banking to pass multi-factor authentication". Ex-1006, [0060] (emphasis added). A POSITA would understand that cached data is stored only temporarily. Ex-1010, 263 ("All of these examples of caching share the common attribute that they increase performance by providing faster access to data, anticipating its potential need in advance, then **storing that data temporarily** where it is rapidly available.") (emphasis added). Moreover, a POSITA would also appreciate that temporary data would be stored in either "local memory **209**" or the "memory controller/cache **208**" in the ODS server **108**. *Id*., [0040].

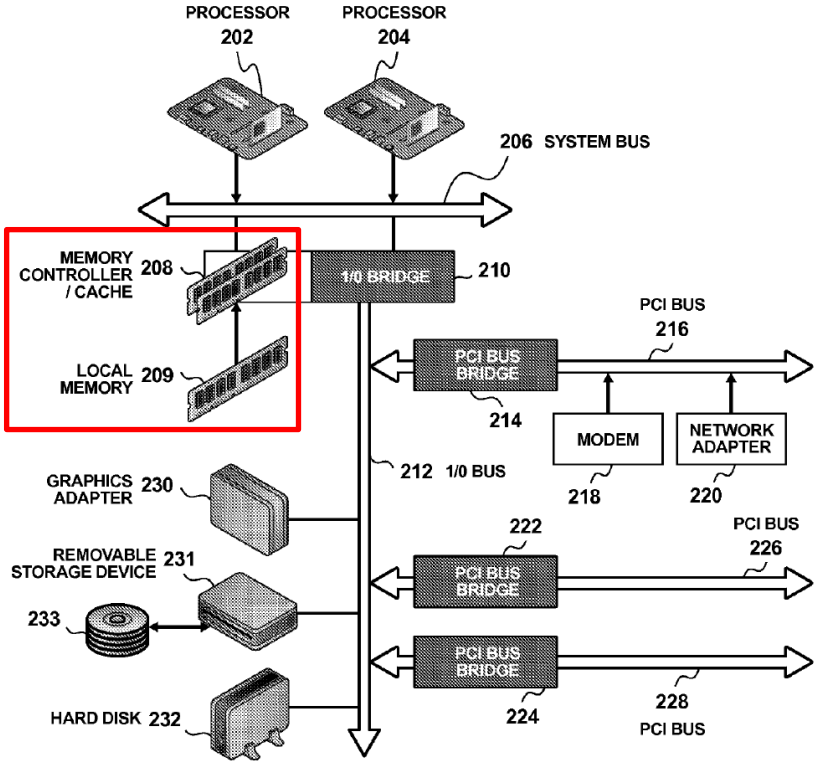


FIG. 2

Emerson (Ex-1006), FIG. 2 (Annotated)

161. In addition, a POSITA would understand that in FIG. 6, when the user ID **602** and password **604** are submitted via page **600**, they are stored temporarily in memory at the ODS server **108** in order to communicate them to the financial institution over the Internet **112**. *Id.*, [0069] ("After hitting a 'next,' or other similar button **606**, the information contained in the fields **602** and **604** is submitted to the specified financial institution in step **318**.").

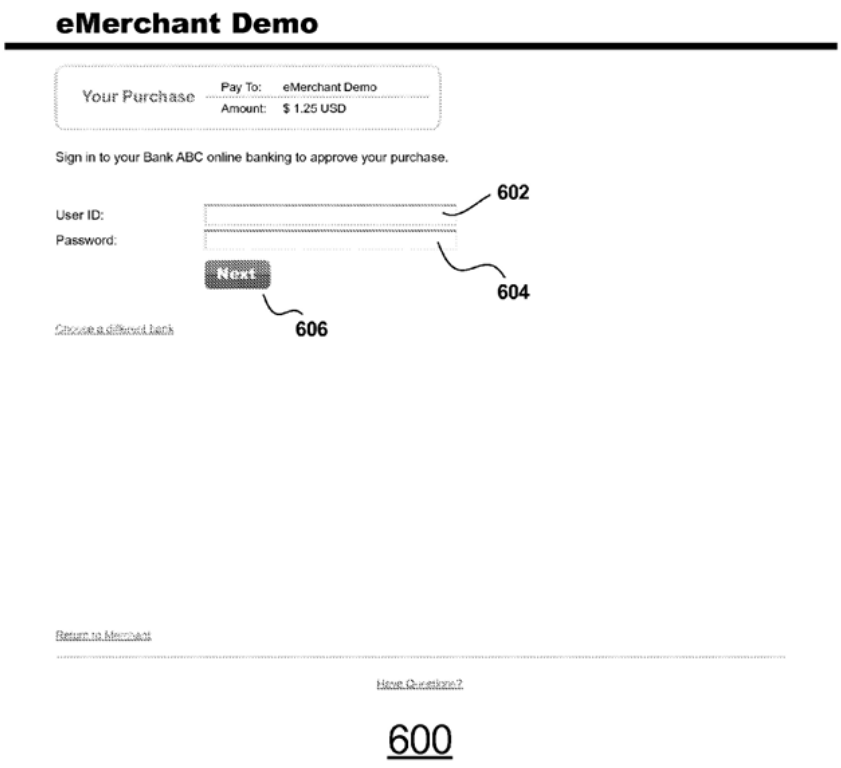


FIG. 6

Emerson (Ex-1006), FIG. 6

162. In an embodiment, Emerson further discloses that "**the ODS agent 114** by default or by a consumer's instruction, **does not store the consumer's log in information**." *Id.*, [0065]. Emerson also discloses that the ODS server **108** contains local memory **209**. *Id.*, [0040] ("Referring to FIG. 2, a block diagram of a data processing system that may be implemented as a server, such as server **104**, **106**, or **108** .... Also, connected to system bus **206** is memory controller/cache **208**, which provides an interface to local memory **209**."). A POSITA would understand

that the local memory **209** can be used to temporarily store the account holder's credentials.

163. Hsu also discloses temporarily storing the account holder's credentials in memory, as discussed above for limitation [6c] in ground 1. It would have been obvious to a POSITA to use Hsu's temporary storage technique in the Emerson system, because Emerson discloses that the ODS agent **114** "does not store the consumer's log in information" so that "the consumer does not have to worry about his/her private information being obtained by a third party" (Ex-1006, [0065]), and Emerson would be able to achieve that by temporarily storing the information for authentication and deleting it afterwards. A POSITA would have a reasonable expectation of success in doing so, because Emerson discloses a similar technique where a type of authentication credentials (e.g., token) is cached (i.e., temporarily stored) for authentication. *Id.*, [0060].

e) *[6d] verifying the account holder's credentials by accessing, via one or more processors, the online account using the account holder's credentials;*

164. As discussed above for limitation [1d], Emerson in view of Hsu teaches this limitation.

f) *[6e] receiving a multi-factor authentication request from the online account, the multi-factor authentication request indicating that a response is required from the account holder before further access to the online account is granted;*

165. As discussed above for limitation [1e], Emerson in view of Hsu teaches this limitation.

g) *[6f] prompting the account holder for a response and receiving the response thereto from the account holder; and*

166. As discussed above for limitation [1f], Emerson in view of Hsu teaches this limitation.

h) *[6g] transmitting the account holder's response to the online account to satisfy the multi-factor authentication request.*

167. As discussed above for limitation [1g], Emerson in view of Hsu teaches this limitation.

6. <u>Claim 9</u>

a) *[9] The method of claim 6, wherein the account holder's credentials are collected via a user interface.*

168. As discussed above for claim 4, Emerson in view of Hsu teaches this limitation.

7. Claim 10

a) *[10] The method of claim 6, wherein the account holder's credentials are collected from a data store.*

169. As discussed above for claim 5, Emerson in view of Hsu teaches this limitation.

E. Grounds 4, 5, and 6: Claims 2, 3, 7, and 8 are obvious over Hsu, Hazlehurst, or Emerson/Hsu in view of Peotta

1. Overview of Peotta (Ex-1007)

170. Peotta was published in the February 2011 issue of the International Journal of Computer Science & Information Technology (IJCSIT) (Ex-1008), which I understand makes it prior art under AIA 35 U.S.C. 102(a)(1). Moreover, The IJCSIT website, where Peotta is located, was indexed by popular search engines by at least September 17, 2011 (Ex-1011, ¶31), and was one of the front-page results provided by Google Scholar when searching for "Internet banking authentication authorization" prior to the effective filing date of the '090 patent. Exhibit G.

171. Literature search engines like Google Scholar represent one of the most frequent ways that a POSITA would have identified existing techniques and methods during the time period of the alleged invention. Hence, a POSITA looking to employ a method to automate authentication and authorization for Internet banking applications would have found it obvious to type in related keywords into

the Google Scholar search bar to locate existing methods. As shown in Exhibit G, a POSITA would have easily uncovered Peotta using such an approach.

172. Peotta relates to the "classification of attacks and vulnerabilities" affecting internet banking systems. Ex-1007, 1. In particular, Peotta describes a "Short Message Service (SMS)" security method, which "has been applied in some banking systems to notify users about transactions requiring their authorization." *Id.*, 4. Peotta explains that the SMS method "provides a second authentication channel for transactions that fit certain characteristics by sending to the user a set of characters which have to [be submitted][2] in order to authorize and process the transaction through the online banking system." *Id.*

173. Peotta indicates that a vulnerability of the "Short Message Service" security method used in internet banking systems is that "[t]he attacker may alter the cellular phone number to which the authorization methods are sent." *Id.*, 5. When the cellular phone number has been altered, a third party (here an "attacker") will be able to provide the authentication information needed to access a user's account without the user's participation because the authentication requests would then be sent to a cellular phone number controlled by the third party so that the

[2] The original sentence reads "which have to *informed* in order to authorize". According to a similar use of the term "informed" in Peotta (page 192), this means "submitted": "When the legitimate user accesses the service, the authentication data *informed* by him is captured by the malware …". Ex-1007, 7.

third party may then provide the authentication response needed. *Id*. Thus, Peotta teaches a POSITA a method of authentication on behalf of user without the user's participation by rerouting authentication requests to a different phone number. Whereas Peotta describes this method being used by an attacker, that same concept may be used by the user's agent, such as an authorized third-party system, to enable authentication without the user's required participation.

2. <u>Claim 2</u>

a) *[2] The method of claim 1, further comprising the steps of: storing one or more of the account holder's endpoints for the online account in a data store associated with the online account; and utilizing said one or more endpoints to permit the third-party to gain access to the online account, or information pertaining to the online account, without requiring a response by the account holder to any subsequent multi-factor authentication requests.*

174. The Hsu/Peotta (**Ground 4**), Hazlehurst/Peotta (**Ground 5**), and Emerson/Hsu/Peotta (**Ground 6**) combinations teach this limitation. Peotta explains that in the SMS authorization method, "banking systems ... notify users" (i.e., "account holder[s]") "about transactions requiring their authorization" by "sending to the user a set of characters" to a "cellular phone number" (i.e., an "endpoint for the online account"). Ex-1007, 4-5. It would have been obvious to a POSITA that Peotta's statement that an "attacker may alter the cellular phone number to which the authorization messages are sent" would require a "banking system" (i.e., an "online account") to store the altered phone number in a data store

associated with the banking system in order for the system to send "a set of characters" to the altered phone number.

175. It would have been obvious to a POSITA to incorporate Peotta's SMS authorization technique with Hsu's financial program, Hazlehurst's Pin Vault software, or Emerson/Hsu's ODS agent (i.e., "the third party") to provide the benefit of fully automated retrieval of information from financial accounts protected by SMS authentication (i.e., "without requiring a response by the account holder to any subsequent multi-factor authentication requests"). Hsu's MFA examples are directed to challenge questions, e.g., "In what city were you born?" or "In what year was your friend born?" Ex-1004, [0035]-[0038]. Hsu discloses that "the financial program ... may retrieve the answer from storage", enabling information retrieval for the user in a "fully automated fashion" without any need to prompt the user for a response. *Id*. Hazlehurst's examples include time-sensitive tokens. Ex-1005, 1:56-2:6. Hazlehurst discloses that the software "may obtain the correct token file" that is "automatically generated and made available within UPV **113**", and thus "the user need not provide any direct input". *Id*., 5:30-37. Emerson also focuses on tokens, and discloses that the ODS agent "may cache (device) tokens set by on-line banking to pass multi-factor authentication", enabling "a user free (i.e., user's further input is not needed) communication session with the financial institution **106**." Ex-1006, [0053], [0060].

176. Modified to incorporate Peotta's SMS authorization technique, Hsu's financial program, Hazlehurst's Pin Vault software, or Emerson/Hsu's ODS agent would take the place of the "attacker" described in Peotta. The financial program/Pin Vault software/ODS agent would "alter the cellular phone number to which the authorization messages are sent" to a number accessible by the financial program/Pin Vault software/ODS agent. Upon an attempt to retrieve information for the user from the bank, the "set of characters" to be submitted to the bank "in order to authorize and process the transaction through the online banking system" (i.e., "gain access to the online account") would be sent to the altered phone number accessible by the financial program/Pin Vault software/ODS agent (i.e., "utilizing said one or more endpoints to permit the third-party to gain access to the online account"). Ex-1007, 4.

177. Similar to Hsu's description of "retriev[ing] the answer" to a challenge question "from storage" to avoid prompting the user, the financial program would submit the received set of characters to the bank to provide a "fully automated" retrieval of information, since the financial program directly receives the set of characters at the altered cellular phone number. Ex-1004, [0035]-[0038]. Similarly, after obtaining the set of characters, Hazlehurst's Pin Vault Software would submit it to the bank directly so that "the user need not provide any direct

input". Ex-1005, 5:35. Emerson/Hsu's ODS agent would do the same to achieve a "user free" communication session with the bank. Ex-1006, [0053].

178. Therefore, instead of "prompting the account holder for a response and receiving a response thereto" as claim 1 recites, the Hsu/Peotta, Hazlehurst/Peotta, and Emerson/Hsu/Peotta combinations each teach using an endpoint (e.g., phone number) "to permit the third-party to gain access to the online account, or information pertaining to the online account, without requiring a response by the account holder to any subsequent multi-factor authentication requests" as claim 2 recites. Accordingly, the Hsu/Peotta, Hazlehurst/Peotta, and Emerson/Hsu/Peotta combinations obviate claim 2.

(i) Motivation to Combine Hsu, Hazlehurst, or Emerson/Hsu with Peotta

179. It would have been obvious to a POSITA to utilize Peotta's teaching of "alter[ing] the cellular phone number to which the authorization messages are sent" in Hsu or Hazlehurst or Emerson/Hsu's systems to provide the benefit of fully automated authentication for systems that are protected by the SMS security method.

180. Hsu, Hazlehurst, Emerson/Hsu, and the supposed "attacker" in Peotta share the same goal of accessing a user's online account without requiring a response by the user to any subsequent MFA requests. Ex-1004, [0038]; Ex-1005, 1:41-45; Ex-1006, [0011]. Although Peotta describes "alter[ing] the cellular phone

number to which the authorization messages are sent" as a "vulnerability" that may be exploited by an "attacker", it would have been obvious to a POSITA that the technique is equally applicable to benevolent automated authentication that has been consented to by the user, such as in the case of Hsu, Hazlehurst, and Emerson/Hsu. Ex-1007, 5.

181. Indeed, Hsu, Hazlehurst, and Emerson/Hsu all recognize that MFA presents challenges for automated financial services. Hsu recognizes that "existing financial software is not configured to perform multi-factor authentication in different environments", and that it is therefore "difficult for such financial software to assemble and share financial information". Ex-1004, [0005]. Hazlehurst recognizes that existing software "cannot complete automated proxy logins if CAPTCHA [or time-sensitive tokens] is required to gain access or to perform a transaction", and these security measures thus "present a challenge" for automated services. Ex-1005, 2:21-29. Emerson also notes that "[m]ulti-factor authentication is a relatively-new procedure for ensuring the person accessing the account has permission to do so", and that "[m]ost banks nowadays require multi-factor authentication when an unusual usage pattern is detected (for example when a user logs in from a new device for the first time)." Ex-1006, [0059], [0070].

182. In response, Hsu, Hazlehurst, and Emerson/Hsu present solutions allowing automatic access to user's account to reduce the burden associated with

the security requirements. Specifically, Hsu's "approach to multi-factor authentication allows the financial program to assemble (i.e., retrieve) information for the user in a semi-automated or fully automated fashion", which "may reduce the burden associated with the security requirements for different documents, host, and/or systems." Ex-1004, [0038]. Hazlehurst discloses that "automatic logins are accomplished for a subscriber at Internet destinations by use of pre-stored configuration information and user authentication credentials." Ex-1005, 1:41-45. Emerson's ODS agent "automatically performs substantially all steps for electronic communication with a financial institution to gain access to an account." Ex-1006, [0011]. Adding Peotta's technique to Hsu, Hazlehurst, or Emerson/Hsu would have provided the known benefit of fully automated authentication for sites that utilize SMS authentication without requiring the user's participation in the authentication process.

183. A POSITA would have had a reasonable expectation of success in incorporating Peotta's SMS authentication technique in the Hsu, Hazlehurst, and Emerson/Hsu systems, in the same manner taught as Peotta, because SMS security is an additional type of "multi-factor authentication" that is like those described in Hsu, Hazlehurst, and Emerson/Hsu. For example, Hsu notes that "multi-factor authentication provided by the user may include … **telephone numbers**, … and/or additional personal information." Ex-1004, [0029] (emphasis added). Hazlehurst

notes that "two-factor authentication generally comprises … Something a person has, such as a **mobile phone**, a credit card or a hardware security token." Ex-1005, 2:30-36 (emphasis added).

184. The incorporation of "alter[ing] the cellular phone number to which the authorization messages are sent" to automatically perform the SMS security authentication would have been nothing more than the use of a known technique (Peotta's SMS authentication technique) to improve similar devices (Hsu's financial program, Hazlehurst's Pin Vault software, and Emerson/Hus's ODS agent) in the same way to yield predictable results (fully automated authentication for account's protected by SMS security). Although Peotta describes changing the phone number for SMS authentication as an attack when done by an unauthorized third party, it provides greater flexibility for the user when conducted by an authorized third party because the user then does not need to provide a response to enable the authorized access to its account information. Thus, applying this method to automate account access, as described in the Challenged Claims, would have been a predictable use of existing technology.

3. <u>Claim 3</u>

a) *[3] The method of claim 2, wherein the endpoint includes an email address or a phone number.*

185. The Hsu/Peotta (**Ground 4**), Hazlehurst/Peotta (**Ground 5**), and Emerson/Hsu/Peotta (**Ground 6**) combinations teach this limitation. Peotta teaches

that "the endpoint includes ... a phone number" as recited in claim 2: "The attacker may alter the **cellular phone number** to which the authorization messages are sent." Ex-1007, 5 (emphasis added).

4. Claim 7

a) *[7] The method of claim 6, further comprising the steps of: storing one or more of the account holder's endpoints for the online account in a data store associated with the online account; and utilizing said one or more endpoints to permit the third-party to gain access to the online account, or information pertaining to the online account, without requiring a response by the account holder to any subsequent multi-factor authentication requests.*

186. As discussed above for claim 2, the Hsu/Peotta (**Ground 4**), Hazlehurst/Peotta (**Ground 5**), and Emerson/Hsu/Peotta (**Ground 6**) combinations teach this limitation.

5. Claim 8

a) *[8] The method of claim 7 wherein the endpoint includes an email address or a phone number.*

187. As discussed above for claim 3, the Hsu/Peotta (**Ground 4**), Hazlehurst/Peotta (**Ground 5**), and Emerson/Hsu/Peotta (**Ground 6**) combinations teach this limitation.

F. Grounds 7, 8, and 9: Claims 2, 3, 7, and 8 are obvious over Hsu, Hazlehurst, or Emerson/Hsu in view of 2FA Q&A

1. Overview of 2FA Q&A (Ex-1009)

188. 2FA Q&A is a question and answer posted on IT Security Stack Exchange on May 29, 2013, which I understand makes it prior art under AIA 35 U.S.C. 102(a)(1). *See* Ex-1012. Like Peotta, Stack Exchange, where 2FA Q&A was published, was indexed by popular search engines like Google prior to the effective filing date. Exhibit H. In fact, Stack Exchange was such a popular resource during that time period that it was among the sites indexed most frequently by Google. *Id*.

189. Around the time of the effective filing date, a POSITA would have turned to Web search engines like Google to locate related techniques and methods by typing in relevant keywords at the search bar. 2FA Q&A was one of the front-page results returned by Google before December 31, 2013 in response to a search for "multi-factor authentication phone number internet banking." Exhibit I.

190. Moreover, a POSITA would have found IT Security Stack Exchange to be a particularly reputable source, as it is one of the family of question-and-answer (Q&A) forums hosted by Stack Exchange, one of the most popular such venues on the Web at the time of the alleged invention. Exhibits J, K. Unlike general-purpose forums like Quora, Stack Exchange hosts domain-specific Q&A sites that allow practitioners looking for advice on specific topics to find

knowledgeable responders. Indeed, I and my students relied heavily on sites hosted by Stack Exchange in that time period, especially Stack Overflow, the original Stack Exchange Q&A site regarding computer programming.

191. 2FA Q&A relates to the question of whether a bank's "two-factor authentication" is "vulnerable to attack, without hacking the bank's software or server, or without access to [the user's] telephone/SMS communications". Ex-1009, 1. The questioner explains that "[w]hen [the user] attempts to log in to [their] bank, an SMS code is sent to [the user's] phone", and the user "then type[s] this nine-character code into the bank's Web site, to login to [the user's] account." *Id.* The questioner indicates that "the only way [they] can imagine" exploitation of the two-factor authentication "would be for someone to install an app on my phone which intercepts SMS traffic and resends the code to an attacker." *Id.*

192. In response, the top answer explains that, in addition to intercepting the code, an attacker could "[s]ocial engineer [the user's] bank to change [the user's] mobile phone details to a phone [the attacker] control[s]", for example, by "call[ing] [the user's] bank pretending to be [the user] and get [the bank] to change the mobile number". *Id.* As with Peotta above, 2FA Q&A teaches a method of enabling authentication without requiring the user's participation by altering the phone number and having the person or system at the new phone number provide any required authentication information. *Id*. Whereas 2FA Q&A describes this

approach being taken by an attacker, that same concept may be used by the user's agent, such as an authorized third-party system, to enable authentication without requiring the user's participation.

2. Claim 2

a) *[2] The method of claim 1, further comprising the steps of: storing one or more of the account holder's endpoints for the online account in a data store associated with the online account; and utilizing said one or more endpoints to permit the third-party to gain access to the online account, or information pertaining to the online account, without requiring a response by the account holder to any subsequent multi-factor authentication requests.*

193. The Hsu/2FA Q&A (**Ground 7**), Hazlehurst/2FA Q&A (**Ground 8**), and Emerson/Hsu/2FA Q&A (**Ground 9**) combinations teach this limitation. The 2FA Q&A explains that in a bank's SMS-based two-factor authentication, when a user (i.e., an "account holder") "attempt[s] to log in to [the user's] bank, an SMS code is sent to [the user's] phone" (i.e., an "endpoint for the online account"). Ex-1009, 1. It would have been obvious to a POSITA that the top answer's statement in the 2FA Q&A that an "attacker could also ... [s]ocial engineer your bank to change your mobile phone details to a phone [the attacker] controls" would require the bank (i.e., an "online account") to store the altered phone number in a data store associated with the banking system in order for the system to send "an SMS code" to the altered mobile phone number. *Id*.

194. It would have been obvious to a POSITA to incorporate the 2FA Q&A's SMS-based two-factor authentication technique with Hsu's financial program, Hazlehurst's Pin Vault software, or Emerson/Hsu's ODS agent (i.e., "the third-party") to provide the benefit of fully automated retrieval of information from financial accounts protected by SMS authentication (i.e., "without requiring a response by the account holder to any subsequent multi-factor authentication requests"). Ex-1004, [0038]. As explained above with respect to grounds 4, 5, and 6, Hsu, Hazlehurst, and Emerson/Hsu present various solutions enabling automated authentication without any user input. Ex-1004, [0035]-[0038] ("retrieve the answer from storage"); Ex-1005, 5:30-37 ("obtain the correct token file" from UPV 113); Ex-1006, [0053], [0060] ("cache (device) tokens set by on-line banking to pass multi-factor authentication").

195. Modified to incorporate the 2FA Q&A's SMS-based two-factor authentication technique, Hsu's financial program, Hazlehurst's Pin Vault software, or Emerson/Hsu's ODS agent would take the place of the "attacker" described in the top answer. Ex-1009, 1. The financial program/Pin Vault software/ODS agent would have the bank "change [the account holder's] mobile phone details to a phone [the financial program] control[s]." *Id*. Upon an attempt to retrieve information for the user from the bank, the "nine-character code" that is typed "into the bank's Web site, to login to [the account holder's] account" (i.e.,

"gain access to the online account") would instead be sent to the changed phone number accessible by the financial program/Pin Vault software/ODS agent (i.e., "utilizing said one or more endpoints to permit the third-party to gain access to the online account") instead of user. *Id.*

196. Similar to Hsu's description of "retriev[ing] the answer" to a challenge question "from storage" to avoid prompting the user, the financial program would submit the SMS code to the bank to provide a "fully automated" retrieval of information, since the financial program directly receives the set of characters at the changed mobile phone number. Ex-1004, [0035]-[0038]. Similarly, after obtaining the set of characters, Hazlehurst's Pin Vault Software would submit it to the bank directly so that "the user need not provide any direct input". Ex-1005, 5:35. Emerson/Hsu's ODS agent would do the same to achieve a "user-free" communication session with the bank. Ex-1006, [0053].

197. Therefore, instead of "prompting the account holder for a response and receiving a response thereto" as claim 1 recites, the Hsu/Peotta, Hazlehurst/Peotta, and Emerson/Hsu/Peotta combinations each teach using an endpoint (*e.g.*, phone number) "to permit the third-party to gain access to the online account, or information pertaining to the online account, without requiring a response by the account holder to any subsequent multi-factor authentication

requests" as claim 2 recites. Accordingly, the Hsu/Peotta, Hazlehurst/Peotta, and Emerson/Hsu/Peotta combinations obviate claim 2.

(i) Motivation to Combine Hsu, Hazlehurst, or Emerson/Hsu with 2FA Q&A

198. It would have been obvious to a POSITA to utilize the 2FA Q&A's teaching of having the user's bank "change [the user's] mobile phone details" in Hsu's financial program, Hazlehurst's Pin Vault software, or Emerson/Hsu's ODS agent (i.e., "the third-party") to provide the benefit of fully automated authentication for systems that are protected by the SMS-based two-factor authentication.

199. Hsu, Hazlehurst, Emerson, and the supposed "attacker" described in 2FA Q&A share the same goal of accessing a user's online account "without requiring a response by the account holder to any subsequent multi-factor authentication requests". Ex-1004, [0038]; Ex-1005, 1:41-45; Ex-1006, [0011]. Although the 2FA Q&A describes "[s]ocial engineer[ing] [the user's] bank to change [the user's] mobile phone details to a phone [the attacker] control[s]" as a potential security exploit, it would have nevertheless been obvious to a POSITA that the technique is equally applicable to benevolent automated authentication that has been consented to by the user, such as in the case of Hsu, Hazlehurst, and Emerson/Hsu. Ex-1007, 5. Indeed, as explained above with respect to Grounds 4, 5, and 6, Hsu, Hazlehurst, and Emerson/Hsu all recognize that MFA presents

challenges for automated financial services (Ex-1004, [0005]; Ex-1005, 2:21-29; Ex-1006, [0059], [0070]), and each present solutions allowing automatic access to user's account to reduce the burden associated with the security requirements. Ex-1004, [0038]; Ex-1005, 1:41-45; Ex-1006, [0011]. Adding the technique described in 2FA Q&A to Hsu, Hazlehurst, or Emerson/Hsu would have provided the known benefit of fully automated authentication for sites that utilize SMS authentication.

200. A POSITA would have had a reasonable expectation of success in incorporating the 2FA Q&A's SMS-based two-factor authentication technique in the Hsu, Hazlehurst, and Emerson/Hsu systems, in the same manner taught as the 2FA Q&A, because SMS-based two-factor authentication is an additional type of "multi-factor authentication" that is like those described in Hsu, Hazlehurst, and Emerson/Hsu. For example, Hsu notes that "multi-factor authentication" includes "**telephone numbers**," "addresses," and "additional personal information." Ex-1004, [0029] (emphasis added). Hazlehurst notes that "two-factor authentication generally comprises … Something a person has, such as a **mobile phone**". Ex-1005, 2:30-36 (emphasis added).

201. The incorporation of "chang[ing] [the user's] mobile phone details to a phone" controlled by a third-party in order to automatically perform the SMS two-factor authentication would have been nothing more than the use of a known technique (2FA Q&A's SMS-based two-factor authentication technique) to

improve similar devices (Hsu's financial program, Hazlehurst's Pin Vault software, and Emerson/Hus's ODS agent) in the same way to yield predictable results (fully automated authentication for account's protected by SMS two-factor authentication). Although 2FA Q&A describes changing the phone number for SMS authentication as a security exploit when done by an unauthorized third party, it provides greater flexibility for the user when conducted by an authorized third party because the user does not need to be involved in providing a response to enable the authorized access to its account information. Thus, applying this method to automate account access, as described in the Challenged Claims, would have been a predictable use of existing technology.

3. Claim 3

a) *[3] The method of claim 2, wherein the endpoint includes an email address or a phone number.*

202. The Hsu/2FA Q&A (**Ground 7**), Hazlehurst/2FA Q&A (**Ground 8**), and Emerson/Hsu/2FA Q&A (**Ground 9**) combinations teach this limitation. The 2FA Q&A teaches that "the endpoint includes ... a phone number" as recited in claim 2: "An attacker could also: ... change your **mobile phone details** to a phone they control." Ex-1009, 1 (emphasis added).

4. Claim 7

a) *[7] The method of claim 6, further comprising the steps of: storing one or more of the account holder's endpoints for the online account in a data store associated with the online account; and utilizing said one or more endpoints to permit the third-party to gain access to the online account, or information pertaining to the online account, without requiring a response by the account holder to any subsequent multi-factor authentication requests.*

203. As discussed above for claim 2, the Hsu/2FA Q&A (**Ground 7**), Hazlehurst/2FA Q&A (**Ground 8**), and Emerson/Hsu/2FA Q&A (**Ground 9**) combinations teach this limitation.

5. Claim 8

a) *[8] The method of claim 7 wherein the endpoint includes an email address or a phone number.*

204. As discussed above for claim 3, the Hsu/2FA Q&A (Ground 7), Hazlehurst/2FA Q&A (Ground 8), and Emerson/Hsu/2FA Q&A (Ground 9) combinations teach this limitation.

VII. ENABLEMENT AND SECONDARY CONSIDERATIONS OF NON-OBVIOUSNESS

205. In my opinion, the prior art references considered in this Declaration are all enabled.

206. I have seen no evidence in either the ’090 patent’s prosecution history or elsewhere supporting any secondary consideration arguments, or evidence of

nexus of such alleged evidence to the challenged claims. To the extent patent owner or its expert asserts the existence of any secondary considerations in its responses, I reserve the right to address any such evidence.

VIII. AVAILABILITY FOR CROSS-EXAMINATION

207. In signing this declaration, I recognize that the declaration will be filed as evidence in a contested case before the Patent Trial and Appeal Board of the United States Patent and Trademark Office. I also recognize that I may be subject to cross-examination in the case and that cross-examination will take place within the United States. If cross-examination is required of me, I will appear for cross-examination within the United States during the time allotted for cross-examination.

IX. RIGHT TO SUPPLEMENT

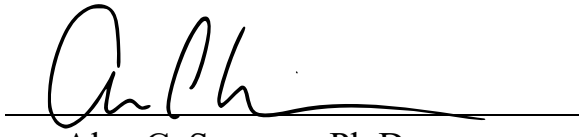
208. I reserve the right to supplement my opinions in the future to respond to any arguments that the Patent Owner raises and to take into account new information as it becomes available to me.

X. JURAT

209. I declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made within the full knowledge that willful

false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code.

Dated: January 14, 2026



Alex C. Snoeren, Ph.D.

EXHIBIT A

Alex C. Snoeren

Computer Science & Engineering
University of California, San Diego
9500 Gilman Drive, EBU3B 3114
La Jolla, CA 92093-0404
(858) 822-2289

snoeren@cs.ucsd.edu
http://www.cs.uscd.edu/~snoeren

Employment summary

University of California, San Diego

2024– Peter Banks Endowed Chair in the Jacobs School of Engineering
2012– Professor in the Department of Computer Science and Engineering
2008–12 Associate Professor in the Department of Computer Science and Engineering
2002–8 Assistant Professor in the Department of Computer Science and Engineering

Google

2023–25 Visiting Faculty Researcher in the Host Networking Group
2022–23 Visiting Faculty Researcher in the Systems Research Group (SRG)

Georgia Institute of Technology

2011–12 Visiting Associate Professor in the School of Computer Science

1999–2002 **BBN Technologies**
Research scientist in the Internetworking Research department

Hewlett-Packard Research Labs

1995–97 Research intern in the Performance, Management & Design group of the Software Technology lab

Education

Massachusetts Institute of Technology

2003 PhD in Electrical Engineering and Computer Science
A Session-Based Architecture for Internet Mobility, Hari Balakrishnan and M. Frans Kaashoek, advisors

Georgia Institute of Technology

1997 MS in Computer Science
BS in Applied Mathematics (*with highest honors*)
1996 BS in Computer Science (*with highest honors*)

Awards

2018, 23 Best Paper Award, ACM SIGCOMM
2020 IEEE Fellow
2018 ACM Fellow
2009 Alfred P. Sloan Fellow
2008 Jay Lepreau Best Paper Award, USENIX OSDI
2001, 7 Best Student Paper Award, ACM SIGCOMM
2004 NSF CAREER Award
George M. Sprowls Thesis Award, MIT EECS Department *(Honorable Mention)*
1996 Outstanding Undergraduate, Georgia Institute of Technology College of Computing

Book chapters; journal and magazine articles

1. "A Developer-Friendly Application-Integrated Far Memory." Anil Yelam, Stewart Grant, Nadav Amit, Radhika Niranjan Mysore, Amy Ousterhout, Marcos K. Aguilera, and Alex C. Snoeren. *USENIX ;login:* March 2025.

2. "Network Topology Facilitates Internet Traffic Control in Autocracies." Eda Keremoglu, Nils B. Weidmann, Alexander Gamero-Garrido, Esteban Carisimo, Alberto Dainotti, and Alex C. Snoeren. *PNAS Nexus*, Volume 3, Number 3, March 2024.

3. "RIPE IPmap Active Geolocation: Mechanism and Performance Evaluation." Ben Du, Massimo Candela, Bradley Huffaker, Alex C. Snoeren, and kc claffy. *ACM SIGCOMM Computer Communications Review* (CCR), Volume 50, Number 1, April 2020.

4. "Understanding the Limits of Passive Realtime Datacenter Fault Detection and Localization." Arjun Roy, Rajdeep Das, Hongyi Zeng, Jasmeet Bagga, and Alex C. Snoeren. *IEEE/ACM Transactions on Networking* (TON), Volume 27, Number 5, October 2019.

5. "Passive Realtime Datacenter Fault Detection and Localization." Arjun Roy, Hongyi Zeng, Jasmeet Bagga, and Alex C. Snoeren. *USENIX ;login:* Volume 42, Number 3, Fall 2017.

6. "Lost in Space: Improving Inference of IPv4 Address Space Utilization." Alberto Dainotti, Karyn Benson, Alistair King, Bradley Huffaker, Eduard Glatz, Xenofontas Dimitropoulos, Philipp Richter, Alessandro Finamore, and Alex C. Snoeren. *IEEE Journal on Selected Areas in Communication* (JSAC), Volume 34, Number 6, June 2016.

7. "Enfold: Downclocking OFDM in Wi-Fi." Feng Lu, Patrick Ling, Geoffrey M. Voelker, and Alex C. Snoeren. *GetMobile: Mobile Computing and Communications* (MC²R), Volume 19, Number 2, April 2015.

8. "Achieving Congestion Diversity in Multi-hop Wireless Mesh Networks." Abhijeet A. Bhorkar, Tara Javidi, and Alex C. Snoeren. *IEEE Transactions on Mobile Computing* (TMC), Volume 14, Number 3, March 2015.

9. "Managing Contention with Medley." Feng Lu, Geoffrey M. Voelker, and Alex C. Snoeren. *IEEE Transactions on Mobile Computing* (TMC), Volume 14, Number 3, March 2015.

10. "Router Support for Fine-Grain Latency Measurements." Ramana Rao Kompella, Kirill Levchenko, Alex C. Snoeren, and George Varghese. *IEEE/ACM Transactions on Networking* (TON), Volume 20, Number 3, June 2012.

11. "Distributed Application Configuration, Management, and Visualization with Plush." Jeannie Albrecht, Christopher Tuttle, Ryan Braud, Darren Dao, Nikolay Topilski, Alex C. Snoeren, and Amin Vahdat. *ACM Transactions on Internet Technology* (TOIT), Volume 11, Number 2, December 2011.

12. "DieCast: Testing Distributed Systems with an Accurate Scale Model." Diwaker Gupta, Kashi V. Vishwanath, Marvin McNett, Amin Vahdat, Ken Yocum, Alex C. Snoeren, and Geoffrey M. Voelker. *ACM Transactions on Computer Systems* (TOCS), Volume 29, Number 2, May 2011.

13. "Privacy-Preserving Network Forensics." Mikhail Afanasyev, Tadayoshi Kohno, Justin Ma, Nick Murphy, Stefan Savage, Alex C. Snoeren, and Geoffrey M. Voelker. *Communications of the ACM* (CACM), Volume 54, Number 5, May 2011.

14. "Fault Localization via Risk Modeling," Ramana Rao Kompella, Jennifer Yates, Albert Greenberg, and Alex C. Snoeren. *IEEE Transactions on Dependable and Secure Computing* (TDSC), Volume 7, Number 4, October–December 2010.

15. "Usage Patterns in an Urban WiFi Network," Mikhail Afanasyev, Tsuwei Chen, Geoffrey M. Voelker, and Alex. C. Snoeren. *IEEE/ACM Transactions on Networking* (TON), Volume 18, Number 5, October 2010.

Plaid Ex1003
Page 125 of 372

16. "Difference Engine: Harnessing Memory Redundancy in Virtual Machines." Diwaker Gupta, Sangmin Lee, Michael Vrable, Stefan Savage, Alex C. Snoeren, George Varghese, Geoffrey M. Voelker, and Amin Vahdat. *Communications of the ACM* (CACM), Volume 53, Number 10, October 2010.

17. "Covenant: An Architecture for Cooperative Scheduling in 802.11 Wireless Networks," Ishwar Ramani, Ramana Rao Kompella, Sriram Ramabhadran, and Alex C. Snoeren. *IEEE Transactions on Wireless Communications* (TWC), Volume 9, Number 1, January 2010.

18. "Two Auction-Based Resource Allocation Environments: Design and Experience." Alvin AuYoung, Phil Buonadonna, Brent N. Chun, Chaki Ng, David C. Parkes, Jeff Shneidman, Alex C. Snoeren, and Amin Vahdat. In *Market Oriented Grid and Utility Computing*, Rajkumar Buyya and Kris Bubendorfer, Eds., Wiley, November 2009. pp. 513–539.

19. "Secure and Policy-Compliant Source Routing." Barath Raghavan, Patrick Verkaik, and Alex C. Snoeren. *IEEE/ACM Transactions on Networking* (TON), Volume 17, Number 3, June 2009.

20. "Difference Engine: Harnessing Memory Redundancy in Virtual Machines." Diwaker Gupta, Sangmin Lee, Michael Vrable, Stefan Savage, Alex C. Snoeren, George Varghese, Geoffrey M. Voelker, and Amin Vahdat. *USENIX ;login:* Volume 34, Number 2, April 2009.

21. "Managing Distributed Applications with Plush." Jeannie Albrecht, Ryan Braud, Darren Dao, Nikolay Topilski, Christopher Tuttle, Alex C. Snoeren, and Amin Vahdat. *USENIX ;login:* Volume 33, Number 1, February 2008.

22. "High Bandwidth Data Dissemination for Large-scale Distributed Systems." Dejan Kostić, Alex C. Snoeren, Amin Vahdat, Ryan Braud, Charles Killian, Jeannie Albrecht, James W. Anderson, Adolfo Rodriguez, and Erik VandeKieft. *ACM Transactions on Computer Systems* (TOCS), Volume 26, Number 1, February 2008.

23. "PlanetLab Application Management using Plush." Jeannie Albrecht, Christopher Tuttle, Alex C. Snoeren, and Amin Vahdat. *ACM Operating Systems Review* (OSR), Volume 40, Number 1, January 2006.

24. "Single-Packet IP Traceback." Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Beverly Schwartz, Stephen T. Kent, and W. Timothy Strayer. *IEEE/ACM Transactions on Networking* (TON), Volume 10, Number 6, December 2002.

25. "FIRE: Flexible Intra-AS Routing Environment." Craig Partridge, Alex C. Snoeren, W. Timothy Strayer, Beverly Schwartz, Matthew Condell, and Isidro Castineyra. *IEEE Journal on Selected Areas in Communications* (JSAC), Volume 19, Number 3, March 2001.

Conference publications

26. "Local Frames: Exploiting Inherited Origins to Bypass Content Blockers." Alisha Ukani, Hamed Haddadi, Alex C. Snoeren, and Pete Snyder. *Proceedings of the ACM Conference on Computer and Communications Security* (CCS), Taipei, Taiwan, October 2025.

27. "Cuckoo for Clients: Disaggregated Cuckoo Hashing." Stewart Grant and Alex C. Snoeren. *Proceedings of the USENIX Annual Technical Conference* (ATC), Boston, MA, July 2025. (100/634 15.8% acceptance)

28. "PageFlex: Flexible and Efficient User-space Delegation of Linux Paging Policies with eBPF." Anil Yelam, Kan Wu, Zhiyuan Guo, Rajath Shashidhara, Stanko Novakovic, Suli Yang, Wei Xu, Alex C. Snoeren, and Kimberly Keeton. *Proceedings of the USENIX Annual Technical Conference* (ATC), Boston, MA, July 2025. (100/634 15.8%)

29. "Eden: Developer-Friendly Application-Integrated Far Memory." Anil Yelam, Stewart Grant, Saarth Deshpande, Nadav Amit, Radhika Niranjan Mysore, Amy Ousterhout, Marcos K. Aguilera, and Alex C. Snoeren. *Proceedings of the 22nd ACM/USENIX Symposium on Networked Systems Design and Implementation* (NSDI), Philadelphia, PA, April 2025.

Plaid Ex1003
Page 126 of 372

30. "Realizing RotorNet: Toward Practical Microsecond Scale Optical Networking." William M. Mellette, Alex Forencich, Rukshani Athapathu, Alex C. Snoeren, George Papen, and George Porter. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), Sydney, Australia, August 2024. (62/366 16.9%)

31. "Cloud-Scale Characterization of Remote Procedure Calls." Korakit Seemakhupt, Brent E. Stephens, Samira Khan, Sihang Liu, Hassan Wassel, Soheil Hassas Yeganeh, Alex C. Snoeren, Arvind Krishnamurthy, David Culler, and Hank M. Levy. *Proceedings of the ACM Symposium on Operating System Principles* (SOSP), Koblenz, Germany, October 2023. (43/229 18.8%)

32. "Memory Management in ActiveRMT: Towards Runtime-programmable Switches." Radjeep Das and Alex C. Snoeren. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), New York, NY, September 2023. (71/323 22%) **Best paper.**

33. "Destination Unreachable: Characterizing Internet Outages and Shutdowns." Zachary S. Bischof, Kennedy Pitcher, Esteban Carisimo, Amanda Meng, Rafael Bezerra Nunes, Ramakrishna Padmanabhan, Margaret E. Roberts, Alex C. Snoeren, and Alberto Dainotti. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), New York, NY, September 2023. (71/323 22%)

34. "Access Denied: Assessing Physical Risks to Internet Access Networks." Alexander Marder, Zesen Zhang, Ricky Mok, Ramakrishna Padmanabhan, Bradley Huffaker, Matthew Luckie, Alberto Dainotti, kc claffy, Alex C. Snoeren, and Aaron Schulman. *Proceedings of the USENIX Security Symposium*, Anaheim, California, August 2023.

35. "Rosebud: Making FPGA-accelerated Middlebox Development More Pleasant." Moein Khazraee, Alex Forencich, George C. Papen, Alex C. Snoeren, and Aaron Schulman. *Proceedings of the 28th International Conference on Architectural Support for Programming Languages and Operating Systems* (ASPLOS), Vancouver, Canada, March 2023. (151/600 25.2%)

36. "Scaling Beyond Packet Switch Limits with Multiple Dataplanes." Yibo Guo, William M. Mellette, Alex C. Snoeren, and George Porter. *Proceedings of the ACM Conference on emerging Networking EXperiments and Technologies* (CoNEXT), Rome, Italy, December 2022. (29/151 19.2%)

37. "Measuring UID Smuggling in the Wild." Audrey Randall, Peter Snyder, Alisha Ukani, Alex C. Snoeren, Geoffrey M. Voelker, Stefan Savage, and Aaron Schulman. *Proceedings of the ACM Internet Measurement Conference* (IMC), Nice, France, October 2022. (56/212 26.4%)

38. "Mind Your MANRS: Measuring the MANRS Ecosystem." Ben Du, Cecilia Testart, Romain Fontugne, Gautam Akiwate, Alex C. Snoeren, and kc Claffy. *Proceedings of the ACM Internet Measurement Conference* (IMC), Nice, France, October 2022. (56/212 26.4%)

39. "Time-division TCP for Reconfigurable Data Center Networks." Shawn Shuoshuo Chen, Weiyang Wang, Christopher Canel, Srinivasan Seshan, Alex C. Snoeren, and Peter Steenkiste. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), Amsterdam, The Netherlands, August 2022. (55/279 19.7%)

40. "Quantifying Nations' Exposure to Traffic Observation and Selective Tampering." Alexander Gamero-Garrido, Esteban Carisimo, Shuai Hao, Bradley Huffaker, Alex C. Snoeren, and Alberto Dainottti. *Proceedings of the Passive and Active Measurement Conference* (PAM), Virtual, March 2022. (30/62 48.4%)

41. "IRR Hygiene In the RPKI Era." Ben Du, Gautam Akiwate, Thomas Krenc, Cecilia Testart, Alexander Marder, Bradley Huffaker, Alex C. Snoeren, and kc claffy. *Proceedings of the Passive and Active Measurement Conference* (PAM), Virtual, March 2022. (30/62 48.4%)

42. "Identifying ASes of State-owned Internet operators." Esteban Carisimo, Alex Gamero-Garrido, Alex C. Snoeren, and Alberto Dainotti. *Proceedings of the ACM Internet Measurement Conference* (IMC), Virtual, November 2021. (55/196 28%)

43. "Inferring Cloud Interconnections: Validation, Geolocation, and Routing Behavior." Alex Marder, kc claffy, and Alex C. Snoeren. *Proceedings of the Passive and Active Measurements Conference* (PAM), Brandenburg, Germany, March 2021. (33/75 44%)

44. "SmartNIC Performance Isolation with FairNIC: Programmable Networking for the Cloud." Stewart Grant, Anil Yelam, Maxwell Bland, and Alex C. Snoeren. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), New York, NY, August 2020. (44/250 21.6%)

45. "Corundum: An Open-Source 100G NIC." Alex Forencich, Alex C. Snoeren, George Porter, and George Papen. *Proceedings of the 28th IEEE International Symposium On Field-Programmable Custom Computing Machines* (FCCM), Fayetteville, AR, May 2020.

46. "Adapting TCP for Reconfigurable Datacenter Networks." Matthew Mukerjee, Christopher Canel, Weiyang Wang, Daehyeok Kim, Srinivasan Seshan, and Alex C. Snoeren. *Proceedings of the 17th ACM/USENIX Symposium on Networked Systems Design and Implementation* (NSDI), Santa Clara, CA, February 2020. (48/275 17.5%)

47. "Expanding Across Time to Deliver Bandwidth Efficiency and Low Latency." William M. Mellette, Rajdeep Das, Yibo Guo, Rob McGuinness, Alex C. Snoeren, and George Porter. *Proceedings of the 17th ACM/USENIX Symposium on Networked Systems Design and Implementation* (NSDI), Santa Clara, CA, February 2020. (17/79 21.5%)

48. "SparSDR: Sparsity-proportional Wideband SDRs." Moein Khazraee, Yeswanth Guddeti, Sam Crow, Alex C. Snoeren, Kirill Levchenko, Dinesh Bharadia, and Aaron Schulman. *Proceedings of the ACM Conference on Mobile Systems, Appliations and Services* (MobiSys), Seoul, Korea, June 2019. (40/172 23.3%)

49. "An Empirical Analysis of the Commercial VPN Ecosystem." Mohammad Taha Khan, Joe DeBlasio, Chris Kanich, Geoffrey M. Voelker, Alex C. Snoeren, and Narseo Vallina-Rodriguez. *Proceedings of the ACM Internet Measurement Conference* (IMC), Boston, Massachusetts, October 2018. (43/174 24.7%)

50. "Inferring Persistent Interdomain Congestion." Amogh Dhamdhere, David D. Clark, Alexander Gamero-Garrido, Matthew Luckie, Ricky K. P. Mok, Gautam Akiwate, Kabir Gogia, Vaibhav Bajpai, Alex C. Snoeren, and kc claffy. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), Budapest, Hungary, August 2018. (40/222 18.0%) **Best paper.**

51. "Tracking Ransomware End-to-end." Danny Yuxing Huang, Maxwell Matthaios Aliapoulios, Vector Guo Li, Luca Invernizzi, Elie Bursztein, Kylie McRoberts, Jonathan Levin, Kirill Levchenko, Alex C. Snoeren, and Damon McCoy. *Proceedings of the IEEE Symposium on Security and Privacy* (OAKLAND), San Francisco, California, May 2018. (49/549 8.9%)

52. "Quantifying the Pressure of Legal Risks on Third-party Vulnerability Research." Alex Gamero-Garrido, Stefan Savage, Kirill Levchenko, and Alex C. Snoeren. *Proceedings of the ACM Conference on Computer and Communications Security* (CCS), Dallas, Texas, November 2017. (151/843 17.9%)

53. "Tripwire: Inferring Internet Site Compromise." Joe DeBlasio, Stefan Savage, Geoffrey M. Voelker, and Alex C. Snoeren. *Proceedings of the ACM Internet Measurement Conference* (IMC), London, England, November 2017. (42/179 23.5%)

54. "Exploring the Dynamics of Search Advertiser Fraud." Joe DeBlasio, Saikat Guha, Geoffrey M. Voelker and Alex C. Snoeren. *Proceedings of the ACM Internet Measurement Conference* (IMC), London, England, November 2017. (42/179 23.5%)

Plaid Ex1003
Page 128 of 372

55. "RotorNet: A Scalable, Low-complexity, Optical Datacenter Network." William M. Mellette, Rob McGuinness, Arjun Roy, Alex Forencich, George Papen, Alex C. Snoeren, and George Porter. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), Los Angeles, California, August 2017. (36/250 14.4%)

56. "Using Indirect Routing to Recover from Network Traffic Scheduling Estimation Error." Conglong Li, Matthew Mukerjee, David G. Andersen, Srinivasan Seshan, Michael Kaminsky, George Porter, and Alex C. Snoeren. *Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems* (ANCS), Beijing, China, May 2017. (15/55 27.3%)

57. "Pinning Down Abuse on Google Maps." Danny Yuxing Huang, Doug Grundman, Kurt Thomas, Elie Bursztein, Abhishek Kumar, Kirill Levchenko, and Alex C. Snoeren. *Proceedings of the International World Wide Web Conference* (WWW), Perth, Australia, April 2017. (164/966 17.0%)

58. "Passive Realtime Datacenter Fault Detection and Localization." Arjun Roy, Hongyi Zeng, Jasmeet Bagga, and Alex C. Snoeren. *Proceedings of the 14th ACM/USENIX Symposium on Networked Systems Design and Implementation* (NSDI), Boston, MA, March 2017. (46/253 18.2%)

59. "Scheduling Techniques for Hybrid Circuit/Packet Networks." He Liu, Matthew K. Mukerjee, Conglong Li, Nicolas Feltman, George Papen, Stefan Savage, Srinivasan Seshan, Geoffrey M. Voelker, David G. Andersen, Michael Kaminsky, George Porter, and Alex C. Snoeren. *Proceedings of the ACM Conference on emerging Networking EXperiments and Technologies* (CoNEXT), Heidelberg, Germany, December 2015. (41/196 20.9%)

60. "Empirical Analysis of Search Advertising Strategies." Bhanu C. Vattikonda, Vacha Dave, Saikat Guha, and Alex C. Snoeren. *Proceedings of the ACM Internet Measurement Conference* (IMC), Tokyo, Japan, October 2015. (44/169 26%)

61. "Revisiting Internet Background Radiation for Opportunistic Network Analysis." Karyn Benson, Alberto Dainotti, kc claffy, Alex C. Snoeren, and Michael Kallitsis. *Proceedings of the ACM Internet Measurement Conference* (IMC), Tokyo, Japan, October 2015. (44/169 26%)

62. "Security by Any Other Name: On the Effectiveness of Provider Based Email Security." Ian Foster, Jon Larson, Max Masich, Alex C. Snoeren, Stefan Savage, and Kirill Levchenko. *Proceedings of the ACM Conference on Computer and Communications Security* (CCS), Denver, Colorado, October 2015. (128/646 19.8%)

63. "Inside the Social Network's (Datacenter) Network." Arjun Roy, Hongyi Zeng, Jasmeet Bagga, George Porter, and Alex C. Snoeren. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), London, England, August 2015. (40/256 15.6%)

64. "Interpreting Advertiser Intent in Sponsored Search." Bhanu C. Vattikonda, Santhosh Kodipaka, Hongyan Zhou, Vacha Dave, Saikat Guha, and Alex C. Snoeren. *Proceedings of the ACM SIGKDD Conference* (KDD), Sydney, Australia, August 2015. (64/187 34.2%)

65. "Deniable Liaisons." Ahbinav Narain, Nick Feamster, and Alex C. Snoeren. *Proceedings of the ACM Conference on Computer and Communications Security* (CCS), Scottsdale, Arizona, November 2014. (114/585 19.5%)

66. "On the Security of Mobile Cockpit Information Systems." Devin Lundberg, Brown Fairnholt, Edward Sullivan, Ryan Mast, Stephen Checkoway, Stefan Savage, Alex C. Snoeren, and Kirill Levchenko. *Proceedings of the ACM Conference on Computer and Communications Security* (CCS), Scottsdale, Arizona, November 2014. (114/585 19.5%)

67. "Blender: Upgrading Tenant-based Data Center Networking." Kevin C. Webb, Arjun Roy, Kenneth Yocum, and Alex C. Snoeren. *Proceedings of the ACM/IEEE Symposium on Architectures for Networking and Communications Systems* (ANCS), Marina del Rey, California, October 2014. (19/56 33.9%)

68. "Enfold: Downclocking OFDM in WiFi." Feng Lu, Patrick Ling, Geoffrey M. Voelker, and Alex C. Snoeren. *Proceedings of the Annual International Conference on Mobile Computing and Networking* (MobiCom), Maui, Hawaii, September 2014. (36/220 16.4%)

69. "Circuit Switching Under the Radar with REACToR." He Liu, Feng Lu, Alex Forencich, Rishi Kapoor, Malveeka Tewari, Geoffrey M. Voelker, George Papen, Alex C. Snoeren, and George Porter. *Proceedings of the ACM/USENIX Symposium on Networked Systems Design and Implementation* (nsdi), Seattle, WA, April 2014. (38/213 17.8%)

70. "Exposing Inconsistent Web Search Results with Bobble." Xinyu Xing, Wei Meng, Dan Doozan, Nick Feamster, Wenke Lee, and Alex C. Snoeren. *Proceedings of the Passive and Active Measurements Conference* (pam), Los Angeles, CA, March 2014. (24/76 31.6%)

71. "Botcoin: Monetizing Stolen Cycles." Danny Yuxing Huang, Hitesh Dharmdasani, Sarah Meiklejohn, Vacha Dave, Chris Grier, Damon McCoy, Stefan Savage, Nicholas Weaver, Alex C. Snoeren, and Kirill Levchenko. *Proceedings of the Network and Distributed System Security Symposium* (ndss), San Diego, CA, February 2014. (55/293 18.8%)

72. "Take This Personally: Pollution Attacks on Personalized Services." Xinyu Xing, Wei Ming, Dan Doozan, Alex C. Snoeren, Nick Feamster, and Wenke Lee. *Proceedings of the USENIX Security Symposium,* Washington D.C., August 2013. (45/277 16.2%)

73. "Quantifying the Benefits of Joint Content and Network Routing." Vytautas Valancius, Bharath Ravi, Nick Feamster, and Alex C. Snoeren. *Proceedings of ACM SIGMETRICS,* Pittsburgh, Pennsylvania, June 2013. (27/196 13.8%)

74. "SloMo: Downclocking WiFi Communication." Feng Lu, Geoffrey M. Voelker, and Alex C. Snoeren. *Proceedings of the 10th USENIX Symposium on Networked Systems Design & Implementation* (nsdi), Lombard, Illinois, April 2013. (38/171 22.2%)

75. "Practical TDMA for Datacenter Ethernet." Bhanu Vattikonda, George Porter, Amin Vahdat, and Alex C. Snoeren. *Proceedings of the ACM European Conference on Computer Systems* (EuroSys), Bern, Switzerland, April 2012. (27/178 15.2%)

76. "*scc:* Cluster Storage Provisioning Informed by Application Characteristics and SLAs." Harsha V. Madhyastha, John C. McCullough, George Porter, Rishi Kapoor, Stefan Savage, Alex C. Snoeren, and Amin Vahdat. *Proceedings of the USENIX Conference on File and Storage Technologies* (fast), San Jose, California, February 2012. (26/137 19.0%)

77. "On the Empirical Performance of Self-calibrating WiFi Location Systems." Daniel J. Turner, Stefan Savage, and Alex C. Snoeren. *Proceedings of the IEEE Conference on Local Computer Networks* (lcn), Bonn, Germany, October 2011. (50/181 27.6%)

78. "Evaluating the Effectiveness of Model-Based Power Characterization." John McCullough, Yuvraj Agarwal, Jaideep Chandrashekhar, Sathyanarayan Kuppuswamy, Alex C. Snoeren, and Rajesh Gupta, *Proceedings of the USENIX Annual Technical Conference* (atc), Portland, Oregon, June 2011. (27/180 15.0%)

79. "California Fault Lines: Understanding the Causes and Impact of Network Failures." Daniel Turner, Kirill Levchenko, Alex C. Snoeren, and Stefan Savage, *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (sigcomm), New Delhi, India, August 2010. (33/276 12.0%)

80. "Stout: An Adaptive Interface to Scalable Cloud Storage." John C. McCullough, John Dunagan, Alec Wolman, and Alex C. Snoeren, *Proceedings of the USENIX Annual Technical Conference* (atc), Boston, Massachusetts, June 2010. (24/141 17.0%)

81. "Neon: System Support for Derived Data Management." Qing Zhang, John McCullough, Justin Ma, Nabil Schear, Michael Vrable, Amin Vahdat, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage, *Proceedings of the ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments* (VEE), Pittsburgh, Pennsylvania, March 2010. (14/51 27.5%)

82. "The Importance of Being Overheard: Throughput Gains in Wireless Mesh Networks." Mikhail Afanasyev and Alex C. Snoeren. *Proceedings of the ACM/USENIX Internet Measurement Conference* (IMC), Chicago, Illinois, November 2009. (41/183 22.4%)

83. "Evaluating the Impact of Inaccurate Information in Utility-Based Scheduling." Alvin AuYoung, Amin Vahdat, and Alex C. Snoeren. *Proceedings of the ACM Conference on High Performance Computing and Networking* (SC), Portland, Oregon, November 2009. (59/261 22.6%)

84. "Every Microsecond Counts: Tracking Fine-Grain Latencies with a Lossy Difference Aggregator." Ramana Rao Kompella, Kirill Levchenko, Alex C. Snoeren, and George Varghese. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), Barcelona, Spain, August 2009. (27/270 10%)

85. "Softspeak: Making VoIP Play Well in Existing 802.11 Deployments." Patrick Verkaik, Yuvraj Agarwal, Rajesh Gupta, and Alex C. Snoeren. *Proceedings of the USENIX Symposium on Networked Systems Design & Implementation* (NSDI), Boston, Massachusetts, April 2009. (31/163 19%)

86. "Difference Engine: Harnessing Memory Redundancy in Virtual Machines." Diwaker Gupta, Sangmin Lee, Michael Vrable, Stefan Savage, Alex C. Snoeren, George Varghese, Geoffrey M. Voelker, and Amin Vahdat. *Proceedings of the ACM/USENIX Symposium on Operating System Design & Implementation* (OSDI), San Diego, California, December 2008. (26/193 13.5%) **Best paper.**

87. "Analysis of a Mixed-Use Urban WiFi Network: When Metropolitan becomes Neapolitan." Mikhail Afanasyev, Tsuwei Chen, Geoffrey M. Voelker, and Alex C. Snoeren. *Proceedings of the 6th ACM/USENIX Internet Measurement Conference* (IMC), Vouliagmeni, Greece, October 2008. (32/185 17.3%)

88. "Efficiency through Eavesdropping: Link-layer Packet Caching." Mikhail Afanasyev, David G. Andersen, and Alex C. Snoeren. *Proceedings of the USENIX Symposium on Networked Systems Design & Implementation* (NSDI), San Francisco, California, April 2008. (30/175 17.1%)

89. "Remote Control: Distributed Application Configuration, Management, and Visualization with Plush." Jeannie Albrecht, Ryan Braud, Darren Dao, Nikolay Topilski, Christopher Tuttle, Alex C. Snoeren, and Amin Vahdat. *Proceedings of the 21st USENIX Large Installation System Administration Conference* (LISA), Dallas, Texas, November 2007.

90. "Cloud Control with Distributed Rate Limiting." Barath Raghavan, Kashi Vishwanath, Sriram Ramabhadran, Kenneth Yocum, and Alex C. Snoeren. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), Kyoto, Japan, August 2007. (35/258 13.6%) **Best student paper.**

91. "Automating Cross-Layer Diagnosis of Enterprise Wireless Networks." Yu-Chung Cheng, Mikhail Afanasyev, Patrick Verkaik, Peter Benko, Jennifer Chiang, Alex C. Snoeren, Stefan Savage, and Geoffrey M. Voelker. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), Kyoto, Japan, August 2007. (35/258 13.6%)

92. "Wresting Control from BGP: Scalable Fine-grained Route Control." Patrick Verkaik, Dan Pei, Tom Scholl, Aman Shaikh, Alex C. Snoeren, and Jacobus van der Merwe, *Proceedings of the USENIX Annual Technical Conference* (ATC), Santa Clara, California, June 2007. (24/102 23.5%)

93. "Detection and Localization of Network Black Holes." Ramana Rao Kompella, Jennifer Yates, Albert Greenberg, and Alex C. Snoeren. *Proceedings of the IEEE International Conference on Computer Communications* (INFOCOM), Anchorage, Alaska, May 2007. (252/1400 18%)

94. "A Performance Analysis of Indirect Routing." Joshua Opos, Sriram Rambhadaran, Andrew Terry, Joseph Pasquale, Alex C. Snoeren, and Amin Vahdat. *Proceedings of the IEEE International Parallel & Distributed Processing Symposium* (IPDPS), Long Beach, California, March 2007. (109/419 26.0%)

95. "Jigsaw: Solving the Puzzle of Enterprise 802.11 Analysis." Yu-Chung Cheng, John Bellardo, Peter Benko, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), Pisa, Italy, September 2006. (37/298 12.4%)

96. "Loose Synchronization for Large-Scale Networked Systems." Jeannie Albrecht, Christopher Tuttle, Alex C. Snoeren, and Amin Vahdat. *Proceedings of the USENIX Annual Technical Conference* (ATC), Boston, Massachusetts, June 2006. (21/153 13.7%)

97. "Service Placement in Shared Wide-Area Platforms." David Oppenheimer, Brent Chun, David Patterson, Alex C. Snoeren, and Amin Vahdat. *Proceedings of the USENIX Annual Technical Conference* (ATC), Boston, Massachusetts, June 2006. (21/153 13.7%)

98. "To Infinity and Beyond: Time-Warped Network Emulation." Diwaker Gupta, Kenneth Yocum, Marvin McNett, Alex C. Snoeren, Amin Vahdat, and Geoffrey M. Voelker. *Proceedings of the USENIX Symposium on Networked Systems Design & Implementation* (NSDI), San Jose, California, May 2006. (28/110 25.5%)

99. "Scalability, Fidelity and Containment in the Potemkin Virtual Honeyfarm." Michael Vrable, Justin Ma, Jay Chen, David Moore, Erik VandeKieft, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. *Proceedings of the ACM Symposium on Operating System Principles* (SOSP), Brighton, United Kingdom, October 2005. (20/155 12.9%)

100. "IP Fault Localization Via Risk Modeling." Ramana Rao Kompella, Jennifer Yates, Albert Greenberg, and Alex C. Snoeren. *Proceedings of the USENIX Symposium on Networked Systems Design & Implementation* (NSDI), Boston, Massachusetts, May 2005. (25/112 22.3%)

101. "Maintaining High-Bandwidth under Dynamic Network Conditions." Dejan Kostić, Ryan Braud, Charles Killian, Erik VandeKieft, James W. Anderson, Alex C. Snoeren, and Amin Vahdat. *Proceedings of the USENIX Annual Technical Conference* (ATC), Anaheim, California, April 2005. (24/118 20.3%)

102. "Routing in an Internet-Scale Network Emulator." Jay Chen, Diwaker Gupta, Kashi V. Vishwanath, Alex C. Snoeren and Amin Vahdat. *Proceedings of the IEEE/ACM International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems* (MASCOTS), Volendam, The Netherlands, October 2004. (67/172 40%)

103. "A System for Authenticated Policy-Compliant Routing." Barath Raghavan and Alex C. Snoeren. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), Portland, Oregon, September 2004. (31/340 9.1%)

104. "Practical Lazy Scheduling in Wireless Sensor Networks." Ramana Rao Kompella and Alex C. Snoeren. *Proceedings of the 1st ACM Conference on Embedded Networked Sensor Systems* (SENSYS), Los Angeles, California, November 2003. (24/135 17.8%)

105. "Best-Path vs. Multi-Path Overlay Routing." David G. Andersen, Alex C. Snoeren, and Hari Balakrishnan. *Proceedings of the 1st ACM/USENIX Internet Measurement Conference* (IMC), Miami, Florida, October 2003. (32/109 29.4%)

106. "TESLA: A Transparent, Extensible Session-Layer Architecture for End-to-end Network Services." Jon Salz, Alex C. Snoeren, and Hari Balakrishnan. *Proceedings of the USENIX Symposium on Internet Technologies and Systems* (USITS), Seattle, Washington, March 2003. (21/76 27.6%)

107. "Mesh-Based Content Routing using XML." Alex C. Snoeren, Kenneth Conley, and David K. Gifford. *Proceedings of the 18th ACM Symposium on Operating System Principles* (SOSP), Banff, Canada, October 2001. (17/85 20%)

Plaid Ex1003
Page 132 of 372

108. "Hash-Based IP Traceback." Alex C. Snoeren, Craig Partridge, Luis A. Sanchez, Christine E. Jones, Fabrice Tchakountio, Stephen T. Kent, and W. Timothy Strayer. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), San Diego, California, August 2001. (23/203 11.3%) **Best student paper.**

109. "Hardware Support for a Hash-Based IP Traceback." Luis A. Sanchez, Walter C. Milliken, Alex C. Snoeren, Fabrice Tchakountio, Christine E. Jones, Stephen T. Kent, Craig Partridge, and W. Timothy Strayer. *Proceedings of the DARPA Information Survivability Conference and Exposition* (DISCEX), Anaheim, California, June 2001.

110. "Fine-Grained Failover Using Connection Migration." Alex C. Snoeren, David G. Andersen, and Hari Balakrishnan. *Proceedings of the USENIX Symposium on Internet Technologies and Systems* (USITS), San Francisco, California, March 2001. (19/70 27.1%)

111. "FIRE: Flexible Intra-AS Routing Environment." Craig Partridge, Alex C. Snoeren, W. Timothy Strayer, Beverly Schwartz, Matthew Condell, and Isidro Castineyra. *Proceedings of the ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication* (SIGCOMM), Stockholm, Sweden, August 2000. (26/238 10.9%)

112. "An End-to-End Approach to Host Mobility." Alex C. Snoeren and Hari Balakrishnan. *Proceedings of the Annual International Conference on Mobile Computing and Networking* (MOBICOM), Boston, Massachusetts, August 2000. (28/226 12.3%)

113. "Adaptive Inverse Multiplexing for Wide-Area Wireless Networks." Alex C. Snoeren. *Proceedings of the IEEE Conference on Global Communications* (GLOBECOM), *Global Internet Symposium*, Rio de Janeiro, Brazil, December 1999. (28/117 23.9%)

Refereed workshop & short conference papers

114. "Sublet Your Subnet: Inferring IP Leasing in the Wild." Ben Du, Romain Fontugne, Cecilia Testart, Alex C. Snoeren, and kc Claffy, *Proceedings of the ACM Internet Measurement Conference* (IMC), Madrid, Spain, November 2024. (short paper) (54/253 21.3%)

115. "IRRegularities in the Internet Routing Registry." Ben Du, Katherine Izhikevich, Sumanth Rao, Cecilia Testart, Gautam Akiwate, Alex C. Snoeren, and kc Claffy. *Proceedings of the ACM Internet Measurement Conference* (IMC), Montreal, Canada, October 2023. (short paper) (52/208 25%)

116. "Limited Access: The Truth Behind Far Memory." Anil Yelam, Stewart Grant, Enze Liu, Radhika Niranjan Mysore, Marcos K. Aguilera, Amy Ousterhout, and Alex C. Snoeren. *Proceedings of the Workshop on Resource Disaggregation and Serverless Computing* (WORDS), October 2023. (8/14 57.1%)

117. "In the Line of Fire: Risks of DPI-triggered Data Collection." Ariana Mirian, Alisha Ukani, Ian Foster, Gautam Akiwate, Taner Halicioglu, Cindy Moore, Alex C. Snoeren, Geoffrey M. Voelker, and Stefan Savage. *Proceedings of the Workshop on Cyber Security Experimentation and Test* (CSET), Marina del Rey, CA, August 2023. (9/19 47.4%)

118. "Locked-In during Lock-Down: Undergraduate Life on the Internet in a Pandemic." Alisha Ukani, Ariana Mirian, and Alex C. Snoeren. *Proceedings of the ACM Internet Measurement Conference* (IMC), Virtual, November 2021. (short paper) (55/196 28%)

119. "In-network Contention Resolution for Disaggregated Memory." Stewart Grant and Alex C. Snoeren. *Proceedings of the Workshop on Resource Disaggregation and Serverless Computing* (WORDS), April 2021. (5/7 71.4%)

120. "Enabling Active Networking on RMT Hardware." Rajdeep Das and Alex C. Snoeren. *Proceedings of the ACM Workshop on Hot Topics in Networks* (HOTNETS), Chicago, IL, November 2020. (30/120 25%)

Plaid Ex1003
Page 133 of 372

121. "Triton: A Software-Reconfigurable Federated Avionics Testbed." Sam Crow, Brown Farinholt, Brian Johannesmeyer, Karl Koscher, Stephen Checkoway, Stefan Savage, Aaron Schulman, Alex C. Snoeren, and Kirill Levchenko. *Proceedings of the USENIX Workshop on Cyber Security Experimentation and Test* (CSET), Santa Clara, CA, August 2019. (19/61 31.1%)

122. "LEED: A Lightwave Energy-Efficient Datacenter." Y. Shaya Fainman, Joseph Ford, William M. Mellette, Shayan Mookherjea, George Porter, Alex C. Snoeren, George Papen, Saman Saeedi, John Cunningham, Ashok Krishnamoorthy, Michael Gehl, Christopher T. DeRose, Paul S. Davids, Douglas C. Trotter, Andrew L. Starbuck, Christina M. Dallo, Dana Hood, Andrew Pomerene, and Anthony Lentine. *Proceedings of the Optical Fiber Communications Conference and Exhibition* (OFC), San Diego, CA, March 2019.

123. "Cloud Datacenter SDN Monitoring: Experiences and Challenges." Arjun Roy, Deepak Bansal, David Brumley, Harish Kumar Chandrappa, Parag Sharma, Rishabh Tewari, Behnaz Arzani, and Alex C. Snoeren. *Proceedings of the ACM Internet Measurement Conference* (IMC), Boston, Massachusetts, October 2018. (short paper) (43/174 24.7%)

124. "Measuring Profitability of Alternative Crypto-currencies." Danny Yuxing Huang, Kirill Levchenko, and Alex C. Snoeren. *Proceedings of the International Conference on Financial Cryptography and Data Security* (FC), Nieuwpoort, Curacao, February 2018. (short paper) (29/110 26.4%)

125. "P-FatTree: A Multi-channel Datacenter Network Topology." William M. Mellette, Alex C. Snoeren, and George Porter. *Proceedings of the ACM Workshop on Hot Topics in Networks* (HOTNETS), Atlanta, GA, November 2016. (30/108 27.8%)

126. "CQIC: Revisiting Cross-Layer Congestion Control for Cellular Networks." Feng Lu, Hao Du, Ankur Jain, Geoffrey M. Voelker, Alex C. Snoeren, and Andreas Terzis. *Proceedings of the ACM International Workshop on Mobile Computing Systems and Applications* (HOTMOBILE), Sante Fe, NM, February 2015. (23/80 28.8%)

127. "Bullet Trains: A Study of NIC Burst Behavior at Microsecond Timescales." Rishi Kapoor, Alex C. Snoeren, Geoffrey M. Voelker, and George Porter. *Proceedings of the ACM International Conference on Emerging Networking Experiments and Technologies* (CoNEXT), Santa Barbara, California, December 2013. (short paper) (15/83 18.1%)

128. "Scalable Coordination of a Tightly-Coupled Service in the Wide Area." Kevin C. Webb, Bhanu C. Vattikonda, Kenneth Yocum, and Alex C. Snoeren. *Proceedings of the ACM Conference on Timely Results in Operating Systems* (TRIOS), Farmington, Pennsylvania, November 2013. (10/17 58.8%)

129. "A Comparison of Syslog and IS-IS for Monitoring Link State." Daniel Turner, Kirill Levchenko, Stefan Savage, and Alex C. Snoeren. *Proceedings of the ACM/USENIX Internet Measurement Conference* (IMC), Barcelona, Spain, October 2013. (short paper) (42/178 23.6%)

130. "High-Fidelity Switch Models for Software-Defined Network Emulation." Danny Yuxing Huang, Ken Yocum, and Alex C. Snoeren. *Proceedings of the ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking* (HOTSDN), Hong Kong, China, August 2013. (24/84 28.6%)

131. "Challenges in the Emulation of Large Scale Software Defined Networks." Arjun Roy, Kenneth Yocum, and Alex C. Snoeren. *Proceedings of the Asia-Pacific Workshop on Systems* (APSYS), Singapore, July 2013. (20/73 27.4%)

132. "Weighted Fair Queuing with Differential Dropping." Feng Lu, Geoffrey M. Voelker, and Alex C. Snoeren. *Proceedings of the IEEE INFOCOM Mini-Conference*, Orlando, Florida, March 2012. (395/1578 25.0%)

133. "Achieving Congestion Diversity in Wireless Ad-hoc Networks." Abhijeet A. Bhorkar, Tara Javidi, and Alex C. Snoeren. *Proceedings of the IEEE INFOCOM Mini-Conference*, Shanghai, China, April 2011. (427/1823 23.4%)

134. "Topology Switching for Data Center Networks." Kevin Webb, Alex C. Snoeren, and Kenneth Yocum. *Proceedings of the USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services* (HotICE), Boston, Massachusetts, March 2011. (14/26 54%)

135. "mPlane: An Architecture For Scalable Fault Localization." Ramana Rao Kompella, Alex C. Snoeren, and George Varghese. *Proceedings of the ACM CoNext Workshop on Re-Architecting the Internet* (ReArch), Rome, Italy, December 2009. (13/48 27.1%)

136. "Enlisting ISPs to Improve Online Privacy: IP Address Mixing by Default." Barath Raghavan, Tadayoshi Kohno, Alex C. Snoeren, and David Wetherall. *Proceedings of the Privacy Enhancing Technology Symposium* (pets), Seattle, Washington, August 2009.

137. "Decongestion Control." Barath Raghavan and Alex C. Snoeren. *Proceedings of the ACM Workshop on Hot Topics in Networks* (HotNets), Irvine, California, November 2006. (23/114 20.2%)

138. "PRIMED: Community-of-Interest-Based DDoS Mitigation." Patrick Verkaik, Oliver Spatscheck, Jacobus van der Merwe, and Alex C. Snoeren. *Proceedings of the ACM SIGCOMM Workshop on Large Scale Attack Defense*, Pisa, Italy, September 2006. (11/32 34.4%)

139. "Cross-layer Visibility as a Service." Ramana Rao Kompella, Albert Greenberg, Jennifer Rexford, Alex C. Snoeren, and Jennifer Yates. *Proceedings of the ACM Workshop on Hot Topics in Networks* (HotNets), College Park, Maryland, October 2005. (20/100 20%)

140. "Addressing Strategic Behavior in a Deployed Microeconomic Resource Allocator." Chaki Ng, Philip Buonadonna, Brent N. Chun, Alex C. Snoeren, and Amin Vahdat. *Proceedings of the Workshop on Economics of Peer-to-Peer Systems* (P2PEcon), Philadelphia, Pennsylvania, August 2005. (11/36 30.6%)

141. "Cooperative Packet Scheduling via Pipelining in 802.11 Wireless Networks." Ramana Rao Kompella, Sriram Ramabhadran, Ishwar Ramani, and Alex C. Snoeren. *Proceedings of the Workshop on Experimental Approaches to Wireless Network Design and Analysis* (e-wind), Philadelphia, Pennsylvania, August 2005. (14/32 43.8%)

142. "Brief Announcement: The Overlay Network Content Distribution Problem." Charles Killian, Michael Vrable, Alex C. Snoeren, Amin Vahdat, and Joseph Pasquale. *Proceedings of the ACM Symposium on Principles of Distributed Computing* (podc '05), Las Vegas, NV, July 2005. (short paper) (58/206 28.2%)

143. "Why Markets Could (But Don't Currently) Solve Resource Allocation Problems in Systems." Jeffrey Shneidman, Chaki Ng, David C. Parkes, Alvin AuYoung, Alex C. Snoeren, Amin Vahdat, and Brent N. Chun. *Proceedings of the USENIX Workshop on Hot Topics in Operating Systems* (HotOS), Santa Fe, New Mexico, June 2005. (24/124 19.4%)

144. "Mirage: A Microeconomic Resource Allocation System for Sensornet Testbeds." Brent N. Chun, Philip Buonadonna, Alvin AuYoung, Chaki Ng, David C. Parkes, Jeffrey Shneidman, Alex C. Snoeren, and Amin Vahdat. *Proceedings of the IEEE Workshop on Embedded Networked Sensors* (EmNetS), Sydney, Australia, May 2005. (17/43 39.5%)

145. "Resource Allocation in Federated Distributed Computing Infrastructures." Alvin AuYoung, Brent N. Chun, Alex C. Snoeren, and Amin Vahdat. *Proceedings of the Workshop on Operating System and Architectural Support for the on demand IT InfraStructure* (oasis '04), Boston, Massachusetts, October 2004.

146. "Decoupling Policy from Mechanism in Internet Routing." Alex C. Snoeren and Barath Raghavan. *Proceedings of the ACM Workshop on Hot Topics in Networks* (HotNets), Cambridge, Massachusetts, November 2003. (23/119 19.3%)

147. "Priority Forwarding in Ad Hoc Networks with Self-Interested Parties." Barath Raghavan and Alex C. Snoeren. *Proceedings of the Workshop on Economics of Peer-to-Peer Systems* (P2PEcon), Berkeley, California, June 2003. (22/60 36.7%)

Plaid Ex1003
Page 135 of 372

148. "The Case for Service Provider Deployment of Super-Peers in Peer-to-Peer Networks." Sumeet Singh, Sriram Ramabhadran, Florin Baboescu, and Alex C. Snoeren. *Proceedings of the Workshop on Economics of Peer-to-Peer Systems* (P2PEcon), Berkeley, California, June 2003. (22/60 36.7%)

149. "Reconsidering Internet Mobility." Alex C. Snoeren, Hari Balakrishnan, and M. Frans Kaashoek. *Proceedings of the 8th Workshop on Hot Topics in Operating Systems* (HotOS), Elmau, Germany, May 2001.

150. "Automated Whole-System Diagnosis of Distributed Services Using Model-Based Reasoning." George Forman, Mudita Jain, Masoud Mansouri-Samani, Joseph Martinka, and Alex C. Snoeren. *Proceedings of the IFIP/IEEE Workshop on Distributed Systems: Operations and Management* (dsom '98), Newark, Delaware, October 1998. (24/48 50%)

Selected other works

151. "Optical networking within the Lightwave Energy-Efficient Datacenter project [Invited]." William M. Mellette, Alex Forencich, Jason Kelley, Joseph Ford, George Porter, Alex C. Snoeren, and George Papen, *Journal of Optical Communications and Networking*, Volume 12, Issue 12, December 2020.

152. "Toward Optical Switching in the Data Center." William M. Mellette, Alex C. Snoeren, and George Porter. *Proceedings of the IEEE International Conference on High Performance Switching and Routing* (hpsr), Bucharest, Romania, June 2018. (Invited paper)

153. "Estimating Profitability of Alternative Crypto-currencies." Danny Yuxing Huang, Kirill Levchenko, and Alex C. Snoeren. *UC San Diego Tech Report CS2017-1019*, December 2017.

154. "The Role of End-to-End Congestion Control in Networks with Fairness-Enforcing Routers." Barath Raghavan, John McCullough, and Alex C. Snoeren. *UC San Diego Tech Report CS2013-0998*, April 2013.

155. "On Failure in Managed Enterprise Networks." Daniel Turner, Kirill Levchenko, Jeffrey C. Mogul, Stefan Savage, Alex C. Snoeren. *Hewlett-Packard Labs Technical Report HPL-2012-101*. May 2012.

156. "Suspending and Resuming Network Applications using Session Continuations." Alex C. Snoeren, Debashis Panigrahi, Shoubhik Mukhopadhyay, Hari Balakrishnan, and M. Frans Kaashoek. *UC San Diego Tech Report CS2008-0921*, April 2008.

157. "Universal Honeyfarm Containment." Jay Chen, John McCullough, and Alex C. Snoeren. *UC San Diego Tech Report CS2007-0902*, September 2007.

158. "The Overlay Network Content Distribution Problem." Charles Killian, Michael Vrable, Alex C. Snoeren, Amin Vahdat, and Joseph Pasquale. *UC San Diego Tech Report CS2005-0824*, May 2005.

159. "New Protocols to Support Internet Traceback." Craig Partridge, Christine Jones, David Waitzman, and Alex C. Snoeren. IETF Internet Draft, `draft-partridge-ippt-discuss-00.txt`, November 2001. *Work-in-progress, expired May 2002.*

160. "TCP Connection Migration." Alex C. Snoeren and Hari Balakrishnan. IETF Internet Draft, `draft-snoeren-tcp-migrate-00.txt`, November 2000. *Work-in-progress, expired May 2001.*

161. "A Short Overview of PROMISE: Automated Diagnosis of Network Printing Systems Using Model-Based Reasoning." George Forman, Mudita Jain, Masoud Mansouri-Samani, Joseph Martinka, and Alex C. Snoeren. *Hewlett-Packard Laboratories Technical Report HPL-98-71*, April 1998.

Plaid Ex1003
Page 136 of 372

Research funding

2023 Cisco, **"Virtualizing Programmable Switch Hardware through Active Networking"**, $179,999.

2022 Intel, **"Dynamic Memory Management on High-speed Programmable Switches"**, $13,275.

2021–24 VMWare, **"Multi-resource Scheduling for Rack-sale Computers"**, $240,000.

2020 Google, **"A Reconfigurable, Low-Latency Expander-based Cloud Network Fabric"**, $62,000.

2019–22 National Science Foundation (George Porter co-PI), **"Designing Efficient Cloud Datacenter Network Fabrics"**, CNS-1911104, $500,000.

2018 Microsoft, **Microsoft Research Award**, $25,000.

2017–24 Advanced Research Projects Agency-Energy (co-PI; George Papen PI, Alic, Fainman, Ford, Mookherjea, Porter UCSD co-PIs) **"LEED: A Lightwave Energy-Efficient Datacenter"**, $8,800,000.

2017 Facebook, **Facebook Faculty Research Award**, $30,000.

2016–19 National Science Foundation (co-PI; Kirill Levchenko PI, Ranjit Jhala, Stefan Savage co-PIs), **"Foundations of Secure Cyber-Physical Systems of Systems"**, CNS-1646493, $560,000.

2016–19 National Science Foundation (Kirill Levchenko, George Porter, Stefan Savage, Geoff Voelker co-PIs), **"A Dual-Purpose Data Analytics Laboratory"**, CNS-1629973, $548,688.

2016–18 Hewlett Foundation (co-PI; Stefan Savage PI, Kirill Levchenko, George Voelker co-PIs), **"Reasoning about Zero-Day Vulnerabilities and Legal Risks on Third-party Vulnerability Research"**, $200,000.

2015–18 National Science Foundation, (co-PI; George Porter PI),**"Improving Network Performance and Efficiency through Multi-Channel Network Links"**, CNS-1565185, $1,100,000.

2014–17 National Science Foundation, **"Studying and Improving the Performance of Access Networks"**, CNS-1422240, $228,299.

2013–16 National Science Foundation, (co-PI; George Porter PI, George Papen, Joe Ford co-PIs), **"Hybrid Circuit/Packet Networking"**, CNS-1314921, $1,800,000.

2012–17 National Science Foundation, (senior staff; Stefan Savage PI), **"Incorporating the Social and Economic Milieu into Securing Cyberspace"**, CNS-1237264, $4,665,975.

2012–13 National Science Foundation, **"Personalization in the Information Age"**, CNS-1255274, $150,000.

2012 Broadcom (Ken Yocum co-PI), **"SDN Interfaces for Merchant Silicon"**, $40,000.

2012 Google (Ken Yocum co-PI), **"An App-Store Framework for Data Center Networks"**, $58,955.

2012 Ericsson (Ken Yocum co-PI), **"Topology Switching for Data Center Networks"**, $50,000

2011–14 National Science Foundation (Kirill Levchenko, Stefan Savage co-PIs), **"Understanding Network Failure"**, CNS-1116904, $450,000.

2010–12 UCSD Center for Networked Systems (co-PI; Tara Javidi PI), **"Congestion Aware Routing for Mesh Wireless Networks"**, $124,000.

2009–12 GENI Projects Office, **"Open Cirrus Federated Resource Allocation"**, $165,602.

2009–12 National Science Foundation (co-PI; Mohan Paturi PI, Stefan Savage, Mat McCubbins, Joel Sobel co-PIs), **"Network Structure, Incentives, and Outcomes"**, CNS-0905645, $900,000.

2009–12 National Science Foundation (co-PI; Ken Yocum PI), **"Topology Switching for Data Centers and the Clouds Above"**, CNS-0917339, $400,000.

2007–10 National Science Foundation (Stefan Savage, Amin Vahdat, Geoff Voelker co-PIs), **"Privacy Preserving Attribution & Provenance,"** CNS-0722031, $402,000.

2007–9 UCSD Center for Networked Systems (co-PI; Geoff Voelker PI, Stefan Savage, Amin Vahdat co-PIs), **"Extending Virtual Cluster Management and Resource Utilization,"** $130,000.

Plaid Ex1003
Page 137 of 372

2007–9 UCSD Center for Networked Systems (co-PI; Ken Yocum PI, Amin Vahdat co-PI), **"Seeding Cloud-Based Services,"** $130,000.

2007–9 UCSD Center for Networked Systems (co-PI; Stefan Savage PI, Geoff Voelker, Amin Vahdat co-PIs), **"System Support for Derived Data Management,"** $130,000.

2007 Cisco (George Varghese, Ramana Rao Kompella co-PIs), **"Designing Router Primitives to Monitor Network Health,"** $100,000.

2007 Google (Stefan Savage, Geoff Voelker co-PIs), **"Automated Cross-Layer Diagnosis of 802.11 Wireless Networks,"** $65,000.

2006–9 National Science Foundation (Ken Yocum co-PI), **"Distributed Rate Limiting,"** CNS-0627167, $360,000.

2006–9 National Science Foundation (Amin Vahdat, Geoff Voelker, Ken Yocum co-PIs), **"Harnessing Virtualized Cluster Resources,"** CNS-0615392, $450,000.

2006–8 UCSD Center for Networked Systems (co-PI; Bill Lin PI), **"Application of Coding to the Internet Backbone,"** $130,000.

2006–8 UCSD Center for Networked Systems (co-PI; Stefan Savage PI and Geoff Voelker co-PI), **"Automating Cross-layer Diagnosis of Enterprise Wireless Networks,"** $110,000.

2006–7 National Science Foundation (Stefan Savage, Amin Vahdat co-PIs), **"Enabling Defense and Deterrence through Private Attribution,"** CNS-0627157, $400,000.

2005–8 National Science Foundation (co-PI; Amin Vahdat PI and Joseph Pasquale co-PI), **"Algorithms and Infrastructure for Shared Mesh-based Broadcast,"** CNS-0520313, $500,000.

2005–7 UC Discovery (Amin Vahdat co-PI), **"Resource Allocation in Federated Distributed Computing Infrastructures,"** $130,821.

2005–7 UCSD Center for Networked Systems (co-PI; Geoff Voelker PI and Amin Vahdat co-PI), **"Network Vivisection: Cloning the Internet,"** $180,000.

2005–7 UCSD Center for Networked Systems (Stefan Savage, Amin Vahdat, Geoff Voelker co-PIs), **"Agile Resource Configuration and Scheduling with Virtual Clusters,"** $140,000.

2004–9 National Science Foundation, **"CAREER: Decoupling Policy from Mechanism in Internet Routing,"** CNS-0347949, $474,000.

2004–9 National Science Foundation (senior staff; Stefan Savage PI), Cybertrust Center Grant, **"Center for Internet Epidemiology and Defenses,"** NSF-0433668, $3,100,000.

2004–7 National Science Foundation (co-PI; Amin Vahdat PI), **"Framework for Designing, Evaluating, and Deploying Global Scale Adaptive Networked Systems,"** CNS-0411307, $345,636.

2004–6 UCSD Center for Networked Systems (co-PI Amin Vahdat), **"Resource Allocation in Federated Distributed Computing Infrastructures,"** $160,000.

2004–6 UCSD Center for Networked Systems (co-PI; Joseph Pasquale PI and Amin Vahdat co-PI), **"QoS-Based Application Middleware,"** $160,000.

2004–6 UCSD Center for Networked Systems (co-PI; Stefan Savage PI and Kimberly Claffy, Amin Vahdat, George Varghese, Geoff Voelker co-PIs), **"Internet Epidemiology,"** $240,000.

2004 National Science Foundation, **"Student Travel Support for ACM HotNets-III Workshop,"** CNS-0436331, $10,000.

Invited presentations

2017 "Big Data Begets Big Data: Understanding Modern Datacenter Networks", Keynote, SIGCOMM Big-DAMA Workshop, Los Angeles, California, August 2017.

2011 "Clue: Privacy-Preserving Network Foresnics", Information Security Seminar, Georgia Tech, Atlanta, Georgia, September 2011.

Plaid Ex1003
Page 138 of 372

2010 "BICMIC: Matching Cluster Configurations to Application Demands", CloudS Workshop, Sydney, Australia, November 2010.

"Practical 802.11: Addressing the Challenges of Diverse Environments", CSIRO, Brisbane, Australia, November 2010.

"California Fault Lines: Understanding the Causes and Impacts of Network Failures", Microsoft Research, Seattle, Washington, September 2010.

"Resource Management in the Cloud", First Mysore-Park Workshop on Building and Programming the Cloud, Infosys, Mysore, India, January 2010.

2008 "Traffic Policing in the Large: Distributed Rate Limiting," Keynote, EuroFGI Workshop on Quality of Service and Traffic Control, Lisbon, Portugal, December 2008.

"Router Primitives for Fault & Performance Monitoring," Nerd Lunch, Cisco Systems, San Jose, California, November 2008.

"Shaman: Diagnosing Performance Problems in Enterprise 802.11 Networks," UCSD Jacobs School Bay-Area Alumni Meeting, Cisco Systems, San Jose, California, June 2008.

"Shaman: Automatic Cross-Layer Diagnosis of Enterprise 802.11 Wireless Networks," Computer Science Colloquium, Harvey Mudd College, Claremont, California, March 2008.

2007 "Using Jigsaw to Automatically Diagnose Cross-Layer Faults in Enterprise Wireless Networks," Workshop on Methods and Tools for Network Analysis, Paris, France, July 2007.

"Reconsidering the Requirements for Congestion Control," NeXtworking '07, the Second COST-NSF Workshop on Future Internet, Berlin, Germany, April 2007.

2004 "Authenticated Policy-Compliant Routing," Ericsson, Stockholm, Sweden, September 2004.

2003 "Exploiting the Power of Single-Packet IP Traceback," DARPA Workshop on Control-Theoretic Approaches for Dynamic Information Assurance, Berkeley, California, February 2003.

2002 "Single-Packet IP Traceback," IEEE Computer Communications Workshop (CCW), Sante Fe, New Mexico, October 2002.

2001 "Mesh-Based Content Routing using XML," PEO Interchange XML Initiative (PIXIT) Meeting, The MITRE Corporation, Bedford, Massachusetts, October 2001.

"Hash-Based IP Traceback," Systems Seminar, College of Computing, Georgia Institute of Technology, Atlanta, October 2001.

2000 "TCP Connection Migration," End-to-End Research Group, Cambridge, Massachusetts, June 2000.

Advisees

PHD Ben Du (w/kc claffy, 2025), PostDoc, University of California Los Angeles
Rajdeep Das (2024), Microsoft
Anil Yelam (2024), Google
Stewart Grant (2024), Roblox
Alex Gamero Garrido (w/Alberto Dainotti, 2021), Assistant Professor, University of California Davis
Joe DeBlasio (2018), Google
Arjun Roy (w/Ken Yocum 2018), Google
Danny Yuxing Huang (w/Kirill Levchenko, 2017), Assistant Professor, New York University
Karyn Benson (w/kc claffy, 2016), Meta
Bhanu Vattikonda (2015), Google
Feng Lu (w/Tara Javidi and Geoffrey Voelker, 2014), Databricks
Daniel Turner (w/Stefan Savage, 2013), Shopify
Kevin Webb (w/Ken Yocum, 2013), Associate Professor, Swarthmore College

Plaid Ex1003
Page 139 of 372

John McCullough (2012), Google
Alvin AuYoung (2010), Aurora
Patrick Verkaik (2010), Cisco Meraki
Barath Raghavan (2009), Associate Professor, University of Southern California
Mikhail Afanasyev (2009), Toyota Research Institute
Jeannie Albrecht (w/Amin Vahdat, 2007), Professor and Chair, Williams College
Ramana Rao Kompella (w/George Varghese, 2007), Director of Research, Cisco
Liam Arzola (w/Amy Ousterhout), Ryan Kosta (w/Amy Ousterhout), and Alisha Ukani

MS Sen Zhang (2015), Google
Shaan Mahbubani (2008), HKUST MBA program
Jay Chen (2006), Assistant Professor, NYU Abu Dhabi
Chris Tuttle (w/Stefan Savage, 2005), Google
Erik VandeKieft (2004), IBM
Jon M. Salz (MIT 2002), Streambase

Courses taught

CSE 224 Graduate Networked Systems
Spring '24

CSE 223B Distributing Computing and Systems
Spring '25, '22, '21, '20, '18, '17, '15, '14, '13, '11, '10, '09, '06, '05, '04, '03; Winter '08

CSE 222A Computer Communication Networks
Winter '19, '18, '15, '14, '13

CSE 221 Graduate Operating Systems
Fall '16, '08, '07; Winter '16, '11

CSE 123 Computer Networks
Fall '24, '23, '19, '18, '17, '15, '14, '13, '10; Winter '22, '21, '17; Spring '16

CSE 121 Operating Systems: Architecture and Implementation
Winter '10, '09, '07, '06, '05; Fall '03

CSE 120 Principles of Computer Operating Systems
Spring '08; Fall '06, '05

CSE 8B Introduction to Computer Science: Java (II)
Spring '19

CSE 291 Topics in Wireless Networking
Spring '07; Winter '03

6.033 Computer Systems Engineering (*Massachusetts Institute of Technology*)
Spring '01

Professional activities

2025– DARPA Information Sciences and Technology (ISAT) Study Group
2018–21 Technical steering committee, ACM SIGCOMM Conference (Chair 2019–20)
2011–18 Steering committee, USENIX Symp. on Networked Systems Design & Implementation (NSDI)
2014 Guest Editor, IEEE Micro
2012–13 IDA/DARPA Defense Science Study Group (DSSG)
2008–12 Area editor, IEEE Transactions on Mobile Computing (TMC)

Plaid Ex1003
Page 140 of 372

2007–8 Planning committee, NSF Future Internet Design (FIND) Initiative
2004–8 Steering committee, ACM Workshop on Hot Topics in Networks (HotNets)
2008 Workshop & Tutorial co-chair, ACM SIGCOMM Conference
2006 Shadow PC co-coordinator, ACM SIGCOMM Conference
2004 General chair, ACM Workshop on Hot Topics in Networks (HotNets-III)

Patents

2019 William Mellette, Alex C. Snoeren, George Porter, and George Papen, *Optimizing Connectivity in Reconfigurable Networks*. US Patent 11,632,330. Filed Junary 31, 2019; issued April 18, 2023.

2017 George Porter, Joseph Ford, Alex C. Snoeren, George Papen, and William Mellette, *Selector Switch*. US Patent 11,550,104. Filed September 19, 2017; issued January 10, 2023.

2011 Jacobus van der Merwe, Dan Pei, Thomas Scholl, Aman Shaikh, Alex C. Snoeren, and Patrick Verkaik, *Intelligent Computer Network Routing Using Logically Centralized, Physically Distributed Servers Distinct from Network Routers*. US Patent 8,166,195. Filed January 31, 2011; issued April 24, 2012.

2007 Jacobus van der Merwe, Dan Pei, Thomas Scholl, Aman Shaikh, Alex C. Snoeren, and Patrick Verkaik, *Intelligent Computer Network Routing Using Logically Centralized, Physically Distributed Servers Distinct from Network Routers*. US Patent 7,904,589. Filed May 19, 2007; issued March 8, 2011.

2006 Albert Greenberg, Ramana Rao Kompella, Alex C. Snoeren, and Jennifer Yates, *Method and Apparatus for Fault Localization in a Network*. US Patent 7,577,099. Filed April 6, 2006; issued August 18, 2009.

2002 Walter C. Milliken, Luis A. Sanchez, and Alex C. Snoeren, *Systems and Methods for Point of Ingress Traceback of a Network Attack*. US Patent 7,200,105. Filed January 11, 2002; issued April 3, 2007.

Program committees

2017 **Chair**, ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)
2016 **Chair**, ACM Workshop on Hot Topics in Networks (HotNets)
2012 **Chair**, ACM International Conference on Mobile Computing and Networking (MobiCom)
2012 **Chair**, ACM/USENIX Internet Measurement Conference (IMC)
2010 **Chair**, USENIX Symposium on Networked Systems Design & Implementation (NSDI)

2004, 6, 8, 9, 14, 16, 17, 19, 21, 23, 24, 25 ACM Conference on Applications, Technologies, Architectures, and Protocols for Computer Communication (SIGCOMM)
2020 ACM Conference on Computer and Communications Security (CCS)
2009 ACM Conference on Embedded Networked Sensor Systems (SenSys)
2015 ACM International Conference on Emerging Networking Experiments and Technologies (CoNEXT)
2010, 12, 15 ACM International Conference on Measurement and Modeling of Computer Systems (SIGMETRICS)
2003,7–10, 12–15, 18 ACM International Conference on Mobile Computing and Networking (MobiCom)
2010 ACM/USENIX International Conference on Mobile Systems, Applications, and Services (MobiSys)
2006, 8, 12, 14, 16, 18, 20 ACM/USENIX Internet Measurement Conference (IMC)
2005 IEEE International Conference on Computer Communications (INFOCOM)
2017–18 IEEE Symposium on Security and Privacy (OAKLAND)
2012 ISOC Network and Distributed System Security Symposium (NDSS)
2003, 11 USENIX Annual Technical Conference (ATC)
2006, 7, 10 11, 16, 18, 19 21, 23, 26 USENIX Symposium on Networked Systems Design & Implementation (NSDI)

Plaid Ex1003
Page 141 of 372

2022, 23, 26 USENIX Symposium on Operating Systems Design & Implementation (OSDI)
2015 ACM International Workshop on Mobile Computing Systems and Applications (HotMobile)
2011 ACM International Workshop on Mobility in the Evolving Internet Architecture (MobiArch)
2015 ACM SIGCOMM Symposium on SDN Research (SOSR)
2018 ACM SIGCOMM Workshop on Self-Driving Networks (SelfDN)
2004, 16, 24 ACM Workshop on Hot Topics in Networks (HotNets)
2009 ACM Workshop on Networking, Systems, and Applications on Mobile Handhelds (MobiHeld)
2008 IEEE Workshop on Internet Network Management (INM)
2013 USENIX Workshop on Hot Topics in Operating Systems (HotOS)
2011 USENIX Workshop on Hot Topics in Security (HotSec)
2006, 7 Workshop on the Economics of Networked Systems and Incentive-Based Computing (NetEcon+IBC)

University/Departmental service

2021–22, 23–25 UCSD Academic Senate Committee on Academic Personnel
2021–22 UCSD School of Computing, Information, and Data Science Task Force
2014–18 UCSD Academic Senate Committee on Academic Information Technology
2017–18 UCSD Infrastructure Services Governance Committee
2017–18 UCSD Cybersecurity Governance Committee

2019–21 **Faculty Equity Advisor**, UCSD Jacobs School of Engineering
2019–20 Chair, UCSD Jacobs School of Engineering Faculty Recruiting Excellence Committee
2015–16 UCSD Jacobs School of Enginering Faculty Recruiting Diversity Committee
2004 UCSD Jacobs School of Engineering Research Review Planning Committee

2019–21, 25– **Vice Chair**, UCSD Department of Computer Science (Academic Personnel)
2018–19 Chair, UCSD Department of Computer Science Teaching Faculty Recruiting Committee
2013–19 Co-chair, UCSD Department of Computer Science Bylaws Committee
2013–15, 16 Co-chair, UCSD Department of Computer Science Admissions Committee
2011 Chair, UCSD Department of Computer Science Publicity Committee
2009, 17, 24 UCSD Department of Computer Science Faculty Recruiting Committee
2003, 4, 8 UCSD Department of Computer Science Graduate Student Affairs Committee
2004–7 UCSD Department of Computer Science Admissions Committee (Visit Day chair 2005–6)
2006 UCSD Department of Computer Science Computing Committee
2005 UCSD Department of Computer Science Building and Space Committee

Expert witness/Legal consulting experience

2021–23 Gibson, Dunn & Crutcher LLP/Cooley LLP, *Certain Fitness Devices, Streaming Components thereof, and Systems Containing Same*
No. 337-TA-1265, US International Trade Commission, filed April 13, 2021
Expert witness on non-infringement. Deposed; testified at hearing.

2020– Latham & Watkins LLP, *Intellectual Ventures I & Intellectual Ventures II LLC v. Arista Networks, Inc.*
No. 6:20-cv-749, Western District of Texas, filed August 18, 2020
Expert consultant for the defense.

2020–21 Osler, Hoskin & Harcourt LLP, *Route1 Inc. v. VMWare, Inc., AirWatch LLC, and VMware Canada Inc.*
No. T-2149-18, Canadian Federal Court
Expert witness for the defense. Case settled.

Plaid Ex1003
Page 142 of 372

2020 Winston & Strawn LLP, *Intellectual Ventures II LLC v. VMware, Inc.*
No. 6:20-cv-457, Western District of Texas
Expert consultant for the defense. Case settled.

2019–20 Winston & Strawn LLP, *Intellectual Ventures I LLC & Intellectual Ventures II LLC v. VMware, Inc.*
No. 6:19-cv-449, Western District of Texas, filed August 23, 2019
Expert witness on invalidity and non-infringement. Case settled.

2019–20 King & Spalding LLP, *Sound View Innovations, LLC v. Delta Air Lines, Inc.*
No. 1:19-cv-0065, Delware District Court, filed April 9, 2019
Expert consultant for the defense. Case settled.

2019–22 Morrison & Foerster LLP, *Advanced Voice Recognition Systems, Inc. v. Apple Inc.*
No. 2:18-cv-02083-DGC, Arizona District Court
Expert consultant for the defense. Case dismissed.

2018–21 Nixon Peabody LLP/Gibson, Dunn & Crutcher LLP. *Citrix Systems, Inc. v. Workspot Inc.*
No. 1:2018-cv-00588, Delaware District Court, filed April 19, 2018
Expert witness on invalidity and non-infringement. Deposed; case settled.

2018–19 Wilmer Cutler Pickering Hale and Dorr LLP, *Route1 Inc. v. Airwatch LLC*
No. 1:2017-cv-00331, Delaware District Court, filed March 27, 2017
Expert witness on invalidity and non-infringement. Deposed; granted summary judgement on non-infringement.

2018 Tensegrity Law Group LLP, *Arista Networks, Inc. v. Cisco Systems, Inc.*
No. 1:2017-mc-00009, Indiana Southern District Court, filed January 31, 2017
Expert consultant for the plaintiff. Case settled.

2017–18 Norton Rose Fulbright LLP, *Ameranth, Inc. v. Pizza Hut, Inc. et al.*
No. 3:2011-cv-01810, California Southern District Court, filed August 15, 2011
Expert witness on invalidity. Case stayed due to judgement of unpatentability in related matter.

2017–18 Latham & Watkins LLP. *Certain Network Devices, Related Software and Components Thereof (II)*
No. 337-TA-945, U.S. International Trade Commission, filed December 19, 2014
Expert witness on non-infringement. Deposed; testified at modification proceeding hearing.

2017 Wilmer Cutler Pickering Hale and Dorr LLP, *Airwatch LLC and VMWare Inc. v. Route1 Inc.*
IPR 2017-02145, PTAB, filed September 22, 2017
Expert on invalidity in IPR proceedings.

2017 Gibson, Dunn & Crutcher LLP, *University of Tennessee Research Foundation, et al. v. Amazon, Inc. et al.*
No. 3:2017-cv-00181, Tennessee Eastern District Court, filed May 2, 2017
Expert witness on non-infringement. Case settled.

2017 Norton Rose Fulbright LLP, *Clean Energy Management Solutions, LLC v. AT&T Digital Life, Inc.*
No. 2:2016-cv-00413, Texas Eastern District Court, filed April 15, 2015
Expert witness on non-infrignement and invalidity. Case settled.

2016–17 Gibson, Dunn & Crutcher LLP, *Yodlee v. Plaid Technologies*
No. 1:2014-cv-01445, Delaware District Court, filed December 1, 2014
Expert witness on non-infringement. Deposed; case settled.

2014–15 Irell & Manella LLP. *ReefEdge Networks, LLC v. Aruba Networks et al.*
No. 1:2012-cv-01042, Delaware District Court, filed August 15, 2012
Expert witness on non-infringement. Deposed; case settled.

2014–15 Morrison & Foerster LLP. *VMware Inc. v. Good Technology Software, Inc.*
IPR 2014-1324, PTAB, filed August 18, 2014; 2015-00027,00030,00031, PTAB, filed October 6, 2014
Expert on invalidity in IPR proceedings. Case settled.

2013–14 Gibson, Dunn & Crutcher LLP/Morrison & Foerster LLP. *Apple v. Samsung Electronics Co., Ltd. et al.*
No. 5:2012-cv-00630, California Northern District Court, filed February 8, 2012
Expert witness on infringement and validity. Deposed; testified at trial.

2012–13 Simspon Thatcher & Bartlett LLP. *CyberFone Systems, LLC v. Cellco Partnership, et al.*
No. 1:2011-cv-00827-SLR, District of Delaware, filed September 15, 2011
Expert witness for the defense. Case settled.

2012 Morrison & Foerster LLP. *Apple v. Samsung Electronics Co., Ltd. et al.*
No. 5:2011-cv-01846, California Northern District Court, filed April 15, 2011
Expert witness on infringement and validity. Deposed; patent removed from litigation.

2011–12 Weil, Gotshal & Manges LLP. *Apple & NEXT Software v. Motorola & Motorola Mobility*
No. 1:2011-cv-08540, Illinois Northern District Court, filed December 1, 2011
Expert witness on infringement and validity. Deposed; case dismissed.

2010–13 Sidley Austin LLP. *Nomadix v. Hewlett-Packard Company, et al.*
No. 2:2009-cv-08441, California Central District Court, filed November 17, 2009
Expert witness on non-infringement. Case settled.

2010 Fulbright & Jaworski, LLP, *Ameranth, Inc. v. Menusoft*
No. 1:2010-cv-01419, Colorado District Court, filed June 16, 2010
Technical consultant for the defense.

2009 Coast Law Group, LLP. *eBay Inc. v. Digital Point Solutions, Inc., et al.*
No. 08-4052 JF (PVT), Northern District of California, filed February 24, 2009
Technical consultant for the defense. Case settled.

2008 Proskauer Rose LLP. *C2 Communications v. AT&T et al.*
No. 3:2007-cv-04457, New Jersey District Court, filed September 17, 2007
Technical consultant for the defense. Case settled.

2006–8 Weil, Gotshal & Manges LLP. *Visto v. Microsoft*
No. 2:2005-cv-00546, Texas Eastern District Court, filed December 14, 2005
Expert witness and technical consultant for the defense. Deposed; case settled.

2007 Coast Law Group, LLP. *Universal City Studios Productions LLLP v. Shawn Hogan*
No. 3:06-cv-00545-W-WMC, Southern District of California, filed March 14, 2006
Expert witness for the defense. Case settled.

2005 Post Kirby Noonan & Sweat LLP. *Stratagene v. Ariadne Genomics*
No. 3:2005-cv-01321, California Southern District Court, filed June 29, 2005
Technical consultant for claimant in arbitration. Parties settled.

2004–6 Musick, Peeler & Garrett LLP, and White & Oliver APC. *Alabanza Corporation v. ABACUS America*
No. 1:2002-cv-01946, Maryland District Court, filed June 10, 2002
Expert witness for the plaintiff. Case settled.

Plaid Ex1003
Page 144 of 372

EXHIBIT B

Preface

For generations, people have defined and protected their property and their privacy using locks, fences, signatures, seals, account books, and meters. These have been supported by a host of social constructs ranging from international treaties through national laws to manners and customs.

This is changing, and quickly. Most records are now electronic, from bank accounts to registers of real property; and transactions are increasingly electronic, as shopping moves to the Internet. Just as important, but less obvious, are the many everyday systems that have been quietly automated. Burglar alarms no longer wake up the neighborhood, but send silent messages to the police; students no longer fill their dormitory washers and dryers with coins, but credit them using a smartcard they recharge at the college bookstore; locks are no longer simple mechanical affairs, but are operated by electronic remote controls or swipe cards; and instead of renting videocassettes, millions of people get their movies from satellite or cable channels. Even the humble banknote is no longer just ink on paper, but may contain digital watermarks that enable many forgeries to be detected by machine.

How good is all this new security technology? Unfortunately, the honest answer is "nowhere near as good as it should be." New systems are often rapidly broken, and the same elementary mistakes are repeated in one application after another. It often takes four or five attempts to get a security design right, and that is far too many.

The media regularly report security breaches on the Internet; banks fight their customers over "phantom withdrawals" from cash machines; VISA reports huge increases in the number of disputed Internet credit card transactions; satellite TV companies hound pirates who copy their smartcards; and law enforcement agencies try to stake out territory in cyberspace with laws controlling the use of encryption. Worse still, features interact. A mobile phone that calls the last number again if one of the keys is pressed by accident may be just a minor nuisance—until someone invents a machine that dispenses a can of soft drink every time its phone number is called. When all of a sudden you find 50 cans of Coke on your phone bill, who is responsible, the phone company, the handset manufacturer, or the vending machine operator? Once almost every electronic device that affects your life is connected to the Internet—which Microsoft expects to happen by 2010—what does 'Internet security' mean to you, and how do you cope with it?

As well as the systems that fail, many systems just don't work well enough. Medical record systems don't let doctors share personal health information as they would like, but still don't protect it against inquisitive private eyes. Zillion-dollar military systems prevent anyone without a "top secret" clearance from getting at intelligence data, but are often designed so that almost everyone needs this clearance to do any work. Passenger ticket systems are designed to prevent customers cheating, but when trustbusters break up the railroad, they cannot stop the new rail companies cheating each other.

Plaid Ex1003
Page 146 of 372

Many of these failures could have been foreseen if designers had just a little bit more knowledge of what had been tried, and had failed, elsewhere.

Security engineering is the new discipline that is starting to emerge out of all this chaos.

Although most of the underlying technologies (cryptology, software reliability, tamper resistance, security printing, auditing, etc.) are relatively well understood, the knowledge and experience of how to apply them effectively is much scarcer. And since the move from mechanical to digital mechanisms is happening everywhere at once, there just has not been time for the lessons learned to percolate through the engineering community. Time and again, we see the same old square wheels being reinvented.

The industries that have managed the transition most capably are often those that have been able to borrow an appropriate technology from another discipline. Examples include the reuse of technology designed for military identify-friend-or-foe equipment in bank cash machines and even prepayment gas meters. So even if a security designer has serious expertise in some particular speciality—whether as a mathematician working with ciphers or a chemist developing banknote inks—it is still prudent to have an overview of the whole subject. The essence of good security engineering is understanding the potential threats to a system, then applying an appropriate mix of protective measures—both technological and organizational—to control them. Knowing what has worked, and more importantly what has failed, in other applications is a great help in developing judgment. It can also save a lot of money.

The purpose of this book is to give a solid introduction to security engineering, as we understand it at the beginning of the twenty-first century. My goal is that it works at four different levels:

- *As a textbook that you can read from one end to the other over a few days as an introduction to the subject.* The book is to be used mainly by the working IT professional who needs to learn about the subject, but it can also be used in a one-semester course in a university.
- *As a reference book to which you can come for an overview of the workings of some particular type of system.* These systems include cash machines, taxi meters, radar jammers, anonymous medical record databases, and so on.
- *As an introduction to the underlying technologies, such as crypto, access control, inference control, tamper resistance, and seals.* Space prevents me from going into great depth; but I provide a basic road map for each subject, plus a reading list for the curious (and a list of open research problems for the prospective graduate student).
- *As an original scientific contribution in which I have tried to draw out the common principles that underlie security engineering, and the lessons that people building one kind of system should have learned from others.* In the many years I have been working in security, I keep coming across these. For example, a simple attack on stream ciphers wasn't known to the people who designed a common antiaircraft fire control radar so it was easy to jam; while a trick well known to the radar community wasn't understood by banknote printers and people who design copyright marking schemes, which led to a quite general attack on most digital watermarks.

Plaid Ex1003
Page 147 of 372

I have tried to keep this book resolutely mid-Atlantic; a security engineering book has to be, as many of the fundamental technologies are American, while many of the interesting applications are European. (This isn't surprising given the better funding of U.S. universities and research labs, and the greater diversity of nations and markets in Europe.) What's more, many of the successful European innovations—from the smartcard to the GSM mobile phone to the pay-per-view TV service—have crossed the Atlantic and now thrive in the Americas. Both the science, and the case studies, are necessary.

This book grew out of the security engineering courses I teach at Cambridge University, but I have rewritten my notes to make them self-contained and added at least as much material again. It should be useful to the established professional security manager or consultant as a first-line reference; to the computer science professor doing research in cryptology; to the working police detective trying to figure out the latest computer scam; and to policy wonks struggling with the conflicts involved in regulating cryptography and anonymity. Above all, it is aimed at Dilbert. My main audience is the working programmer or engineer who is trying to design real systems that will keep on working despite the best efforts of customers, managers, and everybody else.

This book is divided into three parts.

- The first looks at basic concepts, starting with the central concept of a security protocol, and going on to human-computer interface issues, access controls, cryptology, and distributed system issues. It does not assume any particular technical background other than basic computer literacy. It is based on an Introduction to Security course that I teach to second-year undergraduates.
- The second part looks in much more detail at a number of important applications, such as military communications, medical record systems, cash machines, mobile phones, and pay-TV. These are used to introduce more of the advanced technologies and concepts. It also considers information security from the viewpoint of a number of different interest groups, such as companies, consumers, criminals, police, and spies. This material is drawn from my senior course on security, from research work, and from experience consulting.
- The third part looks at the organizational and policy issues: how computer security interacts with law, with evidence, and with corporate politics; how we can gain confidence that a system will perform as intended; and how the whole business of security engineering can best be managed.

I believe that building systems that continue to perform robustly in the face of malice is one of the most important, interesting, and difficult tasks facing engineers in the twenty-first century.

Ross Anderson
Cambridge, January 2001

Plaid Ex1003
Page 148 of 372

About the Author

Why should I have been the person to write this book? Well, I seem to have accumulated the right mix of experience and qualifications over the last 25 years. I graduated in mathematics and natural science from Cambridge (England) in the 1970s, and got a qualification in computer engineering; my first proper job was in avionics; and I became interested in cryptology and computer security in the mid-1980s. After working in the banking industry for several years, I started doing consultancy for companies that designed equipment for banks, and then working on other applications of this technology, such as prepayment electricity meters.

I moved to academia in 1992, but continued to consult to industry on security technology. During the 1990s, the number of applications that employed cryptology rose rapidly: burglar alarms, car door locks, road toll tags, and satellite TV encryption systems all made their appearance. As the first legal disputes about these systems came along, I was lucky enough to be an expert witness in some of the important cases. The research team I lead had the good fortune to be in the right place at the right time when several crucial technologies, such as tamper resistance and digital watermarking, became hot topics.

By about 1996, it started to become clear to me that the existing textbooks were too specialized. The security textbooks focused on the access control mechanisms in operating systems, while the cryptology books gave very detailed expositions of the design of cryptographic algorithms and protocols. These topics are interesting, and important. However they are only part of the story. Most system designers are not overly concerned with crypto or operating system internals, but with how to use these tools effectively. They are quite right in this, as the inappropriate use of mechanisms is one of the main causes of security failure. I was encouraged by the success of a number of articles I wrote on security engineering (starting with "Why Cryptosystems Fail" in 1993); and the need to teach an undergraduate class in security led to the development of a set of lecture notes that made up about half of this book. Finally, in 1999, I got round to rewriting them for a general technical audience.

I have learned a lot in the process; writing down what you think you know is a good way of finding out what you don't. I have also had a lot of fun. I hope you have as much fun reading it!

Plaid Ex1003
Page 149 of 372

Foreword

In a paper he wrote with Roger Needham, Ross Anderson coined the phrase "programming Satan's computer" to describe the problems faced by computer-security engineers. It's the sort of evocative image I've come to expect from Ross, and a phrase I've used ever since.

Programming a computer is straightforward: keep hammering away at the problem until the computer does what it's supposed to do. Large application programs and operating systems are a lot more complicated, but the methodology is basically the same. Writing a reliable computer program is much harder, because the program needs to work even in the face of random errors and mistakes: Murphy's computer, if you will. Significant research has gone into reliable software design, and there are many mission-critical software applications that are designed to withstand Murphy's Law.

Writing a *secure* computer program is another matter entirely. Security involves making sure things work, not in the presence of random faults, but in the face of an intelligent and malicious adversary trying to ensure that things fail in the worst possible way at the worst possible time ... again and again. It truly is programming Satan's computer.

Security engineering is different from any other kind of programming. It's a point I made over and over again: in my own book, *Secrets and Lies*, in my monthly newsletter *Crypto-Gram*, and in my other writings. And it's a point Ross makes in every chapter of this book. This is why, if you're doing any security engineering ... if you're even *thinking* of doing any security engineering, you need to read this book. It's the first, and only, end-to-end modern security design and engineering book ever written.

And it comes just in time. You can divide the history of the Internet into three waves. The first wave centered around mainframes and terminals. Computers were expensive and rare. The second wave, from about 1992 until now, centered around personal computers, browsers, and large application programs. And the third, starting now, will see the connection of all sorts of devices that are currently in proprietary networks, standalone, and non-computerized. By 2003, there will be more mobile phones connected to the Internet than computers. Within a few years we'll see many of the world's refrigerators, heart monitors, bus and train ticket dispensers, burglar alarms, and electricity meters talking IP. Personal computers will be a minority player on the Internet.

Security engineering, especially in this third wave, requires you to think differently. You need to figure out not how something works, but how something can be made to not work. You have to imagine an intelligent and malicious adversary inside your system (remember Satan's computer), constantly trying new ways to subvert it. You have to consider all the ways your system can fail, most of them having nothing to do with the design itself. You have to look at everything backwards, upside down, and sideways. You have to think like an alien.

As the late great science fiction editor John W. Campbell, said: "An alien thinks as well as a human, but not like a human." Computer security is a lot like that. Ross is

Plaid Ex1003
Page 150 of 372

one of those rare people who can think like an alien, and then explain that thinking to humans. Have fun reading.

Bruce Schneier
January 2001

Plaid Ex1003
Page 151 of 372

Acknowledgments

A great many people have helped in various ways with this book. I probably owe the greatest thanks to those who read the manuscript (or a large part of it) looking for errors and obscurities. They were Anne Anderson, Ian Brown, Nick Bohm, Richard Bondi, Caspar Bowden, Richard Clayton, Steve Early, Rich Graveman, Markus Kuhn, Dan Lough, David MacKay, John McHugh, Bob Morris, Roger Needham, Jerry Saltzer, Marv Schaefer, Karen Spärck Jones and Frank Stajano. Much credit also goes to my editor, Carol Long, who (among many other things) went through the first six chapters and coached me on the style appropriate for a professional (as opposed to academic) book. At the proofreading stage, I got quite invaluable help from Carola Bohm, Mike Bond, Richard Clayton, George Danezis, and Bruce Godfrey.

A large number of subject experts also helped me with particular chapters or sections. Richard Bondi helped me refine the definitions in Chapter 1; Jianxin Yan, Alan Blackwell and Alasdair Grant helped me investigate the applied psychology aspects of passwords; John Gordon and Sergei Skorobogatov were my main sources on remote key entry devices; Whit Diffie and Mike Brown on IFF; Steve Early on Unix security (although some of my material is based on lectures given by Ian Jackson); Mike Roe, Ian Kelly, Paul Leyland, and Fabien Petitcolas on the security of Windows NT4 and Win2K; Virgil Gligor on the history of memory overwriting attacks, and on mandatory integrity policies; and Jean Bacon on distributed systems. Gary Graunke told me the history of protection in Intel processors; Orr Dunkelman found many bugs in a draft of the crypto chapter and John Brazier pointed me to the Humpty Dumpty quote.

Moving to the second part of the book, the chapter on multilevel security was much improved by input from Jeremy Epstein, Virgil Gligor, Jong-Hyeon Lee, Ira Moskowitz, Paul Karger, Rick Smith, Frank Stajano, and Simon Wiseman, while Frank also helped with the following two chapters. The material on medical systems was originally developed with a number of people at the British Medical Association, most notably Fleur Fisher, Simon Jenkins, and Grant Kelly. Denise Schmandt-Besserat taught the world about bullae, which provided the background for the chapter on banking systems; that chapter was also strengthened by input from Fay Hider and Willie List. The chapter on alarms contains much that I was taught by Roger Needham, Peter Dean, John Martin, Frank Clish, and Gary Geldart. Nuclear command and control systems are much the brainchild of Gus Simmons; he and Bob Morris taught me much of what's in that chapter.

Sijbrand Spannenburg reviewed the chapter on security printing; and Roger Johnston has taught us all an enormous amount about seals. John Daugman helped polish the chapter on biometrics, as well as inventing iris scanning which I describe there. My tutors on tamper resistance were Oliver Kömmerling and Markus Kuhn; Markus also worked with me on emission security. I had substantial input on electronic warfare from Mike Brown and Owen Lewis. The chapter on phone fraud owes a lot to Duncan Campbell, Richard Cox, Rich Graveman, Udi Manber, Andrew Odlyzko and Roy Paterson. Ian Jackson contributed some ideas on network security. Fabien Petitcolas

Plaid Ex1003
Page 152 of 372

'wrote the book' on copyright marking, and helped polish my chapter on it. Johann Bezuidenhoudt made perceptive comments on both phone fraud and electronic commerce, while Peter Landrock gave valuable input on bookkeeping and electronic commerce systems. Alistair Kelman was a fount of knowledge on the legal aspects of copyright; and Hal Varian kept me straight on matters of economics, and particularly the chapters on e-commerce and assurance.

As for the third part of the book, the chapter on e-policy was heavily influenced by colleagues at the Foundation for Information Policy Research, notably Caspar Bowden, Nick Bohm, Fleur Fisher, Brian Gladman, Ian Brown, Richard Clayton—and by the many others involved in the fight, including Whit Diffie, John Gilmore, Susan Landau, Brian Omotani and Mark Rotenberg. The chapter on management benefited from input from Robert Brady, Jack Lang, and Willie List. Finally, my thinking on assurance has been influenced by many people, including Robin Ball, Robert Brady, Willie List, and Robert Morris.

There were also many people over the years who taught me my trade. The foremost of them is Roger Needham, who was my thesis advisor; but I also learned a lot from hundreds of engineers, programmers, auditors, lawyers, and policemen with whom I worked on various consultancy jobs over the last 15 years. Of course, I take the rap for all the remaining errors and omissions.

Finally, I owe a huge debt to my family, especially to my wife Shireen for putting up with over a year in which I neglected household duties and was generally preoccupied. Daughter Bavani and dogs Jimmy, Bess, Belle, Hobbes, Bigfoot, Cat, and Dogmatix also had to compete for a diminished quantum of attention, and I thank them for their forbearance.

Legal Notice

I cannot emphasize too strongly that the tricks taught in this book are intended only to enable you to build better systems. They are not in any way given as a means of helping you to break into systems, subvert copyright protection mechanisms, or do anything else unethical or illegal.

Where possible I have tried to give case histories at a level of detail that illustrates the underlying principles without giving a "hacker's cookbook."

Should This Book Be Published at All?

There are people who believe that the knowledge contained in this book should not be published. This is an old debate; in previous centuries, people objected to the publication of books on locksmithing, on the grounds that they were likely to help the bad guys more than the good guys.

I think that these fears are answered in the first book in English that discussed cryptology. This was a treatise on optical and acoustic telegraphy written by Bishop John Wilkins in 1641 [805]. He traced scientific censorship back to the Egyptian priests who forbade the use of alphabetic writing on the grounds that it would spread literacy among the common people and thus foster dissent. As he said:

> *It will not follow that everything must be suppresst which may be abused... If all those useful inventions that are liable to abuse should therefore be concealed there is not any Art or Science which may be lawfully profest.*

The question was raised again in the nineteenth century, when some well-meaning people wanted to ban books on locksmithing. A contemporary writer on the subject replied [750]:

> *Many well-meaning persons suppose that the discussion respecting the means for baffling the supposed safety of locks offers a premium for dishonesty, by showing others how to be dishonest. This is a fallacy. Rogues are very keen in their profession, and already know much more than we can teach them respecting their several kinds of roguery. Rogues knew a good deal about lockpicking long before locksmiths discussed it among themselves ... if there be harm, it will be much more than counterbalanced by good.*

These views have been borne out by long experience since. As for me, I worked for two separate banks for three and a half years on cash machine security, but I learned significant new tricks from a document written by a convicted card fraudster that circulated in the U.K. prison system. Many government agencies are now coming round to this point of view. It is encouraging to see, for example, that the U.S. National Security Agency has published the specifications of the encryption algorithm (Skipjack) and the key management protocol (KEA) used to protect secret U.S. government traffic.

Plaid Ex1003
Page 154 of 372

Their judgment is clearly that the potential harm done by letting the Iraqis use a decent encryption algorithm is less than the good that will be done by having commercial off-the-shelf software compatible with Federal encryption standards.

In short, while some bad guys will benefit from a book such as this, they mostly know the tricks already, and the good guys will benefit much more.

Plaid Ex1003
Page 155 of 372

CHAPTER

19

Protecting E-Commerce Systems

If you try to buck the markets, then the markets will buck you.

—MARGARET THATCHER

19.1 Introduction

The protection of electronic commerce systems pulls together a lot of the topics discussed in previous chapters. Failures come from misconfigured access control, implementation blunders, theft of network services, inappropriate use of crypotology—you name it. In this chapter, I'll cover some protection issues specific to e-commerce, such as how online credit card payments are handled, and what goes wrong.

If you are a programmer building e-commerce systems for a dot-com startup, much of the material in this chapter should be fairly familiar to you. You are much more likely to get value from the chapters on access control, network security, and (particularly) on banking. The most likely attacks on your business don't involve the vulnerabilities in the Internet protocol suite or the payment infrastructure—and you can't do anything about those anyway.

The typical e-business startup appears to be most at risk from internal fraud. This is where most frauds in normal businesses come from, despite things like double-entry bookkeeping, which have evolved over centuries to control them. Many startups have none of these internal controls. They begin as a few people who all know each other and, if successful at raising capital, rapidly hire a lot of new staff who're focused on money and not very carefully screened. A survey found in October 2000, for example, that 37% of dot-com executives have shady pasts, compared with 10% found in due diligence checks on normal companies [257].

Plaid Ex1003
Page 156 of 372

19.2 A Telegraphic History of E-Commerce

Many of the problems afflicting e-businesses stem from the popular notion that e-commerce is something completely new, invented in the mid-1990s. This is simply untrue.

Various kinds of visual signalling were deployed from classical times. Systems included heliographs (which used mirrors to flash sunlight at the receiver), semaphones (which used the positions of moving arms to signal letters and numbers), and flags. Land-based systems sent messages along chains of beacon towers, and naval systems between ships. To begin with, their use was military, but after the Napoleonic War, the French government opened its heliograph network to commercial use. Very soon, the first frauds were carried out. For two years, until they were discovered in 1836, two bankers bribed an operator to signal the movements of the stock market to them covertly by making errors in transmissions, which they could observe from a safe distance. Other techniques were devised to signal the results of horseraces. Various laws were passed to criminalize this kind of activity, but they were ineffective. The only solution for the bookies was to "call time" by a clock, rather than waiting for the result and hoping that they were the first to hear it.

From the 1760s to the 1840s, the electric telegraph was developed by a number of pioneers, of whom the most influential was Samuel Morse. He persuaded Congress in 1842 to fund an experimental line from Washington to Baltimore; this so impressed people that serious commercial investment started, and by the end of that decade, there were 12,000 miles of line being operated by 20 companies. This was remarkably like the Internet boom of the late 1990s [729].

Banks were the first big users of the telegraph, and they decided that they needed technical protection mechanisms to prevent transactions being altered by crooked operators en route. (I discussed the *test key* systems they developed for the purpose in the chapter on banking systems.) Telegrams were also used to create national markets. For the first time, commodity traders in New York could find out within minutes the prices that had been set in auctions in Chicago; likewise, fishing skippers arriving in Boston could find out the price of cod in Gloucester. A recent history of the period shows that most of the concepts and problems of electronic commerce were familiar to the Victorians [729]. How do you know who you're speaking to? How do you know if they're trustworthy? How do you know whether the goods will be delivered, and whether payments will arrive? The answers found in the nineteenth century involved intermediaries—principally banks that helped businesses manage risk using instruments such as references, guarantees, and letters of credit.

In the 1960s, banks in many countries computerized their bookkeeping, and introduced national interbank systems for handling direct payments to customer accounts, enabling banks to offer services such as payroll to corporate customers. In the early 1970s, this was extended to international payments, as described in the banking systems chapter. The next large expansion of electronic commerce took place in the late 1970s to mid-1980s with the spread of *electronic data interchange* (EDI). Companies ranging from General Motors to Marks and Spencer built systems that enabled them to link up their computers to their suppliers', so that goods could be ordered automatically. Travel agents built similar systems to order tickets in real time from airlines.

Plaid Ex1003
Page 157 of 372

In 1985, the first retail electronic banking system was offered by the Bank of Scotland, whose customers could use Prestel, a proprietary email system operated by British Telecom, to make payments. When Steve Gold and Robert Schifreen hacked Prestel—as described in the chapter on passwords—it initially terrified the press and the bankers. They realized that the hackers could easily have captured and altered transactions. But once the dust settled and people thought through the detail, it became clear there was little real risk. The system allowed only payments between your own accounts and to accounts which you'd previously notified to the bank, such as your gas and electricity suppliers.

This pattern, of high-profile hacks—which caused great consternation but which, on sober reflection, turned out to be not really a big deal—has continued ever since.

To resume this brief history, the late 1980s and early 1990s saw the rapid growth of call centers, which—despite all the hoopla about the Web—remain in 2000 by far the largest delivery channel for business-to-consumer electronic commerce. As for the Internet, it was not something that suddenly sprung into existence in 1995, as a Martian monitoring our TV channels might believe. The first time I used an online service to sell software I'd written was in 1984 or 1985; and I first helped the police investigate an online credit card fraud in 1987. In the latter case, the bad guy got a list of hot credit card numbers from his girlfriend, who worked in a supermarket; he used them to buy software from companies in California, which he downloaded to order for his customers. This worked because hot card lists at the time carried only those cards that were being used fraudulently in that country; it also guaranteed that the bank would not be able to debit an innocent customer. As it happens, the criminal quit before there was enough evidence to nail him. A rainstorm washed away the riverbank opposite his house and exposed a hide which the police had built to stake him out.

The use of credit cards to buy stuff electronically "suddenly" became mainstream in about 1994 or 1995, when the public started to go online in large numbers. Suddenly there was a clamor that the Internet was insecure, that credit card numbers could be harvested on a huge scale, and that encryption would be needed.

19.3 An Introduction to Credit Cards

For many years after their invention in the 1950s, credit cards were treated by most banks as a loss leader used to attract high-value customers. Eventually, in most countries, the number of merchants and cardholders reached critical mass, and the transaction volume took off. In Britain, it took almost 20 years before most banks found the business profitable; then all of a sudden it became extremely profitable. The credit card system is now extremely well entrenched as the payment mechanism used on the Net.

Because of the huge investment involved in rolling out a competitor to tens of thousands of banks, millions of merchants, and billions of customers worldwide, any new payment mechanism is likely to take some time to get established—with a possible exception, which I'll discuss shortly. When you use a credit card to pay for a purchase

Plaid Ex1003
Page 158 of 372

in a store, the transaction flows from the merchant to her bank (the acquiring bank), which pays her after deducting a *merchant discount* of, typically, 4–5%. If the card was issued by a different bank, the transaction next flows to a switching center run by the brand (such as VISA), which takes a commission and passes it to the issuing bank for payment. Daily payments between the banks and the brands settle the net cash flows. The issuer also gets a slice of the merchant discount, but makes most of its money from extending credit to cardholders at rates usually much higher than the interbank rate.

19.3.1 Fraud

The risk of fraud from stolen cards was traditionally managed by a system of *hot card lists* and *merchant floor limits*. Each merchant gets a local hot card list—formerly on paper, now stored in her terminal—plus a limit set by their acquiring bank, above which they have to call for authorization. The call center, or online service, which she uses for this has access to a national hot card list; above a higher limit, they will contact the brand which has a complete list of all hot cards being used internationally; above a still higher limit, the transaction will be checked all the way back to the card issuer.

The introduction of *mail order and telephone order* (MOTO) transactions in the 1970s meant that the merchant did not have the customer present, and was not able to inspect the card. What was to stop a crook ordering goods using a credit card numbers he'd picked up from a discarded receipt?

Banks managed the risk by using the expiry date as a password, lowering the floor limits, increasing the merchant discount, and insisting on delivery to a cardholder address, which is supposed to be checked during authorization. But the main change was to shift liability so that the merchant bore the full risk of disputes. If you challenge an online credit card transaction (or in fact any transaction made under MOTO rules), the full amount is immediately debited back to the merchant, together with a significant handling fee. The same procedure applies whether the debit is a fraud, a dispute, or a return.

Of course, having the cardholder present doesn't guarantee that fraud will be rare. For many years, most fraud was done in person with stolen cards, and the stores that got badly hit tended to be those selling goods that can be easily fenced, such as jewelry and consumer electronics. Banks responded by lowering their floor limits. More recently, as technical protection mechanisms have improved, there has been an increase in scams involving cards that were never received by genuine customers. This *pre-issue fraud* can involve thefts from the mail of the many "pre-approved" cards that arrive in junk mail, or even applications made in the names of people who exist and are creditworthy, but are not aware of the application (*identity theft*). These attacks on the system are intrinsically hard to tackle using purely technical means.

19.3.2 Forgery

In the early 1980s, electronic terminals were introduced through which a sales clerk could swipe a card and get an authorization automatically. But the sales draft was still captured from the embossing, so crooks figured out how to re-encode the magnetic strip of a stolen card with the account number and expiry date of a valid card, which they often got by fishing out discarded receipts from the trash cans of expensive restaurants. A re-encoded card would authorize perfectly, but when the merchant submitted

Plaid Ex1003
Page 159 of 372

the draft for payment, the account number didn't match the authorization code (a six-digit number typically generated by encrypting the account number, date, and amount). The merchants didn't get paid, and raised hell.

Banks responded in the mid-1980s by introducing *terminal draft capture*, where a sales draft is printed automatically using the data on the card strip. The crooks' response was a flood of forged cards, many produced by Triad gangs: between 1989 and 1992, magnetic strip counterfeiting grew from an occasional nuisance into half the total fraud losses [6]. VISA's response was *card verification values* (CVVs), three-digit MACs computed on the card strip contents (account number, version number, expiry date) and written at the end of the strip. They worked well at first; in the first quarter of 1994, VISA International's fraud losses dropped by 15.5%, while Mastercard's rose 67% [165]. Subsequently, Mastercard adopted similar checksums, too.

The crooks' response was *skimming*—operating businesses where genuine customer cards were swiped through an extra, unauthorized, terminal to grab a copy of the magnetic strip, which would then be re-encoded on a genuine card. The banks' response was intrusion detection systems, which in the first instance tried to identify criminal businesses by correlating the previous purchase histories of customers who complained.

In the late 1990s, credit card fraud rose sharply due to another simple innovation in criminal technology: the crooked businesses that skim card data absorb the cost of the customer's transaction rather than billing it. You have a meal at a Mafia-owned restaurant, offer a card, sign the voucher, and fail to notice when the charge doesn't appear on your bill. Perhaps a year later, there is suddenly a huge bill for jewelry, electronic goods, or even casino chips. By then you've completely forgotten about the meal, and the bank never had a record of it [318].

19.3.3 Automatic Fraud Detection

Consequently, a lot of work was done in the 1990s on beefing up intrusion detection. There are a number of generic systems that do anomaly detection, using techniques such as neural networks, but it's unclear how effective they are. When fraud is down one year, it's hailed as a success for the latest fraud-spotting system [61]; when the figures go up a few years later, the vendors let the matter pass quietly [714].

More convincing are projects undertaken by specific store chains that look for known patterns of misuse. For example, an electrical goods chain in the New York area observed that offender profiling (by age, sex, race, and so on) was ineffective, and used purchase profiling instead to cut fraud by 82% in a year. Its technique involved not just being suspicious of high-value purchases, but training staff to be careful when customers were careless about purchases and spent less than the usual amount of time discussing options and features. These factors can be monitored online, too, but one important aspect of the New York success is harder for a Web site: employee rewarding. Banks give a $50 reward per bad card captured, which many stores just keep, so their employees don't make an effort to spot cards or risk embarrassment by confronting a customer. In New York, some store staff were regularly earning a weekly bonus of $150 or more [525].

With the human out of the loop at the sales end, the only psychology from which a site designer can leverage is that of the villain. It has been suggested that an e-commerce site should have an unreasonably expensive "platinum" option, which few genuine customers will want to buy [721]. This performs two functions. First, it allows

Plaid Ex1003
Page 160 of 372

you to do rudimentary purchase profiling. Second, it fits with the model of *Goldilocks pricing*, developed by online economists Shapiro and Varian, who point out that the real effect of airlines offering first-class fares is to boost sales of business class seats to travelers who can now convince their bosses (or themselves) that they are being "economical" [696]. Another idea is to have a carefully engineered response to suspect transactions: if you just say "bad card, try another one," then the fraudster probably will. You may even end up being "used" by the crooks as an online service that tells them which of their stolen cards are on the hot list, and this can upset your bank (even though the banks are to blame for the system design). A better approach is claim that you're out of stock, so the bad guy will go elsewhere [721].

19.3.4 Economics

There's a lot of misinformation about credit card fraud, with statistics quoted selectively to make points. In one beautiful example, VISA was reported to have claimed that card fraud was up *and* that card fraud was down, on the same day [380]

However, a consistent pattern of figures can be dug out of the trade publications. The actual cost of credit card fraud, before the recent rise, was about 0.15% of all international transactions processed by VISA and MasterCard [652], while national rates varied from 1% in America to 0.2% in the U.K. to under 0.1% in France and Spain. The prevailing business culture has a large effect on the rate. U.S. banks, for example, are much more willing to send out huge junk mailings of pre-approved cards to increase their customer base, and write off the inevitable pre-issue fraud as a cost of doing business. In other countries, banks are more risk-averse.

France is interesting, as it seems, at first sight, to be an exceptional case, in which a particular technology has brought real benefits. French banks introduced chip cards for all domestic transactions in the late 1980s, and this reduced losses from 0.269% of turnover in 1987 to 0.04% in 1993 and 0.028% in 1995. However, there is now an increasing amount of cross-border fraud. French villains use foreign magnetic stripe cards— particularly from Britain [315, 652]—while French chip cards are used at merchants in non-chip countries [166]. But the biggest reduction in Europe was not in France but in Spain, where the policy was to reduce all merchant floor limits to zero and make all transactions online. This cut their losses from 0.21% of turnover in 1988 to 0.008% in 1991 [73].

The lessons appear to be that, first, card fraud is cyclical, as new defenses are introduced and the villains learn to defeat them; and second, that the most complicated and expensive technological solution doesn't necessarily work best in the field.

19.4 Online Credit Card Fraud: The Hype and the Reality

We turn now from traditional credit card fraud to the online variety. There was great anxiety in the mid-1990s that the use of credit cards on the Internet would lead to an avalanche of fraud, as "Evil Hackers" intercepted emails and Web forms, and har-

Plaid Ex1003
Page 161 of 372

vested credit card numbers by the million. These fears drove banks and software vendors to devise two protocols to protect Web-based credit card transactions: SSL and SET, which I'll explain in the next section.

The hype surrounding this type of fraud has been grossly overdone. Intercepting email is indeed possible, but it's surprisingly difficult in practice—so much so that governments are bullying ISPs to install snooping devices on their networks to make court-authorized wiretaps easier [114]. But the cost of such devices is so high that the ISPs are resisting this pressure as forcefully as they can. I'll go into this further in Chapter 21. And although it is possible to redirect a popular Web page to your own site using tricks such as DNS cache poisoning, it's a lot simpler to tap the plain old telephone system—and no one worries much about the few credit card numbers that get harvested from hotel guests this way.

Credit card numbers are indeed available on the Net, but usually because someone hacked the computer of a merchant who disobeyed the standard bank prohibition against retaining customer credit card numbers after being paid. (As this book was going to press, VISA announced that, starting in 2001, all its merchants will have to obey 10 new security rules; for example, they must install a firewall, keep security patches up to date, encrypt stored and transmitted data, and regularly update antivirus software [752].) Likewise, fraudulent Web-based transactions do occur, but mainly because of poor implementation of the system whereby cardholder addresses are checked during authorization. The real problem facing dot-coms is disputes.

It is easy to repudiate a transaction. Basically, all the customer has to do is call the credit card company and say, "I didn't authorize that," and the merchant will be saddled with the bill. This was workable in the days when almost all credit card transactions took place locally, and most were for significant amounts. If a customer fraudulently repudiated a transaction, the merchant would pursue them through the courts and harrass them using local credit reference agencies. In addition, the banks' systems are often quite capable of verifying local cardholder addresses.

But the Internet differs from the old mail order/telephone order regime, in that many transactions are international, amounts often are small, and verifying overseas addresses via the credit card system is problematic. Often, all the call center operator can do is check that the merchant seems confident when reading an address in the right country. Thus, the opportunity for repudiating transactions—and getting away with it—is hugely increased. There are particularly high rates of repudiation of payment to porn sites. No doubt some of these disputes happen when a transaction made under the influence of a flush of hormones turns up on the family credit card bill, and the cardholder has to repudiate it to save his marriage; but many are the result of blatant fraud by operators.

At the time of writing, the press was reporting that the Federal Trade Commission was prosecuting the operators of scores of adult Web sites, including `playboy. com`, for billing thousands of users for supposedly free services. The scam was to offer a "free tour" of the site, demand a credit card number, supposedly to verify that the user was over 18, and then bill him anyway. Some sites billed other consumers who have never visited them at all [389]. (Of course, none of this should have surprised the student of more traditional telecomms fraud, as it's just cramming in a new disguise.) If even apparently large and "respectable" Web sites such as `playboy. com` indulge in such practices, it's much easier for consumers to get away with fraudulently repudiating transactions.

Plaid Ex1003
Page 162 of 372

The critical importance of this for online businesses is that, if more than a small percentage of your transactions are challenged by customers, your margins will be eroded; and in extreme cases your bank may withdraw your card acquisition service. It has been reported that the collapse of sportswear merchant `boo.com` was because it had too many returns: its business model assumed a no-quibble exchange or refund policy. But too many of its shipments were the wrong size, or the wrong color, or just didn't appeal to the customers. In the end, the credit card penalties were the straw that broke the camel's back [721].

This history suggests that technological fixes may not be as easy as many vendors claim, and that the main recources will be essentially procedural. American Express has announced that it will offer its customers credit card numbers that can be used once only; this will protect customers from some of the scams. In order not to run out of numbers, they will issue them one at a time to customers via their Web site (which will drive lots of traffic to them) [204]. Many other bankers are already coming to the conclusion that the way forward lies with better address verification, rather than with cryptography [62].

However, if you're working as a security engineer, then a lot of your clients will want to talk about technical matters, such as encrypting credit card numbers, so we'll look at the available crypto mechanisms anyway.

19.5 Cryptographic Protection Mechanisms

The existing cryptographic protection mechanisms used by the bank card industry—the PINs used at ATMs and some point-of-sale terminals, and the CVVs described in Section 19.3.2, which make card forgery more difficult—are largely ineffective online, so new mechanisms were developed. The most widely used is the Secure Sockets Layer protocol (SSL), an encryption system bundled with most Web browsers.

19.5.1 SSL

Recall that in public key encryption, a server can publish a public key *KS* and any Web browser can then send a message *M* containing a credit card number to it, encrypted using *KS:* $\{M\}_{KS}$. This is, in essence, what SSL does, although in practice it is more complicated. SSL was developed to support encryption and authentication in both directions, so that both `http` requests and responses could be protected against both eavesdropping and manipulation.

Here is a simplified description of the SSL version used to protect Web pages that solicit credit card numbers.

1. The client sends a *client hello* message to the server, which contains its name *C*, a transaction serial number *C#*, and a random nonce N_C.
2. The server sends a *server hello* message, which contains its name *S*, a transaction serial number *S#*, a random nonce N_S, and a certificate *CS* containing

Plaid Ex1003
Page 163 of 372

its public key *KS*. The client now checks the certificate *CS* back to a root certificate issued by a company such as Verisign and stored in the browser.

3. The client sends a *key exchange* message containing a *pre-master-secret* key K_0, encrypted under the server public key *KS*. It also sends a *finished* message with a message authentication code (MAC) computed on all the messages to date. The key for this MAC is the *master-secret* K_1. This key is computed by hashing the pre-master-secret key with the nonces sent by the client and server: $K_1 = h(K_0, N_C, N_S)$. From this point onward, all the traffic is encrypted; we'll write this as $\{...\}_{KCS}$ in the client-server direction and $\{...\}_{KSC}$ from the server to the client. These keys are generated in turn by hashing the nonces with K_1.
4. The server also sends a *finished* message with a MAC computed on all the messages to date. It then finally starts sending the data.

$C \rightarrow S$: $C, C\#, N_C$

$S \rightarrow C$: $S, S\#, N_S, CS$

$C \rightarrow S$: $\{K_0\}_{KS}$

$C \rightarrow S$: $\{finished, MAC(K_1, everything_to_date)\}_{KCS}$

$S \rightarrow C$: $\{finished, MAC(K_1, everything_to_date)\}_{KSC}, \{data\}_{KSC}$

The SSL design goals included minimizing the load on the browser, and then minimizing the load on the server. Thus, the public key encryption operation is done by the client, and the decryption by the server; the standard encryption method (*ciphersuite*) uses RSA, for which encryption can be arranged to be much faster than decryption. (This was a wrong design decision, as browsers generally have a lot more compute cycles to spare than servers; the use of RSA has created a brisk aftermarket for crypto accelerator boards for Web servers.) Also, once a client and server have established a pre-master-secret, no more public key operations are needed, as further master-secrets can be obtained by hashing it with new nonces.

The full SSL protocol is more complex than this, and has gone through a number of versions. It supports a number of different ciphersuites; for example, export versions of browsers can be limited to 40-bit keys—a condition of export licensing that was imposed for many years by the U.S. government. Other ciphersuites support signed Diffie-Hellman key exchanges for transient keys, to provide forward and backward secrecy. SSL also has options for bidirectional authentication, so that if the client also has a certificate, this can be checked by the server. In addition, the working keys *KCS* and *KSC* can contain separate subkeys for encryption and authentication. For example, the most commonly used ciphersuite uses the stream cipher RC4 for the former and HMAC for the latter, and these need separate keys.

Although early versions of SSL had a number of bugs [784], the latest version (called TLS by Microsoft) appears to be sound (but it has to be implemented carefully [116]). It is being used for much more than electronic commerce—an example being medical privacy, where it's likely to replace proprietary networks and allow confidential patient data to be sent over the Internet [175]. SSL is also spreading from Web applications, and is now incorporated as an option in Win2K, where it can be used to set up secure sessions between machines in different domains. There are problems,

Plaid Ex1003
Page 164 of 372

though, and some of the most glaring have to do with the nature and management of certificates. We'll talk about them in Section 19.5.3.

19.5.2 SET

Early experience with credit cards and the Internet showed that the real risk of compromise of credit card numbers did not come from eavesdropping on IP traffic—of which no case has ever been confirmed—but from hacking merchant Web servers and other end systems, which often retain credit card numbers, and are frequently attacked. So, in 1995–1996, there was an effort to develop a better payment protocol, which would use digital signatures rather than credit card numbers.

Eventually, a consortium which included Microsoft, Netscape, VISA, and MasterCard, came up with the *Secure Electronic Transaction* (SET) protocol. The ideas behind SET were:

- Customers will have public key certificates too, not just merchants. So customers can sign transactions, which include payment orders to their banks.
- The customer signs and enciphers two separate messages, one to the merchant, which contains a description of the goods and the price but not the credit card number; and another to the bank, which contains the price and the credit card number but not the description of the goods. The signatures are linked.
- The back-end transaction processing from the acquiring bank to the brand to the card-issuing bank uses the existing legacy systems.

SET was supposed to reassure customers that online transactions would be secure, and to reduce the cost of fraud (mainly by denying credit card numbers to merchants). The business model was that SET transactions would be treated as if the cardholder were present; that is, the bank would assume the fraud risk, and the merchant discount would be lower.

Because of the number of different legacy systems that had to be supported, and the range of features demanded by various industry players, SET is even more complex than SSL. Again, I'll give a simplified account of it.

First, the customer sends the merchant server her certificate *CC* for her public key *KC*, and a nonce, N_C. The server replies with certificated public keys for the merchant (*CS, KS*) and its bank (*CB, KB*), plus a transaction sequence number *S#*. Then the customer sends a message containing an order, encrypted under the merchant's public key, and a payment instruction, encrypted under the bank's public key. Hashes of both of these are signed with the customer's private signing key. Next is an authorization step, which can be performed online or deferred, as appropriate: the server sends the payment instruction to the acquiring bank, together with a summary of the order, which includes the amount payable but not the exact description of the goods. The bank checks all this and refers to the card-issuing bank, if necessary. If everything's in order, it sends the server an authorization response similar to the traditional one (with an amount and an authorization code), fortified with a signature.

$C \rightarrow S$: C, N_C, CC

$S \rightarrow C$: $S, S\#, CS, CB$

Plaid Ex1003
Page 165 of 372

$C \rightarrow S$: $\{Order\}_{KC}$, $\{Payment\}_{KB}$, sig_{KC} $\{h(Order), h(Payment)\}$

$S \rightarrow B$: $\{Summary\}_{KB}$, $\{Payment\}_{KB}$

$B \rightarrow S$: sig_{KB} $\{Auth_response\}$

SET appears to have met its specifications, but failed to succeed in the marketplace. The reasons are instructive.

- *First, the benefits turned out to be less than expected.* Many large merchants were breaking their cardholder agreements by retaining customer credit card numbers—principally for use as indexes in marketing databases—and were not prepared to stop using them. So a feature was added whereby merchants could get the credit card number from the acquiring bank. This was thought to negate much of the hoped-for security improvement. (In fact, it wasn't that bad as banks could have issued credit card numbers that were valid only for SET transactions, so stealing them wouldn't have mattered.)
- *Second, the costs were too high.* Building a public key infrastructure to issue all credit cardholders with public key certificates would have been enormously expensive. Performance was also an issue.
- *Third, there was nothing in it for the customers.* Customers trading on the Web under MOTO rules could reverse the transaction if they were unhappy—not just about the payment, but about the service, the product, or anything else. Using SET transferred them to cardholder-present rules, and in many countries removed this protection. Thus, customers were much worse off and would have been insane to use SET. Also, installing SET usually involved downloading megabytes of SET wallet and going through a laborious certification procedure.

In the end, SET cost too much and delivered too little; and as far as the customers were concerned, it was a disaster. It is being allowed to expire quietly. The main lesson is, perhaps, that when designing systems for e-business, you should deal with issues as they are in practice, rather than in theory, and think about how your design will affect the interests of principals other than your client.

19.5.3 PKI

Public key infrastructures (PKIs) are still an issue; there is frequent semantic confusion between "public (key infrastructure)" and "(public key) infrastructure." In the first, the infrastructure can be used by whatever new applications come along; I call this an *open PKI;* in the second, it can't; I call this a *closed PKI.*

Examples of open PKIs are:

- Merchants using SSL are supposed to have certificates for their public keys, and several companies such as Verisign will certify a public key as belonging to a particular company after doing appropriate due diligence.
- There are many proposals to base new online services, and particularly business-to-business services, on certified digital signatures.

Plaid Ex1003
Page 166 of 372

- Many governments are thinking of issuing their citizens with public key certificates, probably in smartcards, as next-generation identity cards. Although most businesses are really only interested in whether they will be paid, governments offer a range of services (such as tax and welfare) that can be cheated by people who can masquerade as more than one person. So there is much government interest in promoting the use of PKI technology, and this has led to legislation, which I'll discuss in Chapter 21.

The classic examples of closed PKIs are to be found in the networks operated by military agencies and by banking service providers such as SWIFT, which use asymmetric cryptography but do not publish any keys. Now that Win2K includes SSL as an authentication mechanism, it can be used to set up secure wide area networking across a number of scattered sites in a company; and if the number of sites is at all large, this may involve the company operating its own PKI to manage the keys. So closed PKIs may become much more common; and even where a service using asymmetric cryptography is offered to the public, there may be no keys published. An example is the Mondex electronic purse, which uses RSA cryptography and further protects the keys in tamper-resistant smartcards.

At the time of writing, PKI was one of the most heavily promoted protection technologies. However, it has a number of intrinsic limitations, many of which have to do with the first interpretation—namely that the infrastructure is provided as a public service that anyone can use. I discussed many of the underlying problems in Chapter 6. Naming is difficult; and a certificate saying, "Ross Anderson has the right to administer the machine foo.com" means little in a world with dozens of people of that name.

One way to solve the naming problem is for each business to run its own closed PKI, which might be thought of at the system level as giving customers a unique account number which isn't shared with anyone else. This leads to the "one key or many" debate. Should I expect to have a single signing key to replace each of the metal keys, credit cards, swipe access cards, and other tokens that I currently carry around? Or should each of these be replaced by a different signing key? The second option is more convenient for business as sharing access tokens can lead to huge administrative costs and liability issues. It also protects the customer: I don't want to have to use a key with which I can remortgage my house to make calls from a payphone. It's just too easy to dupe me into signing a message by having the equipment display another, innocuous, one. (I don't know how to be confident even of a digital signature I make on my own PC, and I've worked in security for over fifteen years. Checking all the software in the critical path between the display and the signature software is way beyond my patience.) But the existing PKI machinery was largely developed to provide an electronic replacement for the telephone book, and tends to assume that everyone will have a unique name and a unique key. This in turn means an open PKI architecture.

This leads to political issues, such as, which CAs do we trust, and why? Various attempts have been made by governments to license certification authorities and to impose a condition that there be "back doors" for law enforcement access. Governments overwhelmingly favor the one-key-fits-all model of the world. It's also possible that open PKIs will be favored by network economics, which I discuss in Section 19.6: once a single PKI becomes dominant, the pressure on everyone to use it could lead to its being entrenched as a monopoly. (This is the reason for the high stock market valuation of VeriSign.)

Plaid Ex1003
Page 167 of 372

There are numerous issues of implementation detail. For example, the dominant certificate format (X.509) does not have the kind of flexible and globally scalable 'hot card' system that the credit card industry has developed over the years. It rather assumes that anyone relying on a certificate can download a *certificate revocation list* from the issuing authority. This is tiresome and inefficient Better ways of managing certificate revocation have been proposed; the question is whether they'll get implemented. Also, X.509 is designed to certify names, when for most purposes people want to certify an authorization.

There are many other limitations of certificates:

- *Most users disable the security features on their browsers*, even if these weren't disabled by default when the software shipped. Recall that the third step of the SSL protocol was for the client browser to check the certificate against its stored root certificates. If the check fails, the browser may ask the client for permission to proceed; but the way most browsers are configured, it will just proceed anyway. This lets many e-commerce sites save themselves money by using expired certificates or even self-signed certificates; most users don't see the warnings (and wouldn't know how to respond if they did).
- *The main vendors' certificates bind a company name to a DNS name*, but are not authorities on either; and they go out of their way to deny all liability.
- *Competition in the certificate markets is blocked* by the need to get a new root certificate into Microsoft Internet Explorer (VeriSign stockholders will consider this to be not a bug but a feature).
- *Even when you do get a valid certificate, it may be for a company and/or DNS name different from that of the site you thought you were shopping at*, because the Web site hosting or the credit card acquisition was outsourced.
- *U.S. export regulations have meant that large numbers of sites use weak encryption*. A recent survey of SSL security showed that of 8,081 different secure Web servers, 32% weren't, for various reasons—too-short keys, weak ciphersuites, and expired certificates being the main causes [567].

There is also a serious problem with consumer protection law. Introducing a presumption that digital signatures are valid undermines the signer's rights; in paper systems the risk of fraud is borne by the party who relies on the signature. In the absence of such a presumption, it makes no difference to the cardholder's liability whether the sites at which he shopped had valid certificates with appropriate names; and in any case, there's no convenient way for him to record his transactions to show that he exercised due diligence. I go into some of these issues at greater length in Chapter 21.

In short, while public key infrastructures can be useful in some applications, they are unlikely to be the universal solution to security problems as their advocates seem to believe. They don't tackle most of the really important issues at all.

19.5.4 EDI and Business-to-Business Systems

The early examples of electronic commerce given in Section 19.2, such as the telegraph and EDI, were largely business-to-business systems. These systems provide

Plaid Ex1003
Page 168 of 372

many of the examples of working PKIs. We looked at an example in detail in Section 9.3.1—the SWIFT network used since the mid-1970s to send secure payment messages between banks in different countries, and upgraded in the 1990s to use public key techniques. Essentially, the same architecture has been adopted for a number of other systems, including the CREST system used to register ownership of all U.K.-listed equities and to support share dealing between banks and brokers.

A more modern example of a business-to-business system is Bolero, an EU system that handles electronic bills of lading [262, 458, 492]. These are the legal documents that confer ownership of shipping cargoes. Their average value is about $25,000, but, depending on the price of oil, an oil tanker's cargo can be worth $100,000,000. Cargoes are often traded many times while the ship is at sea, and many of the traders are rather dubious shell companies. In addition, the enforcement of sanctions against rogue states means that their national intelligence agencies are often involved in using such shell companies to buy things like oil when they shouldn't. So this is a high-value, high-threat environment. Quite some care has to be taken to ensure that bills of lading cannot be duplicated, and that their ownership history can be established.

Bolero uses two main protection mechanisms. First, there is tamper resistance. IBM 4753 or 4758 cryptoprocessor cards are used to hold the bills; and there is a protocol involving digital signatures that is used to transfer them. Second, a central registry is used to ensure that the name of the unique holder can be determined at all times. There is also a registration authority (which vets the credentials of organizations) and a certification authority (which signs the public keys of individuals authorized by registered organizations).

Another example is healthcare EDI systems, which typically send test results, such as radiology and cytology from large hospitals and laboratories, to family doctors and local clinics. Here the requirement isn't non-duplicability, but authenticity plus confidentiality. Early systems provided this using closed, proprietary messaging networks; more modern systems have encryption and authentication mechanisms that work at message gateways. The dozens of messages sent each day from a hospital laboratory to a general medical practice are batched up, signed with the hospital's signing key, encrypted with the practice's public key, and shipped by email. In an ideal world, messages would be signed individually by the consultant writing the opinion, and addressed to the physician treating the patient; in practice, this is hard, as both labs and physicians use proprietary systems that can communicate only using EDIFACT gateways. These are specialized systems that do format conversion; they are too expensive for everyone to have them running on their PC. This design causes a number of problems. For example, if a record of the digital signatures of lab reports is to be kept in case of malpractice litigation, then the whole batch must be kept, which conflicts with privacy rules about the destruction of records pertaining to patients who have died or moved away.

This kind of problem isn't limited to healthcare. Many systems have message processing functions that cause subsets of transaction data to go to different destinations in an organization. Protecting the integrity of structured data can be much harder than it looks.

So, for a number of reasons, implementing business-to-business secure communications isn't at all straightforward. As innovative business models proliferate, there are bound to be expensive design errors. I'll discuss how to reduce the probability of these in Part 3.

Plaid Ex1003
Page 169 of 372

19.5.5 E-Purses and Micropayments

In the first half of the 1990s, a large number of *electronic purse* systems were developed. The idea was to replace debit cards with something that would work offline without being more vulnerable to forgery. The typical implementation involved both customers and merchants having chip cards, each with a value counter, plus a card-to-card payment protocol whereby two cards would authenticate each other—one would be debited and the other would be credited. I described the design of a typical protocol in Section 2.7.1.

The promoters of these schemes had huge hopes. Europay Austria, for example, was confident that its product would displace 20% of cash transactions within three years. But by the end of the decade, the outcome was disappointing. Even where an e-purse chip has been built into standard bank cards and issued to the entire customer base, as with Proton in Belgium and the Geldkarte in Germany, usage remains disappointing. (The business is surveyed in [763].) The slow take-off might have been expected, given the history of credit cards; like them, e-purses suffer from what economists call *network externalities*, meaning that the more the cards are used, the more merchants accept them, and the more useful they become—not just to their existing customers but to potential adopters. But when few people use them, merchants don't have a motive to buy terminals; and with few terminals, they attract few users. At best, such schemes have some way to go before they reach critical mass; they may get there eventually, driven by applications such as payphones, and then they could grow very quickly indeed. I discuss the underlying economics further in Section 19.6.

The one exception to the slow start imposed on new payment mechanisms by network externalities may be the micropayment mechanism that might be shipped with the second phase of third-generation mobile phones, which I described in Section 17.3.4. As all purchasers of next-generation phones will be using this system to pay for calls, as well as for value-added services, the extra cost involved in adding a new service should be low, while the customer base should be substantial within a few years. There is a risk that, as with Geldkarte, this will end up as a system that's widely deployed but has few active users; but there's also the possibility that it might fly. (Phone-based payments could also become a serious problem if, as seems likely, third-generation phones get the capability to run Java applets—with all the potential for malicious code, feature interaction, and general confusion that this would bring.)

It's also conceivable that third-generation phones could provide a close thing to a universal open PKI, if certification authorities are used to sign users' public key certificates in a way that's usable by other services. On the other hand, if the top-level public key protocol is the Royal Holloway one, and (as in its established government uses), this escrows signing keys as well as confidentiality keys [50], then it will be much less useful, because the evidential value of signatures produced by the system will be undermined.

19.6 Network Economics

The network externalities mentioned above don't just apply to people trying to launch new payment systems, but are of very much wider importance. Many communications systems, and the downstream applications they support, suffer from them. They dictate

Plaid Ex1003
Page 170 of 372

not just e-commerce business models, but also many of the problems that the security engineer must wrestle with.

The key observation is that the more users there are on a system, the more people there are to talk to, and so the more useful it is to each user. (This is sometimes referred to as *Metcalfe's law.*) There are many documented examples of how the resulting positive feedback can give a large advantage to the early players in a market, and lead toward a monopoly. For example, in the early years of the twentieth century, AT&T's dominance of long-distance telephone communication enabled it to crush local competition and to establish itself as the near-monopoly provider in the United States until its break-up in 1984. Compatability is also crucial. In the 1950s, there was a battle between CBS and RCA over whose standard for color television would prevail; although the FCC endorsed CBS, its standard wasn't backward-compatible with the millions of installed black-and-white TV sets. RCA's standard was, so it won.

The most obvious effect of this positive feedback is that once a network passes a certain critical size, it grows rapidly. The telegraph, the telephone, the fax machine and, most recently, the Internet, have all followed this model. Another effect is that there are enormous rewards for being first. This happens to some extent in traditional businesses such as cars where supplyside economies of scale help big firms to grow bigger. But there are limits: once they became too large, companies such as General Motors couldn't react quickly enough to compete against upstarts such as Toyota. With networks, the economies of scale occur on the demand side, and so there is no upper limit on growth.

Network effects aren't limited to the kind of networks that involve shipping electrical or optical signals, which economists refer to as *real networks*. They also apply to *virtual networks*, of which the classic example is software. Recall the battle for supremacy between the PC and the Mac in the mid-1980s: once it was clear that there were going to be more users of PCs than Macs, software houses concentrated on shipping their products for the PC first and the Mac only afterward, if at all. This meant that there was more software for the PC, so people were more likely to buy PCs, and the positive feedback continued.

This isn't limited to PC architecures, but works at all sorts of levels including application software and file formats. Once most people started using Microsoft Word for documents, that was even more reason for everybody else to (regardless of the risk from Word macro viruses). And such a network effect can be enhanced by arranging that your file formats are difficult for other companies' programs to read, and by changing them often; I'll discuss this in the next section.

One of the features of markets exposed to network effects is lock-in. This can be either technology lock-in or vendor lock-in. Technology lock-in often involves complementary suppliers, as with the software vendors whose bandwagon effect carried Microsoft to victory over Apple. A side effect of this is that successful networks appeal to complementary suppliers rather than to users—the potential creators of "killer apps" need to be courted. Once the customers have a substantial investment in complementary assets, they will be locked in. Andrew Odlyzko observes that much of the lack of user-friendliness of both Microsoft software and the Internet is due to the fact that both Microsoft and the Internet achieved success by appealing to developers. The support costs that Microsoft dumps on users—and, in fact, even the cost of the time wasted waiting for PCs to boot up and shut down—greatly exceed its turnover [595].

Plaid Ex1003
Page 171 of 372

So there are three particularly important features of information technology markets.

- *First, technology often has high fixed costs and low marginal costs.* The first copy of a chip or a software package may cost millions to produce, but subsequent copies may cost very little. This isn't unique to information markets; it's also seen in business sectors such as airlines and hotels. In all such sectors, pricing at marginal cost will tend to drive revenues steadily down toward the cost of production (which in the case of information is zero).
- *Second, there are often high costs to users from switching technologies, which leads to lock-in.* Such markets may remain very profitable, despite the low marginal costs.
- *Third, there are often network externalities of the sort discussed earlier.* The value of a product to a user depends on how many other users adopt it.

All three of these effects tend to lead to winner-take-all market structures with dominant firms. Indeed, firms will attempt to manipulate these effects to gain competitive advantage in various ways.

One common strategy, for example, is differentiated pricing. This means pricing the product or service not to its cost but to its value to the customer. This is familiar from the world of air travel: you can spend $200 to fly the Atlantic in coach class, $2,000 in business class, or $5,000 in first. (As noted in Section 19.3.3, this is also a classic case of Goldilocks pricing: the main function of the first-class fare is to enable people who fly business class to claim they are being economical.) This business model is spreading widely in the software and online services sectors. A basic program or service may be available free; a much better one for a subscription; and a "gold" service at a ridiculous price. In many cases, the program is the same except that some features are disabled for the budget user while the "gold" user gets a high-quality helpline. Many of the protection mechanisms you will come across have as their real function the maintenance of this differential.

Another strategy is to manipulate *switching costs.* The long-term value to your ISP of your account, in the sense of the discounted future earnings, should be equal to the total amount of money (and hassle) involved in the customers' switching to a competitor. So the ISP will do its utmost to make it easy for you to switch to them, but difficult to switch from them. This applies with particular force to dominant-firm markets where the incumbent tries to build a monopoly which its competitors try to attack. Incumbents try to increase the cost of switching, whether by indirect methods, such as controlling marketing channels and building industries of complementary suppliers, or by direct methods, such as making systems incompatible and hard to reverse-engineer. Meanwhile, market entrants try to do the reverse: they look for ways to reuse the base of complementary products and services, and to reverse-engineer whatever protection the incumbent builds in. They may use penetration pricing—selling cheaply to subsidize switching—and the incumbent may respond with vaporware, designed to increase users' perception of the opportunity cost of switching.

As technology advances, even seemingly impregnable monopolies can be replaced, so the competition can be vicious. Extensive use is made of protection mechanisms, from tamper-resistant devices to proprietary encryption algorithms.

Plaid Ex1003
Page 172 of 372

19.7 Competitive Applications and Corporate Warfare

This leads us to the applications of information security mechanisms whose goal isn't to protect the customers or their data, but to either entrench or attack a monopoly.

Sometimes the mechanisms in use are obvious. For example, manufacturers of game consoles try to keep their platforms closed so that they can monopolize sales of accessories and impose conditions on games software vendors—which may involve not just royalties but also exclusivity agreements. Legal solutions such as copyrighted interfaces aren't enough, as in many countries there is an exemption from copyright law for firms doing reverse-engineering in order to build a compatible product. So cryptographic challenge-response protocols are used to authenticate genuine accessories and game cartridges; competitors hire reverse-engineering labs to dig out the keys. I mentioned other applications of accessory control in Section 2.2.

Another example comes from Microsoft Passport. This is a system whose ostensible purpose is single sign-on: a Passport user doesn't have to think up separate passwords for each participating Web site, with all the attendant hassle and risk. Instead, some sites use Passport, a central authentication server run by Microsoft, to which users log on. Servers use Web redirection to connect their Passport-carrying visitors to this server; authentication requests and responses are passed between them by the user's browser in encrypted cookies. So far, so good.

But the real functions of Passport are somewhat more subtle [727]. First, by patching itself into all the Web transactions of participating sites, Microsoft can collect a huge amount of data about online shopping habits, and enable participants to swap it. The redirection and cookie mechanisms mean that, in effect, all the browsing sessions you have at participating sites become one single session, managed by Microsoft. If every site can exchange data with every other site, then the value of a network of Web sites is the square of the number of sites, and there is a strong network externality. So one such network may come to dominate, and Microsoft hopes to own it. Second, the authentication protocols used between the merchant servers and the Passport server are proprietary variants of Kerberos, meaning the Web server must use Microsoft software rather than Apache or Netscape. In short, Passport isn't as much a security product as a play for control of both the Web server and purchasing information markets. It comes bundled with services such as Hotmail, is already used by 40 million people, and does 400 authentications per second on average. Its known flaws include that Microsoft keeps all the users' credit card details, creating a huge target; various possible middleperson attacks; and that a user can be impersonated by someone who steals their cookie file. Passport has a "logout" facility that's supposed to delete the cookies for a particular merchant, so users can use a shared PC with less risk, but this feature doesn't work properly for Netscape users [473].

The constant struggles to entrench or undermine monopolies, and to segment and control markets, determine many of the environmental conditions that make the security engineer's work harder. The markup language XML enables document content to be processed easily, and has the potential to build in a rich syntax of protection attributes [46]. However, it's not taking off as many people had hoped, because if Web pages become easily machine-readable, then comparison shopping bots are easier to build. In addition, many online merchants gauge demand by discounting perhaps every hundredth transaction by a random amount. This wouldn't work if shoppers had tools to hit the site again and again until they got a bargain. In general, there is a constant

Plaid Ex1003
Page 173 of 372

struggle between the designers of intermediaries—from Web caches to anonymous communication services—who wish to control a user's transactions is various ways, and merchant sites, that wish to break this control and "own" the user directly.

I'll come back to the effects of network economics on security in Section 22.6.

19.8 What Else Goes Wrong

An important survey of the things that went wrong with First Virtual, one of the first online banks, revealed that the typical problem was an upset customer calling at three in the morning and wanting to speak to someone in Korean about a missing payment. Investigation would typically reveal that the cause was an obscurely broken implementation of one or more Internet protocols, or a mistake made in typing an email address. Solving such problems is not currently a core competence of the typical financial institution [129].

This pattern—that most of the problems come from unanticipated bugs and blunders—was to be expected from other application areas; and it has continued. A number of large online stores, including `Buy. com`, `Staples. com` and Amazon-backed crafts retailer `eZiba. com`, were hit by pricing errors. At `Buy. com`, a coupon meant to be worth $50 off any order of $500 or more actually gave $50 off any purchase, making any item sold for $50 or less free to the buyer. At `eZiba. com`, each customer was offered a $20 voucher; people logged on with multiple names and found that they could use the discount many times [671]. Errors like these are nothing new; there have been numerous cases in the past of a special offer being carelessly designed, or even just being more popular than planned for. The difference is that, on the Net, an error can become widely known very rapidly and lead to large losses.

There have also been some interesting attacks on specific systems. The *Radio Data System* (RDS) adds a data channel to broadcast radio so the receiver can tell which station it is and what sort of content is being broadcast. This enables the radio to switch automatically to the strongest transmitter for your favorite network when you're driving along in your car; you can also program the radio to interrupt you if traffic information appears on another channel. (An RDS radio has two tuners, one of which constantly sweeps the band looking for a higher-strength or higher-priority signal). Pirate radio stations have developed the trick of marking their content falsely as traffic news, so lots of car radios switch to them automatically [306]. This isn't always obvious if the pirate is playing the same general kind of music as the station you were just listening to. It's also not obvious how genuine radio stations could be authenticated; perhaps the FCC would act as a certification authority, and issue certificates along with spectrum licenses. But then how would a genuine station that turned pirate be revoked? The moral is that often it is necessary to protect the integrity of distribution channels; and, more generally, that interesting new features often turn out to be vulnerable to interesting new exploits.

Computer games are another fertile field for finding hacks. An example is Quake, a distributed game whose source code is openly distributed. Some players have exploited

Plaid Ex1003
Page 174 of 372

their access to the code to modify the Quake client so that they can cheat [633]. This raises a number of questions, particularly with e-commerce applications that employ Java applets or other code that is run on a potentially hostile customer machine. There's no obvious way such applets can protect themselves against virtualization. At the very least, it seems prudent to have a non-Java version of your site. This will not only mean that you can do business with people like me, who leave Java turned off for security reasons; it will also give you a fallback mechanism to block fraud based on bad applets without having to close down your site for redevelopment.

19.9 What Can a Merchant Do?

In general, the advice I give to people concerned about e-commerce risks is that they are not much different from normal business and IT risks.

You should make sure your developers understand the business model; use a structured development methodology; test your code thoroughly; don't be too clever; look around for proven ideas which you can adopt; by all means worry about the firewall through which your internal systems are connected to the Net, but pay particular attention to internal controls to discourage, prevent, and detect insider fraud. Expect that no matter what you do, things will go wrong and may have to be changed quickly. Don't be too greedy; if you put "We guarantee you can't lose money shopping online with us" on your Web page and "Our records are the sole and definitive evidence of all transactions between us" in the small print, you may expect to have the TV crew from a consumer rights program camping on your doorstep one day.

This much is motherhood and apple pie for any company IT director. The main way in which e-commerce appears different—at least as of mid-2000—is the seriously enhanced risk and cost of customers repudiating credit card transactions. There's no obvious fix; but a useful damage limitation strategy is to have a controlled procedure through which customers are invited to complain, rather than going directly to their credit card company on day one. For example, you can advertise a policy of allowing exchanges with a discount off the postage; and if customers do want a refund, ask them to print out and sign a form, and mail it to you by physical post, rather than offering a Web form for the purpose.

Above all, focus on the business risks, and have risk management documentation that you upgrade regularly in consultation with your auditors, insurers, and directors, rather than getting carried away with the latest technical security gizmos. Business is business, and just because there are now highly paid computer scientists designing business processes that used to be the domain of the work study clerks of old, this does not mean that things are any better. The risks associated with the "Net mentality" have to be carefully assessed and managed.

Finally, it may make sense to pay some attention to non-technical issues such as product liability. One critical advantage enjoyed by U.S. e-businesses is that foreign civil judgments aren't enforced by U.S. courts. So an e-business operating from the United States doesn't have to worry about being sued by consumers in faraway coun-

Plaid Ex1003
Page 175 of 372

tries, even if the local law lets them sue locally, because the judgments they get cannot be enforced. But in Europe, there are international reciprocal agreements: a U.K. retailer sued in a local Greek court can have the judgment enforced in the U.K. On the other hand, U.S. courts are accessible and can give punitive damages, while courts in many European countries are so expensive and give such small awards that product liability suits are rare. These considerations interact with credit card chargeback issues, but are not subsumed by them. They can be so complex that all I'll say is that you should get legal advice: the best location for your business may depend on what you're selling, and to whom.

19.10 Summary

Most of the problems facing online businesses are no different from those facing other organizations, and the network security risks are not much different from those facing traditional businesses. The real increased risks to an e-business have to do with ways in which traditional risk management mechanisms don't scale properly from a world of local physical transactions to one of worldwide, dematerialized ones. Credit card transaction repudiation is the main example at present. There are also significant risks to rapidly growing companies that have hired a lot of new staff but that don't have the traditional internal controls in place.

Research Problems

Most of the research behind the e-commerce protection mechanisms that are already deployed, or about to be, was done around 1994–1996. It may well be time for a second wave now that we can see what has worked, what can work but failed in the marketplace, and where the real problems are.

Further Reading

The early history of the telegraph can be found in a book by Major General RFH Nalder [569], while Tom Standage tells the story of its rapid deployment in Victorian times [729]. There is a survey of organized credit card counterfeiting in [592]. The official specification of SSL is hard to read; a better exposition is in [604]. The SET protocol is described in a book by its chief architect Li Song [509]. The problems of public key certification and infrastructures are analyzed in [42, 268]. Finally, the best book I know on network economics is by Carl Shapiro and Hal Varian [696].

Plaid Ex1003
Page 176 of 372

EXHIBIT C

Preface to the Second Edition

The first edition of *Security Engineering* was published in May 2001. Since then the world has changed.

System security was one of Microsoft's lowest priorities then; it's now one of the highest. The volume of malware continues to increase along with the nuisance that it causes. Although a lot of effort has gone into defence — we have seen Windows NT replaced by XP and then Vista, and occasional service packs replaced by monthly security patches — the effort put into attacks has increased far more. People who write viruses no longer do so for fun, but for profit; the last few years have seen the emergence of a criminal economy that supports diverse specialists. Spammers, virus writers, phishermen, money launderers and spies trade busily with each other.

Cryptography has also moved on. The Advanced Encryption Standard is being embedded into more and more products, and we have some interesting developments on the public-key side of things too. But just as our algorithm problems get solved, so we face a host of implementation issues. Side channels, poorly designed APIs and protocol failures continue to break systems. Applied cryptography is harder than ever to do well.

Pervasive computing also opens up new challenges. As computers and communications become embedded invisibly everywhere, so problems that used to only afflict 'proper computers' crop up in all sorts of other devices too. What does it mean for a thermometer to be secure, or an air-conditioner?

The great diversity of intelligent devices brings with it a great diversity of interests and actors. Security is not just about keeping the bad guys out, but increasingly concerned with tussles for power and control. DRM pits the content and platform industries against consumers, and against each other; accessory control is used to tie printers to their vendors' cartridges, but leads

Plaid Ex1003
Page 178 of 372

to antitrust lawsuits and government intervention. Security also interacts with safety in applications from cars through utilities to electronic healthcare. The security engineer needs to understand not just crypto and operating systems, but economics and human factors as well.

And the ubiquity of digital devices means that 'computer security' is no longer just a problem for a few systems specialists. Almost all white-collar crime (and much crime of the serious violent sort) now involves computers or mobile phones, so a detective needs to understand computer forensics just as she needs to know how to drive. More and more lawyers, accountants, managers and other people with no formal engineering training are going to have to understand system security in order to do their jobs well.

The rapid growth of online services, from Google and Facebook to massively multiplayer games, has also changed the world. Bugs in online applications can be fixed rapidly once they're noticed, but the applications get ever more complex and their side-effects harder to predict. We may have a reasonably good idea what it means for an operating system or even a banking service to be secure, but we can't make any such claims for online lifestyles that evolve all the time. We're entering a novel world of evolving socio-technical systems, and that raises profound questions about how the evolution is driven and who is in control.

The largest changes, however, may be those driven by the tragic events of September 2001 and by our reaction to them. These have altered perceptions and priorities in many ways, and changed the shape of the security industry. Terrorism is not just about risk, but about the perception of risk, and about the manipulation of perception. This adds psychology and politics to the mix. Security engineers also have a duty to contribute to the political debate. Where inappropriate reactions to terrorist crimes have led to major waste of resources and unforced policy errors, we have to keep on educating people to ask a few simple questions: what are we seeking to prevent, and will the proposed mechanisms actually work?

Ross Anderson
Cambridge, January 2008

Foreword

In a paper he wrote with Roger Needham, Ross Anderson coined the phrase ''programming Satan's computer'' to describe the problems faced by computer-security engineers. It's the sort of evocative image I've come to expect from Ross, and a phrase I've used ever since.

Programming a computer is straightforward: keep hammering away at the problem until the computer does what it's supposed to do. Large application programs and operating systems are a lot more complicated, but the methodology is basically the same. Writing a reliable computer program is much harder, because the program needs to work even in the face of random errors and mistakes: Murphy's computer, if you will. Significant research has gone into reliable software design, and there are many mission-critical software applications that are designed to withstand Murphy's Law.

Writing a *secure* computer program is another matter entirely. Security involves making sure things work, not in the presence of random faults, but in the face of an intelligent and malicious adversary trying to ensure that things fail in the worst possible way at the worst possible time . . . again and again. It truly is programming Satan's computer.

Security engineering is different from any other kind of programming. It's a point I made over and over again: in my own book, *Secrets and Lies*, in my monthly newsletter *Crypto-Gram*, and in my other writings. And it's a point Ross makes in every chapter of this book. This is why, if you're doing any security engineering . . . if you're even *thinking* of doing any security engineering, you need to read this book. It's the first, and only, end-to-end modern security design and engineering book ever written.

And it comes just in time. You can divide the history of the Internet into three waves. The first wave centered around mainframes and terminals.

Plaid Ex1003
Page 180 of 372

Computers were expensive and rare. The second wave, from about 1992 until now, centered around personal computers, browsers, and large application programs. And the third, starting now, will see the connection of all sorts of devices that are currently in proprietary networks, standalone, and non-computerized. By 2003, there will be more mobile phones connected to the Internet than computers. Within a few years we'll see many of the world's refrigerators, heart monitors, bus and train ticket dispensers, burglar alarms, and electricity meters talking IP. Personal computers will be a minority player on the Internet.

Security engineering, especially in this third wave, requires you to think differently. You need to figure out not how something works, but how something can be made to not work. You have to imagine an intelligent and malicious adversary inside your system (remember Satan's computer), constantly trying new ways to subvert it. You have to consider all the ways your system can fail, most of them having nothing to do with the design itself. You have to look at everything backwards, upside down, and sideways. You have to think like an alien.

As the late great science fiction editor John W. Campbell, said: ''An alien thinks as well as a human, but not like a human.'' Computer security is a lot like that. Ross is one of those rare people who can think like an alien, and then explain that thinking to humans. Have fun reading.

Bruce Schneier
January 2001

Preface

For generations, people have defined and protected their property and their privacy using locks, fences, signatures, seals, account books, and meters. These have been supported by a host of social constructs ranging from international treaties through national laws to manners and customs.

This is changing, and quickly. Most records are now electronic, from bank accounts to registers of real property; and transactions are increasingly electronic, as shopping moves to the Internet. Just as important, but less obvious, are the many everyday systems that have been quietly automated. Burglar alarms no longer wake up the neighborhood, but send silent messages to the police; students no longer fill their dormitory washers and dryers with coins, but credit them using a smartcard they recharge at the college bookstore; locks are no longer simple mechanical affairs, but are operated by electronic remote controls or swipe cards; and instead of renting videocassettes, millions of people get their movies from satellite or cable channels. Even the humble banknote is no longer just ink on paper, but may contain digital watermarks that enable many forgeries to be detected by machine.

How good is all this new security technology? Unfortunately, the honest answer is 'nowhere near as good as it should be'. New systems are often rapidly broken, and the same elementary mistakes are repeated in one application after another. It often takes four or five attempts to get a security design right, and that is far too many.

The media regularly report security breaches on the Internet; banks fight their customers over 'phantom withdrawals' from cash machines; VISA reports huge increases in the number of disputed Internet credit card transactions; satellite TV companies hound pirates who copy their smartcards; and law

Plaid Ex1003
Page 182 of 372

enforcement agencies try to stake out territory in cyberspace with laws controlling the use of encryption. Worse still, features interact. A mobile phone that calls the last number again if one of the keys is pressed by accident may be just a minor nuisance — until someone invents a machine that dispenses a can of soft drink every time its phone number is called. When all of a sudden you find 50 cans of Coke on your phone bill, who is responsible, the phone company, the handset manufacturer, or the vending machine operator? Once almost every electronic device that affects your life is connected to the Internet — which Microsoft expects to happen by 2010 — what does 'Internet security' mean to you, and how do you cope with it?

As well as the systems that fail, many systems just don't work well enough. Medical record systems don't let doctors share personal health information as they would like, but still don't protect it against inquisitive private eyes. Zillion-dollar military systems prevent anyone without a 'top secret' clearance from getting at intelligence data, but are often designed so that almost everyone needs this clearance to do any work. Passenger ticket systems are designed to prevent customers cheating, but when trustbusters break up the railroad, they cannot stop the new rail companies cheating each other. Many of these failures could have been foreseen if designers had just a little bit more knowledge of what had been tried, and had failed, elsewhere.

Security engineering is the new discipline that is starting to emerge out of all this chaos.

Although most of the underlying technologies (cryptology, software reliability, tamper resistance, security printing, auditing, etc.) are relatively well understood, the knowledge and experience of how to apply them effectively is much scarcer. And since the move from mechanical to digital mechanisms is happening everywhere at once, there just has not been time for the lessons learned to percolate through the engineering community. Time and again, we see the same old square wheels being reinvented.

The industries that have managed the transition most capably are often those that have been able to borrow an appropriate technology from another discipline. Examples include the reuse of technology designed for military identify-friend-or-foe equipment in bank cash machines and even prepayment gas meters. So even if a security designer has serious expertise in some particular speciality — whether as a mathematician working with ciphers or a chemist developing banknote inks — it is still prudent to have an overview of the whole subject. The essence of good security engineering is understanding the potential threats to a system, then applying an appropriate mix of protective measures — both technological and organizational — to control them. Knowing what has worked, and more importantly what has failed, in other applications is a great help in developing judgment. It can also save a lot of money.

The purpose of this book is to give a solid introduction to security engineering, as we understand it at the beginning of the twenty-first century. My goal is that it works at four different levels:

1. *As a textbook that you can read from one end to the other over a few days as an introduction to the subject.* The book is to be used mainly by the working IT professional who needs to learn about the subject, but it can also be used in a one-semester course in a university.
2. *As a reference book to which you can come for an overview of the workings of some particular type of system.* These systems include cash machines, taxi meters, radar jammers, anonymous medical record databases, and so on.
3. *As an introduction to the underlying technologies, such as crypto, access control, inference control, tamper resistance, and seals.* Space prevents me from going into great depth; but I provide a basic road map for each subject, plus a reading list for the curious (and a list of open research problems for the prospective graduate student).
4. *As an original scientific contribution in which I have tried to draw out the common principles that underlie security engineering, and the lessons that people building one kind of system should have learned from others.* In the many years I have been working in security, I keep coming across these. For example, a simple attack on stream ciphers wasn't known to the people who designed a common antiaircraft fire control radar so it was easy to jam; while a trick well known to the radar community wasn't understood by banknote printers and people who design copyright marking schemes, which led to a quite general attack on most digital watermarks.

I have tried to keep this book resolutely mid-Atlantic; a security engineering book has to be, as many of the fundamental technologies are American, while many of the interesting applications are European. (This isn't surprising given the better funding of U.S. universities and research labs, and the greater diversity of nations and markets in Europe.) What's more, many of the successful European innovations — from the smart-card to the GSM mobile phone to the pay-per-view TV service — have crossed the Atlantic and now thrive in the Americas. Both the science, and the case studies, are necessary.

This book grew out of the security engineering courses I teach at Cambridge University, but I have rewritten my notes to make them self-contained and added at least as much material again. It should be useful to the established professional security manager or consultant as a first-line reference; to the computer science professor doing research in cryptology; to the working police detective trying to figure out the latest computer scam; and to policy wonks struggling with the conflicts involved in regulating cryptography and anonymity. Above all, it is aimed at Dilbert. My main audience is the working

programmer or engineer who is trying to design real systems that will keep on working despite the best efforts of customers, managers, and everybody else.

This book is divided into three parts.

- The first looks at basic concepts, starting with the central concept of a security protocol, and going on to human-computer interface issues, access controls, cryptology, and distributed system issues. It does not assume any particular technical background other than basic computer literacy. It is based on an Introduction to Security course that I teach to second-year undergraduates.
- The second part looks in much more detail at a number of important applications, such as military communications, medical record systems, cash machines, mobile phones, and pay-TV. These are used to introduce more of the advanced technologies and concepts. It also considers information security from the viewpoint of a number of different interest groups, such as companies, consumers, criminals, police, and spies. This material is drawn from my senior course on security, from research work, and from experience consulting.
- The third part looks at the organizational and policy issues: how computer security interacts with law, with evidence, and with corporate politics; how we can gain confidence that a system will perform as intended; and how the whole business of security engineering can best be managed.

I believe that building systems that continue to perform robustly in the face of malice is one of the most important, interesting, and difficult tasks facing engineers in the twenty-first century.

Ross Anderson
Cambridge, January 2001

CHAPTER

3

Protocols

It is impossible to foresee the consequences of being clever.

— Christopher Strachey

Every thing secret degenerates, even the administration of justice; nothing is safe that does not show how it can bear discussion and publicity.

— Lord Acton

3.1 Introduction

If security engineering has a deep unifying theme, it is the study of security protocols. We've come across a few protocols informally already — I've mentioned challenge-response authentication and Kerberos. In this chapter, I'll dig down into the details. Rather than starting off with a formal definition of a security protocol, I will give a rough indication and then refine it using a number of examples. As this is an engineering book, I will also give many examples of how protocols fail.

A typical security system consists of a number of principals such as people, companies, computers and magnetic card readers, which communicate using a variety of channels including phones, email, radio, infrared, and by carrying data on physical devices such as bank cards and transport tickets. The security protocols are the rules that govern these communications. They are typically designed so that the system will survive malicious acts such as people telling lies on the phone, hostile governments jamming radio, or forgers altering the data on train tickets. Protection against all possible attacks is often too expensive, so protocols are typically designed under certain assumptions about the threats. For example, the logon protocol that consists of a user

Plaid Ex1003
Page 186 of 372

entering a password into a machine assumes that she can enter it into the right machine. In the old days of hard-wired terminals in the workplace, this was reasonable; now that people log on to websites over the Internet, it is much less so. Evaluating a protocol thus involves answering two questions: first, is the threat model realistic? Second, does the protocol deal with it?

Protocols may be extremely simple, such as swiping a badge through a reader in order to enter a building. They often involve interaction, and do not necessarily involve technical measures like cryptography. For example, when we order a bottle of fine wine in a restaurant, the standard wine-waiter protocol provides some privacy (the other diners at our table don't learn the price), some integrity (we can be sure we got the right bottle and that it wasn't switched for, or refilled with, cheap plonk) and non-repudiation (it's hard for the diner to complain afterwards that the wine was off). Blaze gives other examples from applications as diverse as ticket inspection, aviation security and voting in [185].

At the technical end of things, protocols can be much more complex. The world's bank card payment system has dozens of protocols specifying how customers interact with cash machines and retail terminals, how a cash machine or terminal talks to the bank that operates it, how the bank communicates with the network operator, how money gets settled between banks, how encryption keys are set up between the various cards and machines, and what sort of alarm messages may be transmitted (such as instructions to capture a card). All these protocols have to work together in a large and complex system.

Often a seemingly innocuous design feature opens up a serious flaw. For example, a number of banks encrypted the customer's PIN using a key known only to their central computers and cash machines, and wrote it to the card magnetic strip. The idea was to let the cash machine verify PINs locally, which saved on communications and even allowed a limited service to be provided when the cash machine was offline. After this system had been used for many years without incident, a programmer (who was playing around with a card reader used in a building access control system) discovered that he could alter the magnetic strip of his own bank card by substituting his wife's bank account number for his own. He could then take money out of her account using the modified card and his own PIN. He realised that this enabled him to loot any other customer's account too, and went on to steal hundreds of thousands over a period of years. The affected banks had to spend millions on changing their systems. And some security upgrades can take years; at the time of writing, much of Europe has moved from magnetic-strip cards to smartcards, while America has not. Old and new systems have to work side by side so that European cardholders can buy from American stores and vice versa. This also opens up opportunities for the crooks; clones of European cards are often used in magnetic-strip cash machines in other countries, as the two systems' protection mechanisms don't quite mesh.

So we need to look systematically at security protocols and how they fail. As they are widely deployed and often very badly designed, I will give a number of examples from different applications.

3.2 Password Eavesdropping Risks

Passwords and PINs are still the foundation on which much of computer security rests, as they are the main mechanism used to authenticate humans to machines. I discussed their usability and 'human interface' problems of passwords in the last chapter. Now let us consider some more technical attacks, of the kind that we have to consider when designing more general protocols that operate between one machine and another. A good case study comes from simple embedded systems, such as the remote control used to open your garage or to unlock the doors of cars manufactured up to the mid-1990's. These primitive remote controls just broadcast their serial number, which also acts as the password.

An attack that became common was to use a 'grabber', a device that would record a code broadcast locally and replay it later. These devices, seemingly from Taiwan, arrived on the market in about 1995; they enabled thieves lurking in parking lots to record the signal used to lock a car door and then replay it to unlock the car once the owner had left[1].

One countermeasure was to use separate codes for lock and unlock. But this is still not ideal. First, the thief can lurk outside your house and record the unlock code before you drive away in the morning; he can then come back at night and help himself. Second, sixteen-bit passwords are too short. It occasionally happened that people found they could unlock the wrong car by mistake (or even set the alarm on a car whose owner didn't know he had one [217]). And by the mid-1990's, devices appeared which could try all possible codes one after the other. A code will be found on average after about 2^{15} tries, which at ten per second takes under an hour. A thief operating in a parking lot with a hundred vehicles within range would be rewarded in less than a minute with a car helpfully flashing its lights.

So another countermeasure was to double the length of the password from 16 to 32 bits. The manufacturers proudly advertised 'over 4 billion codes'. But this only showed they hadn't really understood the problem. There was still

[1]With garage doors it's even worse. A common chip is the Princeton PT2262, which uses 12 tri-state pins to encode 3^{12} or 531,441 address codes. However implementers often don't read the data sheet carefully enough to understand tri-state inputs and treat them as binary instead, getting 2^{12}. Many of them only use eight inputs, as the other four are on the other side of the chip. And as the chip has no retry-lockout logic, an attacker can cycle through the combinations quickly and open your garage door after 2^7 attempts on average.

only one code (or two codes) for each car, and although guessing was now impractical, grabbers still worked fine.

Using a serial number as a password has a further vulnerability: there may be many people with access to it. In the case of a car, this might mean all the dealer staff, and perhaps the state motor vehicle registration agency. Some burglar alarms have also used serial numbers as master passwords, and here it's even worse: the serial number may appear on the order, the delivery note, the invoice and all the other standard commercial paperwork.

Simple passwords are sometimes the appropriate technology, even when they double as serial numbers. For example, my monthly season ticket for the swimming pool simply has a barcode. I'm sure I could make a passable forgery with our photocopier and laminating machine, but as the turnstile is attended and the attendants get to know the 'regulars', there is no need for anything more expensive. My card keys for getting into the laboratory where I work are slightly harder to forge: the one for student areas uses an infrared barcode, while the card for staff areas has an RFID chip that states its serial number when interrogated over short-range radio. Again, these are probably quite adequate — our more expensive equipment is in rooms with fairly good mechanical door locks. But for things that lots of people want to steal, like cars, a better technology is needed. This brings us to cryptographic authentication protocols.

3.3 Who Goes There? — Simple Authentication

A simple example of an authentication device is an infrared token used in some multistorey parking garages to enable subscribers to raise the barrier. This first transmits its serial number and then sends an authentication block consisting of the same serial number, followed by a random number, all encrypted using a key which is unique to the device. We will postpone discussion of how to encrypt data and what properties the cipher should have; we will simply use the notation $\{X\}_K$ for the message X encrypted under the key K.

Then the protocol between the access token in the car and the parking garage can be written as:

$$T \rightarrow G : T, \{T, N\}_{KT}$$

This is the standard protocol engineering notation, and can be a bit confusing at first, so we'll take it slowly.

The in-car token sends its name T followed by the encrypted value of T concatenated with N, where N stands for 'number used once', or *nonce*. Everything within the braces is encrypted, and the encryption binds T and N together as well as obscuring their values. The purpose of the nonce is to assure the recipient that the message is *fresh*, that is, it is not a replay of

an old message that an attacker observed. Verification is simple: the parking garage server reads T, gets the corresponding key KT, deciphers the rest of the message, checks that the nonce N has not been seen before, and finally that the plaintext contains T (which stops a thief in a car park from attacking all the cars in parallel with successive guessed ciphertexts).

One reason many people get confused is that to the left of the colon, T identifies one of the principals (the token which represents the subscriber) whereas to the right it means the name (that is, the serial number) of the token. Another is that once we start discussing attacks on protocols, we can suddenly start finding that the token T's message intended for the parking garage G was actually intercepted by the freeloader F and played back at some later time. So the notation is unfortunate, but it's too well entrenched now to change easily. Professionals often think of the $T \rightarrow G$ to the left of the colon is simply a hint as to what the protocol designer had in mind.

The term *nonce* can mean anything that guarantees the freshness of a message. A nonce can, according to the context, be a random number, a serial number, a random challenge received from a third party, or even a timestamp. There are subtle differences between these approaches, such as in the level of resistance they offer to various kinds of replay attack, and they increase system complexity in different ways. But in very low-cost systems, the first two predominate as it tends to be cheaper to have a communication channel in one direction only, and cheap devices usually don't have clocks.

Key management in such devices can be very simple. In a typical garage token product, each token's key is simply its serial number encrypted under a global master key KM known to the central server:

$$KT = \{T\}_{KM}$$

This is known as *key diversification*. It's a common way of implementing access tokens, and is very widely used in smartcard-based systems as well. But there is still plenty of room for error. One old failure mode that seems to have returned is for the serial numbers not to be long enough, so that someone occasionally finds that their remote control works for another car in the car park as well. Having 128-bit keys doesn't help if the key is derived by encrypting a 16-bit serial number.

Weak ciphers also turn up. One token technology used by a number of car makers in their door locks and immobilisers employs a block cipher known as Keeloq, which was designed in the late 1980s to use the minimum number of gates; it consists of a large number of iterations of a simple round function. However in recent years an attack has been found on ciphers of this type, and it works against Keeloq; it takes about an hour's access to your key to collect enough data for the attack, and then about a day on a PC to process it and recover the embedded cryptographic key [172]. You might not think this a practical attack, as someone who gets access to your key can just drive off with

your car. However, in some implementations, there is also a terrible protocol vulnerability, in that the key diversification is not done using the block cipher itself, but using exclusive-or: $KT = T \oplus KM$. So once you have broken a single vehicle key for that type of car, you can immediately work out the key for any other car of that type. The researchers who found this attack suggested 'Soon, cryptographers will drive expensive cars.'

Indeed protocol vulnerabilities usually give rise to more, and simpler, attacks than cryptographic weaknesses do. At least two manufacturers have made the mistake of only checking that the nonce is different from last time, so that given two valid codes A and B, the series $ABABAB...$ was interpreted as a series of independently valid codes. A thief could open a car by replaying the last-but-one code. A further example comes from the world of prepayment utility meters. Over a million households in the UK, plus many millions in developing countries, have an electricity or gas meter that accepts encrypted tokens; the householder buys a token, takes it home and inserts it into the meter, which then dispenses the purchased quantity of energy. One electricity meter widely used in South Africa checked only that the nonce in the decrypted command was different from last time. So the customer could charge the meter up to the limit by buying two low-value power tickets and then repeatedly feeding them in one after the other [59].

So the question of whether to use a random number or a counter is not as easy as it might seem [316]. If you use random numbers, the lock has to remember a reasonable number of past codes. You might want to remember enough of them to defeat the *valet attack*. Here, someone who has temporary access to the token — such as a valet parking attendant — can record a number of access codes and replay them later to steal your car. Providing enough nonvolatile memory to remember hundreds or even thousands of old codes might push you to a more expensive microcontroller, and add a few cents to the cost of your lock.

If you opt for counters, the problem is synchronization. The key may be used for more than one lock; it may also be activated repeatedly by jostling against something in your pocket (I once took an experimental token home where it was gnawed by my dogs). So there has to be a way to recover after the counter has been incremented hundreds or possibly even thousands of times. This can be turned to advantage by allowing the lock to 'learn', or synchronise on, a key under certain conditions; but the details are not always designed thoughtfully. One common product uses a sixteen bit counter, and allows access when the deciphered counter value is the last valid code incremented by no more than sixteen. To cope with cases where the token has been used more than sixteen times elsewhere (or gnawed by a family pet), the lock will open on a second press provided that the counter value has been incremented

between 17 and 32,767 times since a valid code was entered (the counter rolls over so that 0 is the successor of 65,535). This is fine in many applications, but a thief who can get six well-chosen access codes — say for values 0, 1, 20,000, 20,001, 40,000 and 40,001 — can break the system completely. So you would have to think hard about whether your threat model includes a valet able to get access codes corresponding to chosen counter values, either by patience or by hardware hacking.

A recent example of design failure comes from TinyOS, an operating system used in sensor networks based on the IEEE 802.15.4 ad-hoc networking standard. The TinySec library commonly used for security protocols contains not one, but three counters. The first is lost as the radio chip driver overwrites it, the second isn't remembered by the receiver, and although the third is functional, it's used for reliability rather than security. So if someone monkeys with the traffic, the outcome is 'error' rather than 'alarm', and the network will resynchronise itself on a bad counter [340].

So designing even a simple token authentication mechanism is not at all straightforward. There are many attacks that do not involve 'breaking' the encryption. Such attacks are likely to become more common as cryptographic authentication mechanisms proliferate, many of them designed by programmers who thought the problem was easy and never bothered to read a book like this one. And there are capable agencies trying to find ways to defeat these remote key entry systems; in Thailand, for example, Muslim insurgents use them to detonate bombs, and the army has responded by deploying jammers [1000].

Another important example of authentication, and one that's politically contentious for different reasons, is 'accessory control'. Many printer companies embed authentication mechanisms in printers to ensure that genuine toner cartridges are used. If a competitor's product is loaded instead, the printer may quietly downgrade from 1200 dpi to 300 dpi, or simply refuse to work at all. Mobile phone vendors make a lot of money from replacement batteries, and now use authentication protocols to spot competitors' products so they can be blocked or even drained more quickly. All sorts of other industries are getting in on the act; there's talk in the motor trade of cars that authenticate their major spare parts. I'll discuss this in more detail in Chapter 22 along with copyright and rights management generally. Suffice it to say here that security mechanisms are used more and more to support business models, by accessory control, rights management, product tying and bundling. It is wrong to assume blindly that security protocols exist to keep 'bad' guys 'out'. They are increasingly used to constrain the lawful owner of the equipment in which they are built; their purpose may be of questionable legality or contrary to public policy.

3.3.1 Challenge and Response

Most cars nowadays have remote-controlled door unlocking, though most also have a fallback metal key to ensure that you can still get into your car even if the RF environment is noisy. Many also use a more sophisticated two-pass protocol, called challenge-response, to actually authorise engine start. As the car key is inserted into the steering lock, the engine controller sends a challenge consisting of a random n-bit number to the key using short-range radio. The car key computes a response by encrypting the challenge. So, writing E for the engine controller, T for the transponder in the car key, K for the cryptographic key shared between the transponder and the engine controller, and N for the random challenge, the protocol may look something like:

$$\begin{aligned} E &\rightarrow T: \quad N \\ T &\rightarrow E: \quad \{T, N\}_K \end{aligned}$$

This is still not bulletproof.

In one system, the random numbers generated by the engine management unit turned out to be predictable, so it was possible for a thief to interrogate the key in the car owner's pocket, as he passed, with the anticipated next challenge. In fact, many products that incorporate encryption have been broken at some time or another because their random number generators weren't random enough [533, 395]. The fix varies from one application to another. It's possible to build hardware random number generators using radioactive decay, but this isn't common because of health and safety concerns. There are various sources of usable randomness in large systems such as PCs, such as the small variations in the rotation speed of the hard disk caused by air turbulence [358]. PC software products often mix together the randomness from a number of environmental sources such as network traffic and keystroke timing and from internal system sources [567]; and the way these sources are combined is often critical [703]. But in a typical embedded system such as a car lock, the random challenge is generated by encrypting a counter using a special key which is kept inside the device and not used for any other purpose.

Locks are not the only application of challenge-response protocols. In HTTP Digest Authentication, a web server challenges a client or proxy, with whom it shares a password, by sending it a nonce. The response consists of the hash of the nonce, the password, and the requested URI [493]. This provides a mechanism that's not vulnerable to password snooping. It's used, for example, to authenticate clients and servers in SIP, the protocol for Voice-Over-IP (VOIP) telephony. It is much better than sending a password in the clear, but suffers from various weaknesses — the most serious being middleperson attacks, which I'll discuss shortly.

A much more visible use of challenge-response is in *two-factor authentication*. Many organizations issue their staff with password generators to let them log on to corporate computer systems [1354]. These may look like calculators (and some even function as calculators) but their main function is as follows. When you want to log in to a machine on the network, you call up a logon screen and are presented with a random challenge of maybe seven digits. You key this into your password generator, together with a PIN of maybe four digits. The device encrypts these eleven digits using a secret key shared with the corporate security server, and displays the first seven digits of the result. You enter these seven digits as your password. This protocol is illustrated in Figure 3.1. If you had a password generator with the right secret key, and you entered the PIN right, and you typed in the result correctly, then the corporate computer system lets you in. But if you do not have a genuine password generator for which you know the PIN, your chance of logging on is small.

Formally, with S for the server, P for the password generator, PIN for the user's Personal Identification Number that bootstraps the password generator, U for the user and N for the random nonce:

$$\begin{aligned} S &\rightarrow U: & N \\ U &\rightarrow P: & N, PIN \\ P &\rightarrow U: & \{N, PIN\}_K \\ U &\rightarrow S: & \{N, PIN\}_K \end{aligned}$$

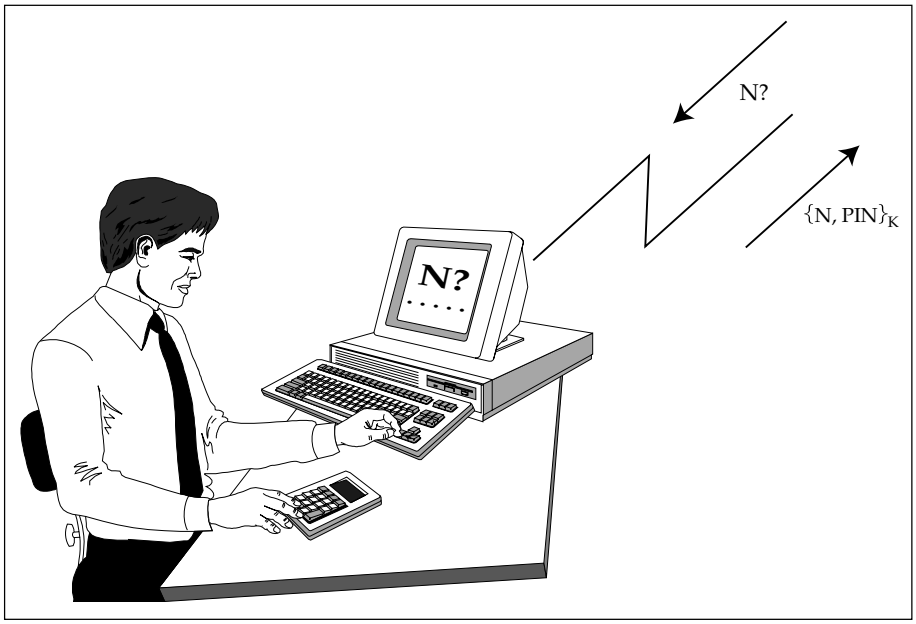


Figure 3.1: Password generator use

These devices appeared from the early 1980s and caught on first with phone companies, then in the 1990s with banks for use by staff. There are simplified versions that don't have a keyboard, but just generate a new access code every minute or so by encrypting a counter: the RSA SecurID is the best known. One sector after another has been adopting authentication tokens of one kind or another to replace or supplement passwords; the US Defense Department announced in 2007 that the introduction of an authentication system based on the DoD Common Access Card had cut network intrusions by 46% in the previous year [225].

The technology is now starting to spread to the customer side of things. By 2001, password generators were used by some exclusive private banks, such as Coutts, to authenticate their online customers. These banks never suffered any phishing fraud. By 2006, some banks in the Netherlands and Scandinavia had rolled out the technology to all their millions of customers; then the frauds started. The phishermen typically use real-time man-in-the-middle attacks (which I'll describe in the next section) to take over a session once the user has authenticated herself to the bank. As of late 2007, some banks in the UK and elsewhere in Europe have been introducing the Chip Authentication Program (CAP), which is implemented by giving bank customers a calculator that uses their bank card to do crypto[2]. This calculator, when loaded with a bank card, will ask for the customer's PIN and, if it's entered correctly, will compute a response code based on either a counter (as a one-off authentication code for a card transaction, or a one-step logon to a banking website) or a challenge (for a two-step logon). There is also a third mode of operation: if session takeover becomes a problem, the CAP calculator can also be used to authenticate transaction data. In this case, it's planned to have the customer enter the amount and the last eight digits of the payee account number into her CAP calculator.

But the result might not be as good in banking as it has been in the armed forces. First, when your wallet is stolen the thief might be able to read your PIN digits from the calculator — they will be the dirty and worn keys. If you just use one bank card, then the thief's chance of guessing your PIN in 3 tries has just come down from about 1 in 3000 to about 1 in 10. Second, when you use your card in a Mafia-owned shop (or in a shop whose terminals have been quietly reprogrammed without the owner's knowledge), the bad guys have everything they need to loot your account. Not only that — they can compute a series of CAP codes to give them access in the future, and use your account for wicked purposes such as money laundering. Third, someone who takes your bank card from you at knifepoint can now verify that you've told them

[2]Bank cards in many European countries have an EMV smartcard chip on them, and new UK bank cards have software to compute authentication codes as well as to operate ATMs and shop terminals.

the right PIN. A further problem is that the mechanisms can be used in a range of protocols; if you have to give a one-off authentication code over the phone to buy a book with your bank card, and the bookseller can then use that code to log on to your bank, it's clearly a bad thing. A deeper problem is that once lots of banks use one-time passwords, the phishermen will just rewrite their scripts to do real-time man-in-the-middle attacks. These have already been used against the early adopter banks in the Netherlands and Scandinavia. To see how they work, we will now look at a military example.

3.3.2 The MIG-in-the-Middle Attack

The ever-increasing speeds of warplanes in the 1930s and 1940s, together with the invention of the jet engine, radar and rocketry, made it ever more difficult for air defence forces to tell their own craft apart from the enemy's. This led to a serious risk of 'fratricide' — people shooting down their colleagues by mistake — and drove the development of systems to 'identify-friend-or-foe' (IFF). These were first fielded in World War II, and in their early form enabled an airplane illuminated by radar to broadcast an identifying number to signal friendly intent. In 1952, this system was adopted to identify civil aircraft to air traffic controllers and, worried about the loss of security once it became widely used, the U.S. Air Force started a research programme to incorporate cryptographic protection in the system. Nowadays, the typical air defense system sends random challenges with its radar signals, and friendly aircraft have equipment and keys that enable them to identify themselves with correct responses. The chapter on electronic warfare has more details on modern systems.

It's tricky to design a good IFF system. One of the problems is illustrated by the following story, which I heard from an officer in the South African Air Force (SAAF). After it was published in the first edition of this book, the story was disputed — as I'll discuss below. Be that as it may, similar games have been played with other electronic warfare systems since World War 2. The 'Mig-in-the-middle' story has in any event become part of the folklore, and it nicely illustrates how attacks can be carried out in real time on challenge-response authentication protocols.

In the late 1980's, South African troops were fighting a war in northern Namibia and southern Angola. The goals were to keep Namibia under white rule, and impose a client government (UNITA) on Angola. Because the South African Defence Force consisted largely of conscripts from a small white population, it was important to limit casualties, so most South African soldiers remained in Namibia on policing duties while the fighting to the north was done by UNITA troops. The role of the SAAF was twofold: to provide tactical support to UNITA by bombing targets in Angola, and to ensure that the Angolans and their Cuban allies did not return the compliment in Namibia.

Suddenly, the Cubans broke through the South African air defenses and carried out a bombing raid on a South African camp in northern Namibia, killing a number of white conscripts. This proof that their air supremacy had been lost helped the Pretoria government decide to hand over Namibia to the insurgents — itself a huge step on the road to majority rule in South Africa several years later. The raid may also have been the last successful military operation ever carried out by Soviet bloc forces.

Some years afterwards, a SAAF officer told me how the Cubans had pulled it off. Several MIGs had loitered in southern Angola, just north of the South African air defense belt, until a flight of SAAF Impala bombers raided a target in Angola. Then the MIGs turned sharply and flew openly through the SAAF's air defenses, which sent IFF challenges. The MIGs relayed them to the Angolan air defense batteries, which transmitted them at a SAAF bomber; the responses were relayed back in real time to the MIGs, who retransmitted them and were allowed through — as in Figure 3.2. According to my informant, this had a significant effect on the general staff in Pretoria. Being not only outfought by black opponents, but actually outsmarted, was not consistent with the world view they had held up till then.

After this tale was published in the first edition of my book, I was contacted by a former officer in SA Communications Security Agency who disputed the story's details. He said that their IFF equipment did not use cryptography yet at the time of the Angolan war, and was always switched off over enemy territory. Thus, he said, any electronic trickery must have been of a more primitive kind. However, others tell me that 'Mig-in-the-middle' tricks were significant in Korea, Vietnam and various Middle Eastern conflicts.

In any case, the tale illustrates the basic idea behind an attack known to the cryptographic community as the *man-in-the-middle* or (more recently) the *middleperson* attack. It applies in a straightforward way to the challenge-response authentication performed by password calculators: the phishing site invites the mark to log on and simultaneously opens a logon session with his bank. The bank sends a challenge; the phisherman relays this to the mark, who uses his device to respond to it; the phisherman relays it to the bank, and is now authenticated to the bank as the mark. This is why, as I discussed above, European banks are introducing not just a simple response to a single challenge, but an authentication code based on input fields such as the amount, the payee account number and a transaction sequence number.

However, once the protocol-level vulnerabilities are fixed by including all the transaction data, the big problem will be usability. If it takes two minutes and the entry of dozens of digits to make a payment, then a lot of customers will get digits wrong, give up, and then either call the call center or send paper checks — undermining the cost savings of online banking. Also, the bad guys will be able to exploit the fallback mechanisms, perhaps by spoofing customers

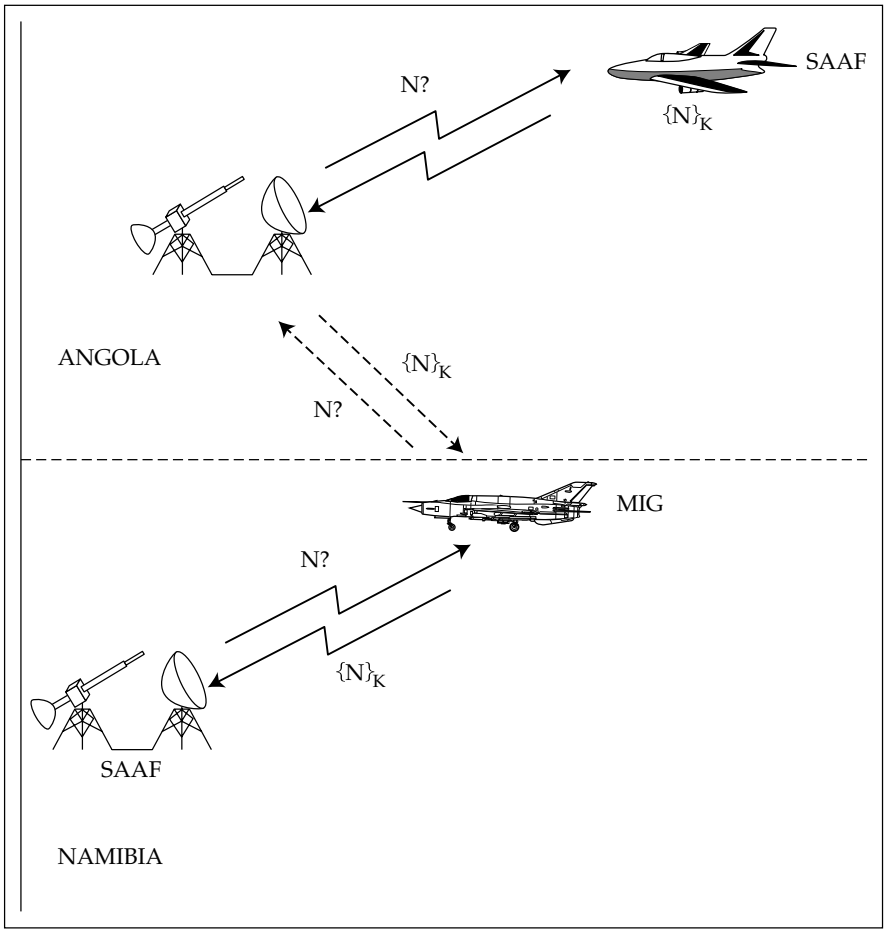


Figure 3.2: The MIG-in-the middle attack

into calling voice phishing phone numbers that run a middleperson attack between the customer and the call center.

We will come across the man-in-the-middle attack again and again in applications ranging from pay-TV to Internet security protocols. It even applies in online gaming. As the mathematician John Conway once remarked, it's easy to get at least a draw against a grandmaster at postal chess: just play two grandmasters at once, one as white and the other as black, and relay the moves between them!

In many cases, middleperson attacks are possible but not economic. In the case of car keys, it should certainly be possible to steal a car by having an accomplice follow the driver and electronically relay the radio challenge to you as you work the lock. (One of our students has actually demonstrated

this for our RFID door locks.) But, for the average car thief, it would be a lot simpler to just pick the target's pocket or mug him.

In early 2007, it became clear that there is a practical middleperson attack on the protocols used by the EMV smartcards issued to bank customers in Europe. A bad man could build a wicked terminal that masqueraded, for example, as a parking meter; when you entered your card and PIN to pay a £2.50 parking fee, the transaction could be relayed to a crook loitering near a self-service terminal in a hardware store, who would use a card emulator to order goods. When you get your statement, you might find you've been debited £2,500 for a wide-screen TV [915]. The basic problem here is the lack of a trustworthy user interface on the card; the cardholder doesn't really know which terminal his card is doing business with. I'll discuss such attacks further in the chapter on Banking and Bookkeeping.

3.3.3 Reflection Attacks

Further interesting problems arise with mutual authentication, that is, when two principals have to identify each other. Suppose, for example, that a simple challenge-response IFF system designed to prevent anti-aircraft gunners attacking friendly aircraft had to be deployed in a fighter-bomber too. Now suppose that the air force simply installed one of their air gunners' challenge units in each aircraft and connected it to the fire-control radar. But now an enemy bomber might reflect a challenge back at our fighter, get a correct response, and then reflect that back as its own response:

$$F \longrightarrow B : N$$

$$B \longrightarrow F : N$$

$$F \longrightarrow B : \{N\}_K$$

$$B \longrightarrow F : \{N\}_K$$

So we will want to integrate the challenge system with the response generator. It is still not enough just for the two units to be connected and share a list of outstanding challenges, as an enemy attacked by two of our aircraft might reflect a challenge from one of them to be answered by the other. It might also not be acceptable to switch manually from 'attack' to 'defense' during combat.

There are a number of ways of stopping this 'reflection attack': in many cases, it is sufficient to include the names of the two parties in the authentication exchange. In the above example, we might require a friendly bomber to reply to the challenge:

$$F \longrightarrow B : N$$

with a response such as:

$$B \longrightarrow F : \{B, N\}_K$$

Thus a reflected response $\{F, N\}$ (or even $\{F', N\}$ from the fighter pilot's wingman) could be detected.

This is a much simplified account of IFF, but it serves to illustrate the subtelty of the trust assumptions that underlie an authentication protocol. If you send out a challenge N and receive, within 20 milliseconds, a response $\{N\}_K$, then — since light can travel a bit under 3,730 miles in 20 ms — you know that there is someone with the key K within 2000 miles. But that's all you know. If you can be sure that the response was not computed using your own equipment, you now know that there is someone *else* with the key K within two thousand miles. If you make the further assumption that all copies of the key K are securely held in equipment which may be trusted to operate properly, and you see $\{B, N\}_K$, you might be justified in deducing that the aircraft with callsign B is within 2000 miles. A clear understanding of trust assumptions and their consequences is at the heart of security protocol design.

By now you might think that the protocol design aspects of IFF have been exhaustively discussed. But we've omitted one of the most important problems — and one which the designers of early IFF systems did not anticipate. As radar returns are weak, the signal from the IFF transmitter on board an aircraft will often be audible at a much greater range than the return. The Allies learned this the hard way; in January 1944, decrypts of Enigma messages revealed that the Germans were plotting British and American bombers at twice the normal radar range by interrogating their IFF. So many modern systems authenticate the challenge as well as the response. The NATO mode XII, for example, has a 32 bit encrypted challenge, and a different valid challenge is generated for every interrogation signal, of which there are typically 250 per second. Theoretically there is no need to switch off over enemy territory, but in practice an enemy who can record valid challenges can replay them as part of an attack. Relays are also possible, as with the Mig in the middle.

Many other IFF design problems are less protocol-related, such as the difficulties posed by neutrals, error rates in dense operational environments, how to deal with equipment failure, how to manage keys, and how to cope with multinational coalitions such as that put together for Operation Desert Storm. I'll return to IFF in Chapter 19. For now, the spurious-challenge problem serves to reinforce an important point: that the correctness of a security protocol depends on the assumptions made about the requirements. A protocol that can protect against one kind of attack (being shot down by your own side) but which increases the exposure to an even more likely attack (being shot down by the other side) does more harm than good. In fact, the spurious-challenge problem became so serious in World War II that some experts advocated abandoning IFF altogether, rather than taking the risk that one bomber pilot

in a formation of hundreds would ignore orders and leave his IFF switched on while over enemy territory.

3.4 Manipulating the Message

We've now seen a number of middleperson attacks that reflect or spoof the information used to authenticate a participant's identity — from ATM cards that could be reprogrammed to 'identify' the wrong customer, to attacks on IFF. However, there are more complex attacks where the attacker does not just obtain false identification, but manipulates the message content in some way.

An example is when dishonest cabbies insert pulse generators in the cable that connects their taximeter to a sensor in their taxi's gearbox. The sensor sends pulses as the prop shaft turns, which lets the meter work out how far the taxi has gone. A pirate device, which inserts extra pulses, makes the taxi appear to have gone further. We'll discuss such attacks at much greater length in the chapter on 'Monitoring Systems', in section 12.3.

Another example is a key log attack which defeated many pay-TV systems in Europe in the 1990s and still appears to work in China. The attack is also known as *delayed data transfer*, or DDT. First-generation pay-TV equipment has a decoder, which deciphers the video signal, and a customer smartcard which generates the deciphering keys. These keys are recomputed every few hundred milliseconds by using a one-way encryption function applied to various 'entitlement control messages' that appear in the signal. Such systems can be very elaborate (and we'll discuss some more complex attacks on them later) but there is a very simple attack which works against a lot of them. If the messages that pass between the smartcard and the decoder are the same for all decoders (which is usually the case) then a subscriber can log all the keys sent by his card to his decoder and post it online somewhere. People without a subscription, but who have video-recorded the enciphered program, can then download the key log and use it to decipher the tape.

Changing pay-TV protocols to prevent DDT attacks can be difficult. The base of installed equipment is huge, and many of the obvious countermeasures have an adverse effect on legitimate customers (such as by preventing them videotaping movies). Pay-TV companies generally ignore this attack, since connecting a PC up to a satellite TV decoder through a special hardware adaptor is something only hobbyists do; it is too inconvenient to be a real threat to their revenue stream. In the rare cases where it becomes a nuisance, the strategy is usually to identify the troublesome subscribers and send entitlement control messages that deactivate their cards.

Message-manipulation attacks aren't limited to 'consumer' grade systems. The Intelsat satellites used for international telephone and data traffic have robust mechanisms to prevent a command being accepted twice — otherwise

an attacker could repeatedly order the same manoever to be carried out until the satellite ran out of fuel [1027].

3.5 Changing the Environment

A very common cause of protocol failure is that the environment changes, so that assumptions which were originally true no longer hold and the security protocols cannot cope with the new threats.

One nice example comes from the ticketing systems used by the urban transport authority in London. In the early 1980's, passengers devised a number of scams to cut the cost of commuting. For example, a passenger who commuted a long distance from a suburban station to downtown might buy two cheaper, short distance season tickets — one between his suburban station and a nearby one, and the other between his destination and another downtown station. These would let him get through the barriers, and on the rare occasions he was challenged by an inspector in between, he would claim that he'd boarded at a rural station which had a broken ticket machine.

A large investment later, the system had all the features necessary to stop such scams: all barriers were automatic, tickets could retain state, and the laws had been changed so that people caught without tickets got fined on the spot.

But suddenly the whole environment changed, as the national transport system was privatized to create dozens of rail and bus companies. Some of the new operating companies started cheating each other, and there was nothing the system could do about it! For example, when a one-day travel pass was sold, the revenue was distributed between the various bus, train and subway operators using a formula that depended on where it was sold. Suddenly, the train companies had a motive to book all their ticket sales through the outlet that let them keep the largest percentage. As well as bad outsiders (passengers), we now had bad insiders (rail companies), and the design just hadn't allowed for them. Chaos and litigation ensued.

The transport system's problem was not new; it had been observed in the Italian ski resort of Val di Fassa in the mid-1970's. There, one could buy a monthly pass for all the ski lifts in the valley. An attendant at one of the lifts was observed with a deck of cards, one of which he swiped through the reader between each of the guests. It turned out that the revenue was divided up between the various lift operators according to the number of people who had passed their turnstiles. So each operator sought to inflate its own figures as much as it could [1217].

Another nice example comes from the world of cash machine fraud. In 1993 and 1994, Holland suffered an epidemic of 'phantom withdrawals'; there was much controversy in the press, with the banks claiming that their systems were secure while many people wrote in to the papers claiming to have been

cheated. Eventually the banks were shamed into actively investigating the claims, and noticed that many of the victims had used their bank cards at a certain filling station near Utrecht. This was staked out and one of the staff was arrested. It turned out that he had tapped the line from the card reader to the PC that controlled it; his tap recorded the magnetic stripe details from their cards while he used his eyeballs to capture their PINs [33].

Why had the system been designed so badly? Well, when the standards for managing magnetic stripe cards and PINs were developed in the early 1980's by organizations such as IBM and VISA, the engineers had made two assumptions. The first was that the contents of the magnetic strip — the card number, version number and expiration date — were not secret, while the PIN was [880]. (The analogy used was that the magnetic strip was your name and the PIN your password. I will have more to say on the difficulties of naming below.) The second assumption was that bank card equipment would only be operated in trustworthy environments, such as in a physically robust automatic teller machine, or by a bank clerk at a teller station. So it was 'clearly' only necessary to encrypt the PIN, on its way from the PIN pad to the server; the magnetic strip data could be sent in clear from the card reader.

Both of these assumptions had changed by 1993. An epidemic of card forgery, mostly in the Far East in the late 1980's, drove banks to introduce authentication codes on the magnetic strips. Also, the commercial success of the bank card industry led banks in many countries to extend the use of debit cards from ATMs to terminals in all manner of shops. The combination of these two environmental changes undermined the original system design: instead of putting a card whose magnetic strip contained no security data into a trusted machine, people were putting a card with security data in clear on the strip into an untrusted machine. These changes had come about so gradually, and over such a long period, that the industry didn't see the problem coming.

3.6 Chosen Protocol Attacks

Some firms are trying to sell the idea of a 'multifunction smartcard' — an authentication device that could be used in a wide range of transactions to save you having to carry around dozens of different cards and keys. Governments keen to push ID cards in the wake of 9/11 have tried to get them used for many other transactions; some want a single card to be used for ID, banking and even transport ticketing. Singapore went so far as to experiment with a bank card that doubled as military ID. This introduced some interesting new risks: if a Navy captain tries to withdraw some cash from an ATM after a good dinner and forgets his PIN, will he be unable to take his ship to sea until Monday morning when they open the bank and give him his card back?

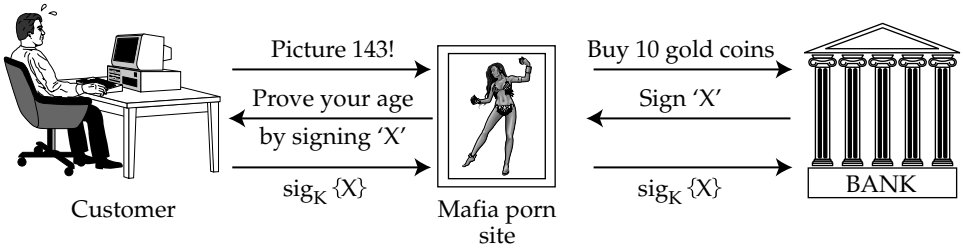


Figure 3.3: The Mafia-in-the-middle attack

Suppose that the banks in Europe were to introduce the CAP protocol to get their customers to authenticate themselves to electronic banking websites, but rather than forcing their customers to fiddle about with a calculator device they just issued all customers with smartcard readers that could be attached to their PC. This would certainly improve convenience and usability. You might think it would improve security too; the EMV protocol enables the card to calculate a message authentication code (MAC) on transaction data such as the amount, merchant number, date and transaction serial number. Message manipulation attacks against electronic banking payments would be prevented.

Or would they? The idea behind the 'Chosen Protocol Attack' is that given a target protocol, you design a new protocol that will attack it if the users can be inveigled into reusing the same token or crypto key. So how might the Mafia design a protocol to attack CAP?

Here's one approach. It used to be common for people visiting a porn website to be asked for 'proof of age,' which usually involves giving a credit card number, whether to the site itself or to an age checking service. If credit and debit cards become usable in PCs, it would be natural for the porn site to ask the customer to authenticate a random challenge as proof of age. A porn site can then mount a 'Mafia-in-the-middle' attack as shown in Figure 3.3. They wait until an unsuspecting customer visits their site, then order something resellable (such as gold coins) from a dealer, playing the role of the coin dealer's customer. When the coin dealer sends them the transaction data for authentication, they relay it through their porn site to the waiting customer. The poor man OKs it, the Mafia gets the gold coins, and when thousands of people suddenly complain about the huge charges to their cards at the end of the month, the porn site has vanished — along with the gold [702].

This is a more extreme variant on the Utrecht scam, and in the 1990s a vulnerability of this kind found its way into international standards: the standards for digital signature and authentication could be run back-to-back in this way. It has since been shown that many protocols, though secure in themselves, can be broken if their users can be inveigled into reusing the same keys in other applications [702]. This is why, for CAP to be secure, it may

Plaid Ex1003
Page 204 of 372

well have to be implemented in a stand-alone device into which the customer enters all the transaction parameters directly. Even so, some way has to be found to make it hard for the phishermen to trick the customer into computing an authentication code on data that they supply to the victim. The use of the customer's bank card in the CAP calculator may at least help to bring home that a banking transaction is being done.

In general, using crypto keys (or other authentication mechanisms) in more than one application is dangerous, while letting other people bootstrap their own application security off yours can be downright foolish. If a bank lets its smartcards also be used to load credit into prepayment electricity meters, it would have to worry very hard about whether bad software could be used in electricity vending stations (or even electricity meters) to steal money. Even if those risks could be controlled somehow, liability issues can arise from unplanned or emergent dependencies. A bank that changed its card specification might break the metering system — leaving its customers literally in the dark and risking a lawsuit from the power company. If the bank heeds these risks and tests system changes properly with all the dependant systems, then changes will be much more expensive. Crooks who hack the bank could black out the neighbourhood. The bank might still want to take this risk, though, reckoning that power company customers would be locked in more tightly to the bank, enabling it to charge them more. Security dependencies can have all sorts of strange effects, and we will return to this subject again and again later.

3.7 Managing Encryption Keys

The examples of security protocols that we have discussed so far are mostly about authenticating a principal's name, or application data such as the impulses driving a taximeter. There is one further class of authentication protocols that is very important — the protocols used to manage cryptographic keys. Until recently, such protocols were largely used in the background to support other operations; much of the technology was developed to manage the keys used by cash machines and banks to communicate with each other. But now, systems such as pay-TV use key management to control access to the system directly.

Authentication protocols are now also used in distributed computer systems for general key management purposes, and are therefore becoming ever more important. Kerberos was the first such system to come into widespread use, and a variant of it is used in Windows. I'll now lay the foundations for an understanding of Kerberos.

3.7.1 Basic Key Management

The basic idea behind key distribution protocols is that where two principals want to communicate, they may use a trusted third party to effect an introduction.

When discussing authentication protocols, it is conventional to give the principals human names in order to avoid getting lost in too much algebraic notation. So we will call the two communicating principals 'Alice' and 'Bob', and the trusted third party 'Sam'. But please don't assume that we are talking about human principals. Alice and Bob are likely to be programs while Sam is a server; for example, Alice might be a program in a taximeter, Bob the program in a gearbox sensor and Sam the computer at the taxi inspection station.

Anyway, a simple authentication protocol could run as follows.

1. Alice first calls Sam and asks for a key for communicating with Bob.
2. Sam responds by sending Alice a pair of certificates. Each contains a copy of a key, the first encrypted so only Alice can read it, and the second encrypted so only Bob can read it.
3. Alice then calls Bob and presents the second certificate as her introduction. Each of them decrypts the appropriate certificate under the key they share with Sam and thereby gets access to the new key. Alice can now use the key to send encrypted messages to Bob, and to receive messages from him in return.

Replay attacks are a known problem with authentication protocols, so in order that both Bob and Alice can check that the certificates are fresh, Sam may include a timestamp in each of them. If certificates never expire, there might be serious problems dealing with users whose privileges have been revoked.

Using our protocol notation, we could describe this as

$$\begin{aligned} A \to S: &\quad A, B \\ S \to A: &\quad \{A, B, K_{AB}, T\}_{K_{AS}}, \{A, B, K_{AB}, T\}_{K_{BS}} \\ A \to B: &\quad \{A, B, K_{AB}, T\}_{K_{BS}}, \{M\}_{K_{AB}} \end{aligned}$$

Expanding the notation, Alice calls Sam and says she'd like to talk to Bob. Sam makes up a session key message consisting of Alice's name, Bob's name, a key for them to use, and a timestamp. He encrypts all this under the key he shares with Alice, and he encrypts another copy of it under the key he shares with Bob. He gives both ciphertexts to Alice. Alice retrieves the key from the ciphertext that was encrypted to her, and passes on to Bob the ciphertext encrypted for him. She now sends him whatever message she wanted to send, encrypted using this key.

3.7.2 The Needham-Schroeder Protocol

Many things can go wrong, and here is a famous historical example. Many existing key distribution protocols are derived from the Needham-Schroeder protocol, which appeared in 1978 [960]. It is somewhat similar to the above, but uses nonces rather than timestamps. It runs as follows:

Message 1	$A \to S:$	A, B, N_A
Message 2	$S \to A:$	$\{N_A, B, K_{AB}, \{K_{AB}, A\}_{K_{BS}}\}_{K_{AS}}$
Message 3	$A \to B:$	$\{K_{AB}, A\}_{K_{BS}}$
Message 4	$B \to A:$	$\{N_B\}_{K_{AB}}$
Message 5	$A \to B:$	$\{N_B - 1\}_{K_{AB}}$

Here Alice takes the initiative, and tells Sam: 'I'm Alice, I want to talk to Bob, and my random nonce is N_A.' Sam provides her with a session key, encrypted using the key she shares with him. This ciphertext also contains her nonce so she can confirm it's not a replay. He also gives her a certificate to convey this key to Bob. She passes it to Bob, who then does a challenge-response to check that she is present and alert.

There is a subtle problem with this protocol — Bob has to assume that the key K_{AB} he receives from Sam (via Alice) is fresh. This is not necessarily so: Alice could have waited a year between steps 2 and 3. In many applications this may not be important; it might even help Alice to cache keys against possible server failures. But if an opponent — say Charlie — ever got hold of Alice's key, he could use it to set up session keys with many other principals.

Suppose, for example, that Alice had also asked for and received a key to communicate with Dave, and after Charlie stole her key he sent messages to Sam pretending to be Alice and got keys for Freddie and Ginger. He might also have observed message 2 in her protocol exchanges with Dave. So now Charlie could impersonate Alice to Dave, Freddie and Ginger. So when Alice finds out that her key has been stolen, perhaps by comparing message logs with Dave, she'd have to get Sam to contact everyone for whom she'd ever been issued a key, and tell them that her old key was no longer valid. She could not do this herself as she doesn't know anything about Freddie and Ginger. In other words, revocation is a problem: Sam may have to keep complete logs of everything he's ever done, and these logs would grow in size forever unless the principals' names expired at some fixed time in the future.

Almost 30 years later, this example still generates controversy in the security protocols community. The simplistic view is that Needham and Schroeder just got it wrong; the view argued by Susan Pancho and Dieter Gollmann (for which I have much sympathy) is that this is one more example of a protocol failure brought on by shifting assumptions [538, 1002]. 1978 was a kinder, gentler world; computer security then concerned itself with keeping 'bad guys' out, while nowadays we expect the 'enemy' to be the users of the

system. The Needham-Schroeder paper explicitly assumes that all principals behave themselves, and that all attacks come from outsiders [960]. With these assumptions, the protocol remains sound.

3.7.3 Kerberos

An important practical derivative of the Needham-Schroeder protocol may be found in Kerberos, a distributed access control system that originated at MIT and is now one of the standard authentication tools in Windows [1224]. Instead of a single trusted third party, Kerberos has two kinds: an authentication server to which users log on, and a ticket granting server which gives them tickets allowing access to various resources such as files. This enables more scalable access management. In a university, for example, one might manage students through their halls of residence but manage file servers by departments; in a company, the personnel people might register users to the payroll system while departmental administrators manage resources such as servers and printers.

First, Alice logs on to the authentication server using a password. The client software in her PC fetches a ticket from this server that is encrypted under her password and that contains a session key K_{AS}. Assuming she gets the password right, she now controls K_{AS} and to get access to a resource B controlled by the ticket granting server S, the following protocol takes place. Its outcome is a key K_{AB} with timestamp T_S and lifetime L, which will be used to authenticate Alice's subsequent traffic with that resource:

$$
\begin{array}{ll}
A \to S: & A, B \\
S \to A: & \{T_S, L, K_{AB}, B, \{T_S, L, K_{AB}, A\}_{K_{BS}}\}_{K_{AS}} \\
A \to B: & \{T_S, L, K_{AB}, A\}_{K_{BS}}, \{A, T_A\}_{K_{AB}} \\
B \to A: & \{T_A + 1\}_{K_{AB}}
\end{array}
$$

Translating this into English: Alice asks the ticket granting server for access to B. If this is permissible, the ticket $\{T_S, L, K_{AB}, A\}_{K_{BS}}$ is created containing a suitable key K_{AB} and given to Alice to use. She also gets a copy of the key in a form readable by her, namely encrypted under K_{AS}. She now verifies the ticket by sending a timestamp T_A to the resource, which confirms it's alive by sending back the timestamp incremented by one (this shows it was able to decrypt the ticket correctly and extract the key K_{AB}).

The vulnerability of Needham-Schroeder has been fixed by introducing timestamps rather than random nonces. But, as in most of life, we get little in security for free. There is now a new vulnerability, namely that the clocks on our various clients and servers might get out of synch; they might even be desynchronized deliberately as part of a more complex attack.

3.7.4 Practical Key Management

So we can use a protocol like Kerberos to set up and manage working keys between users given that each user shares one or more long-term keys with a server that acts as a key distribution centre. I'll describe a number of other similar protocols later; for example, in the chapter on 'Banking and Bookkeeping' I'll discuss how a bank can set up a long-term key with each of its ATMs and with each of the interbank networks with which it's associated. The bank then uses protocols not too unlike Kerberos to establish a 'key of the day' with each ATM and with each network switch; so when you turn up at the ATM belonging to a foreign bank and ask for money from your own bank via the Cirrus network, the ATM will encrypt the transaction using the working key it shares with the bank that owns it, and the bank will then pass on the transaction to Cirrus encrypted with the key of the day for that network.

So far so good. But a moment's thought will reveal that the bank has to maintain several keys for each of the several hundred ATMs that it owns — a long-term master key, plus perhaps an encryption key and an authentication key; several keys for each of the several dozen bank networks of which it's a member; passwords and other security information for each of several million electronic banking customers, and perhaps keys for them as well if they're given client software that uses cryptography. Oh, and there may be encrypted passwords for each of several thousand employees, which might also take the form of Kerberos keys encrypted under user passwords. That's a lot of key material. How is it to be managed?

Key management is a complex and difficult business and is often got wrong because it's left as an afterthought. A good engineer will sit down and think about how many keys are needed, how they're to be generated, how long they need to remain in service and how they'll eventually be destroyed. There is a much longer list of concerns — many of them articulated in the Federal Information Processing Standard for key management [948]. In addition, things go wrong as applications evolve; it's important to provide extra keys to support next year's functionality, so that you don't compromise your existing ones by reusing them in protocols that turn out to be incompatible. It's also important to support recovery from security failure. Yet there are no standard ways of doing either.

As for practical strategies, there are a number — none of them straightforward. Public-key crypto, which I'll discuss in Chapter 5, can slightly simplify the key management task. Long-term keys can be split into a private part and a public part; you don't have to keep the public part secret (as its name implies) but you do have to guarantee its integrity. In banking the usual answer is to use dedicated cryptographic processors called security modules, which I'll describe in detail in the chapter on 'Tamper Resistance'. These do all the cryptography and contain internal keys with which application keys are protected.

Thus you get your security module to generate master keys for each of your ATMs; you store their encrypted values in your ATM master file. Whenever a transaction comes in from that ATM, you retrieve the encrypted key from the file and pass it to the security module along with the encrypted data. The module then does what's necessary: it decrypts the PIN and verifies it, perhaps against an encrypted value kept locally. Unfortunately, the protocols used to set all this up are also liable to failure. Many attacks have been found that exploit the application programming interface, or API, of the security module, where these protocols are exposed. I will describe these attacks in detail in the chapter on API Security. For now, it's enough to note that getting security protocols right is hard. You should not design them at home, any more than you design your own explosives.

3.8 Getting Formal

Subtle difficulties of the kind we have seen with the above protocols, and the many ways in which protection properties depend on quite subtle starting assumptions that protocol designers may get wrong (or that may be misunderstood later), have led researchers to apply formal methods to key distribution protocols. The goal of this exercise was originally to decide whether a protocol was right or wrong: it should either be proved correct, or an attack should be exhibited. More recently this has expanded to clarifying the assumptions that underlie a given protocol.

There are a number of different approaches to verifying the correctness of protocols. The best known is the *logic of belief*, or *BAN logic*, named after its inventors Burrows, Abadi and Needham [249]. It reasons about what a principal might reasonably believe having seen of certain messages, timestamps and so on. A second is the *random oracle model*, which I touch on in the chapter on cryptology and which is favored by people working on the theory of cryptography; this appears less expressive than logics of belief, but can tie protocol properties to the properties of the underlying encryption algorithms. Finally, a number of researchers have applied mainstream formal methods such as CSP and verification tools such as Isabelle.

Some history exists of flaws being found in protocols that had been proved correct using formal methods; the following subsection offers a typical example.

3.8.1 A Typical Smartcard Banking Protocol

The COPAC system is an electronic purse used by VISA in countries with poor telecommunications [48]. It was the first live financial system whose underlying protocol suite was designed and verified using such formal techniques, and

in particular a variant of the BAN logic. A similar protocol is now used in the 'Geldkarte,' an electronic purse issued by banks in Germany, and adopted also by French banks as 'Moneo'. There's also a system in Belgium called 'Proton'. The European applications focus on low-value transactions with devices such as parking meters and vending machines for which it may not be economical to provide a network connection.

Transactions take place from a customer smartcard to a merchant smartcard (which in the case of a vending machine is kept in the machine and changed when it's replenished). The customer gives the merchant an electronic check with two authentication codes on it; one that can be checked by the network, and one that can be checked by the customer's bank. A simplified version of the protocol is as follows.

$$\begin{array}{ll} C \longrightarrow R: & \{C, N_C\}_K \\ R \longrightarrow C: & \{R, N_R, C, N_C\}_K \\ C \longrightarrow R: & \{C, N_C, R, N_R, X\}_K \end{array}$$

In English: the customer and the retailer share a key K. Using this key, the customer encrypts a message containing its account number C and a customer transaction serial number N_C. The retailer confirms its own account number R and his own transaction serial number N_R, as well as the information it's just received from the customer. The customer now sends the electronic check X, along with all the data exchanged so far in the protocol. One can think of the electronic check as being stapled to a payment advice with the customer's and retailer's account numbers and their respective reference numbers. (The reason for repeating all previous data in each message is to prevent message manipulation attacks using cut-and-paste.)

3.8.2 The BAN Logic

The BAN logic provides a formal method for reasoning about the beliefs of principals in cryptographic protocols. Its underlying idea is that we will believe that a message is authentic if it is encrypted with a relevant key and it is also fresh (that is, generated during the current run of the protocol). Further assumptions include that principals will only assert statements they believe in, and that some principals are authorities for certain kinds of statement. This is formalized using a notation which includes:

$A \mid\equiv X$ *A believes X*, or, more accurately, that A is entitled to believe X;

$A \mid\sim X$ *A once said X* (without implying that this utterance was recent or not);

$A \mid\Rightarrow X$ *A has jurisdiction over X*, in other words A is the authority on X and is to be trusted on it;

$A \triangleleft X$ *A sees X*, that is, someone sent a message to A containing X in such a way that he can read and repeat it;

$\sharp X$ *X is fresh*, that is, contains a current timestamp or some information showing that it was uttered by the relevant principal during the current run of the protocol;

$\{X\}_K$ *X encrypted under the key K*, as in the rest of this chapter;

$A \leftrightarrow^K B$ *A and B share the key K*, in other words it is an appropriate key for them to use to communicate.

There are further symbols dealing, for example, with public key operations and with passwords, that need not concern us here.

These symbols are manipulated using a set of postulates which include:

the message meaning rule states that if A sees a message encrypted under K, and K is a good key for communicating with B, then he will believe that the message was once said by B. (We assume that each principal can recognize and ignore his or her own messages.) Formally, $\dfrac{A \mid\equiv A \leftrightarrow^K B, A \triangleleft \{X\}_K}{A \mid\equiv B \mid\sim X}$

the nonce-verification rule states that if a principal once said a message, and the message is fresh, then that principal still believes it. Formally, $\dfrac{A \mid\equiv \sharp X, A \mid\equiv B \mid\sim X}{A \mid\equiv B \mid\equiv X}$

the jurisdiction rule states that if a principal believes something, and is an authority on the matter, then he or she should be believed. Formally, we write that $\dfrac{A \mid\equiv B \mid\Rightarrow X, A \mid\equiv B \mid\equiv X}{A \mid\equiv X}$

In this notation, the statements on the top are the conditions, and the one on the bottom is the result. There are a number of further rules to cover the more mechanical aspects of manipulation; for example, if A sees a statement then he sees its components provided he knows the necessary keys, and if part of a formula is known to be fresh, then the whole formula must be.

3.8.3 Verifying the Payment Protocol

Assuming that the key K is only available to principals who can be trusted to execute the protocol faithfully, formal verification is now straightforward. The trick is to start from the desired result and work backwards. In this case, we wish to prove that the retailer should trust the check, i.e., $R \mid\equiv X$ (the syntax of checks and cryptographic keys is similar for our purposes here; a check is good if and only if it is genuine and the date on it is sufficiently recent).

Now $R \mid\equiv X$ will follow under the jurisdiction rule from $R \mid\equiv C \mid\Rightarrow X$ (R believes C has jurisdiction over X) and $R \mid\equiv C \mid\equiv X$ (R believes C believes X).

The former condition follows from the hardware constraint, that no-one except C could have uttered a text of the form $\{C, \ldots\}_K$.

The latter, that $R \mid\equiv C \mid\equiv X$, must be deduced using the nonce verification rule from $\sharp X$ (X is fresh) and $R \mid\equiv C \mid\sim X$ (R believes C uttered X).

$\sharp X$ follows from its occurrence in $\{C, N_C, R, N_R, X\}_K$ which contains the sequence number N_R, while $R \mid\equiv C \mid\sim X$ follows from the hardware constraint.

The above summary of the proof is, of necessity, telegraphic. If you want to understand logics of authentication in detail, you should consult the original papers [48] and see the recommendations for further reading at the end of this chapter.

3.8.4 Limitations of Formal Verification

Formal methods can be an excellent way of finding bugs in security protocol designs as they force the designer to make everything explicit and thus confront difficult design choices that might otherwise be fudged. However, they have their limitations, too.

One problem is in the external assumptions we make. For example, we assumed that the key wasn't available to anyone who might use it in an unauthorized manner. In practice, this is not always true. Although our purse protocol is executed in tamper-resistant smartcards, their software can have bugs, and in any case the tamper-resistance they offer is never complete. (I'll discuss this in the chapter on Tamper Resistance.) So the system has various fallback mechanisms to detect and react to card forgery, such as shadow accounts which track the amount of money that should be on each card and which are updated as transactions are cleared. It also has lists of hot cards that are distributed to terminals; these are needed anyway for stolen cards, and can be used for forged cards too.

Second, there are often problems with the idealisation of the protocol. An interesting flaw was found in an early version of this system. The key K actually consisted of two keys — the encryption was done first with a 'transaction key' which was diversified (that is, each card had its own variant) and then again with a 'bank key', which was not diversified. The former was done by the network operator, and the latter by the bank which issued the card. The reasons for this included dual control, and to ensure that even if an attacker managed to drill the keys out of a single card, he would only be able to forge that card, not make forgeries which would pass as other cards (and thus defeat the hot card mechanism). But since the bank key was not diversified, it must be assumed to be known to any attacker who has broken a card. This means that he can undo the outer wrapping of encryption, and in some circumstances message replay was possible. (The bank key was diversified in a later version before any villains discovered and exploited the flaw.)

In this case there was no failure of the formal method, as no attempt was ever made to verify the diversification mechanism. But it does illustrate a common problem in security engineering — that vulnerabilities arise at the boundary between two protection technologies. In this case, there were three technologies: the hardware tamper resistance, the authentication protocol and the shadow account / hot card list mechanisms. Different protection technologies are often the domain of different experts who don't completely understand the assumptions made by the others. (In fact, that's one reason security engineers need a book such as this one: to help subject specialists understand each others' tools and communicate with each other more effectively.)

For these reasons, people have explored alternative ways of assuring the design of authentication protocols, including the idea of *protocol robustness*. Just as structured programming techniques aim to ensure that software is designed methodically and nothing of importance is left out, so robust protocol design is largely about explicitness. Robustness principles include that the interpretation of a protocol should depend only on its content, not its context; so everything of importance (such as principals' names) should be stated explicitly in the messages. There are other issues concerning the freshness provided by serial numbers, timestamps and random challenges, and on the way encryption is used. If the protocol uses public key cryptography or digital signature mechanisms, there are further more technical robustness issues.

3.9 Summary

Passwords are just one (simple) example of a more general concept, the security protocol. Protocols specify the series of steps that principals use to establish trust relationships in a system, such as authenticating a claim to identity, demonstrating ownership of a credential, or granting a claim on a resource. Cryptographic authentication protocols, whether one-pass (e.g., using random nonces) or two-pass (challenge-response) are used for a wide range of such purposes, from basic entity authentication to provide infrastructure for distributed systems that allows trust to be taken from where it exists to where it is needed. Security protocols are fielded in all sorts of systems from remote car door locks through military IFF systems to authentication in distributed computer systems.

It is difficult to design effective security protocols. They suffer from a number of potential problems, including middleperson attacks, modification attacks, reflection attacks, and replay attacks. These threats can interact with implementation vulnerabilities such as poor random number generators. Using mathematical techniques to verify the correctness of protocols can help, but it won't catch all the bugs. Some of the most pernicious failures are caused by creeping changes in the environment for which a protocol was designed, so that the protection it gives is no longer adequate.

Research Problems

At several times during the past 20 years, some people have thought that protocols had been 'done' and that we should turn to new research topics. They have been repeatedly proved wrong by the emergence of new protocol applications with a new crop of errors and attacks to be explored. Formal methods blossomed in the early 1990s, then key management protocols; during the mid-1990's the flood of proposals for electronic commerce mechanisms kept us busy; and in the later 1990's a whole series of mechanisms proposed for protecting copyright on the Internet provided us with targets. Since 2000, one strand of protocol research has acquired an economic flavour as security mechanisms are used more and more to support business models; the designer's 'enemy' is often a commercial competitor, or even the customer. Another has applied protocol analysis tools to look at the security of application programming interfaces (APIs), a topic to which I'll return later.

Will people continue to develop faulty protocols which other people attack, or will we manage to develop a methodology for designing them right first time? What are the exact uses and limitations of formal methods, and other mathematical approaches such as the random oracle model?

At the system level, how do we manage the tension between the principle that robust protocols are generally those in which everything is completely specified and checked (principals' names, roles, security policy statement, protocol version, time, date, sequence number, security context, maker of grandmother's kitchen sink) and the system engineering principle that a good specification should not overconstrain the implementer?

Further Reading

Research papers on security protocols are scattered fairly widely throughout the literature. The main introductory papers to read are probably the original Needham-Schroeder paper [960]; the Burrows-Abadi-Needham authentication logic [249]; papers by Abadi and Needham, and Anderson and Needham, on protocol robustness [2, 73]; and there is a survey paper by Anderson and Needham [74]. In [707] there is an analysis of a defective security protocol, carried out using three different formal methods. Beyond that, the proceedings of the security protocols workshops [290, 291] provide leads to current research, and there are many papers scattered around a wide range of conferences.

CHAPTER

10

Banking and Bookkeeping

The arguments of lawyers and engineers pass through one another like angry ghosts.

— Nick Bohm, Brian Gladman and Ian Brown [201]

Computers are not (yet?) capable of being reasonable any more than is a Second Lieutenant.

— Casey Schaufler

Against stupidity, the Gods themselves contend in vain.

— JC Friedrich von Schiller

10.1 Introduction

Banking systems range from cash machine networks and credit card processing, both online and offline, through high-value interbank money transfer systems, to the back-end bookkeeping systems that keep track of it all and settle up afterwards. There are specialised systems for everything from stock trading to bills of lading; and large companies have internal bookkeeping and cash management systems that duplicate many of the functions of a bank.

Such systems are important for a number of reasons. First, an understanding of transaction processing is a prerequisite for tackling the broader problems of electronic commerce and fraud. Many dotcom firms fell down badly on elementary bookkeeping; in the rush to raise money and build web sites, traditional business discipline was ignored. The collapse of Enron led to stiffened board-level accountability for internal control; laws such as

Plaid Ex1003
Page 216 of 372

Sarbanes-Oxley and Gramm-Leach-Bliley now drive much of the investment in information security. When you propose protection mechanisms to a client, one of the first things you're likely to be asked is the extent to which they'll help directors of the company discharge their fiduciary responsibilities to shareholders.

Second, bookkeeping was for many years the mainstay of the computer industry, with banking its most intensive application area. Personal applications such as web browsers and Office might now run on more machines, but accounting is still the critical application for the average business. So the protection of bookkeeping systems is of great practical importance. It also gives us a well-understood model of protection in which confidentiality plays little role, but where the integrity of records (and their immutability once made) is of paramount importance.

Third, transaction processing systems — whether for small debits such as $50 cash machine withdrawals, or multimillion dollar wire transfers — were the application that launched commercial cryptology. Banking applications drove the development not just of encryption algorithms and protocols, but also of the supporting technology such as tamper-resistant cryptographic processors. These processors provide an important and interesting example of a trusted computing base that is quite different from the hardened operating systems discussed in the context of multilevel security. Many instructive mistakes were first made (or at least publicly documented) in the area of commercial cryptography. The problem of how to interface crypto with access control was studied by financial cryptographers before any others in the open research community.

Finally, banking systems provide another example of multilateral security — but aimed at authenticity rather than confidentiality. A banking system should prevent customers from cheating each other, or the bank; it should prevent bank employees from cheating the bank, or its customers; and the evidence it provides should be sufficiently strong that none of these principals can get away with falsely accusing another principal of cheating.

In this chapter, I'll first describe the bookkeeping systems used to keep track of assets despite occasional corrupt staff; these are fairly typical of accounting systems used by other companies too. I'll then describe the banks' principal international funds-transfer systems; similar systems are used to settle securities transactions and to manage trade documents such as bills of lading. Next, I'll describe ATM systems, which are increasingly the public face of banking, and whose technology has been adopted in applications such as utility meters; and then I'll tell the story of credit cards, which have become the main payment mechanism online. I'll then move on to more recent technical advances, including the smartcards recently introduced in Europe, RFID credit

cards, and nonbank payment services such as PayPal. I'll wrap up with some points on money laundering, and what controls really work against fraud.

10.1.1 The Origins of Bookkeeping

Bookkeeping appears to have started in the Neolithic Middle East in about 8500 BC, just after the invention of agriculture [1122]. When people started to produce surplus food, they started to store and trade it. Suddenly they needed a way to keep track of which villager had put how much in the communal warehouse. To start with, each unit of food (sheep, wheat, oil, . . .) was represented by a clay token, or *bulla*, which was placed inside a clay envelope and sealed by rolling it with the pattern of the warehouse keeper. (See Figure 10.1.) When the farmer wanted to get his food back, the seal was broken by the keeper in the presence of a witness. (This may be the oldest known security protocol.) By about 3000BC, this had led to the invention of writing [1018]; after another thousand years, we find equivalents of promissory notes, bills of lading, and so on. At about the same time, metal ingots started to be used as an intermediate commodity, often sealed inside a bulla by an assayer. In 700BC, Lydia's King Croesus started stamping the metal directly and thus invented coins [1045]; by the Athens of Pericles, there were a number of wealthy individuals in business as bankers [531].



Figure 10.1: Clay envelope and its content of tokens representing 7 jars of oil, from Uruk, present day Iraq, ca. 3300 BC (Courtesy Denise Schmandt-Besserat and the Louvre Museum)

The next significant innovation dates to late medieval times. As the dark ages came to a close and trade started to grow, some businesses became too large for a single family to manage. The earliest of the recognisably modern banks date to this period; by having branches in a number of cities, they could finance trade efficiently. But as the economy grew, it was necessary to hire managers from outside, and the owner's family could not supervise them closely. This brought with it an increased risk of fraud, and the mechanism that evolved to control it was *double-entry bookkeeping*. People used to think this was invented in Italy sometime in the 1300s, though the first book on it did not appear until 1494, after the printing press came along [355]. Recently, however, historians have found double-entry records created by Jewish merchants in twelfth-century Cairo, and it's now believed that the Italians learned the technique from them [1140].

10.1.2 Double-Entry Bookkeeping

The idea behind double-entry bookkeeping is extremely simple, as with most hugely influential ideas. Each transaction is posted to two separate books, as a credit in one and a debit in the other. For example, when a firm sells a customer $100 worth of goods on credit, it posts a $100 credit on the Sales account, and a $100 debit onto the Receivables account. When the customer pays the money, it will credit the Receivables account (thereby reducing the asset of money receivable), and debit the Cash account. (The principle taught in accountancy school is 'debit the receiver, credit the giver'.) At the end of the day, the books should *balance*, that is, add up to zero; the assets and the liabilities should be equal. (If the firm has made some profit, then this is a liability to the shareholders.) In all but the smallest firms, the books will be kept by different clerks, and have to balance at the end of every month (at banks, every day).

By suitable design of the ledger system, we can see to it that each shop, or branch, can be balanced separately. Thus each cashier will balance her cash tray before locking it in the vault overnight; the debits in the cash legder should exactly balance the physical banknotes she's collected. So most frauds need the collusion of two or more members of staff; and this principle of *split responsibility*, also known as *dual control*, is complemented by audit. Not only are the books audited at year end, but there are random audits too; a team of inspectors may descend on a branch at no notice and insist that all the books are balanced before the staff go home.

10.1.3 A Telegraphic History of E-commerce

Many of the problems afflicting e-businesses stem from the popular notion that e-commerce is something completely new, invented in the mid-1990s. This is simply untrue.

Various kinds of visual signalling were deployed from classical times, including heliographs (which used mirrors to flash sunlight at the receiver), semaphones (which used the positions of moving arms to signal letters and numbers) and flags. Land-based systems sent messages along chains of beacon towers, and naval systems relayed them between ships. To begin with, their use was military, but after the Napoleonic War the French government opened its heliograph network to commercial use. Very soon the first frauds were carried out. For two years up till they were discovered in 1836, two bankers bribed an operator to signal the movements of the stock market to them covertly by making errors in transmissions that they could observe from a safe distance. Other techniques were devised to signal the results of horseraces. Various laws were passed to criminalise this kind of activity but they were ineffective. The only solution for the bookies was to 'call time' by a clock, rather than waiting for the result and hoping that they were the first to hear it.

From the 1760's to the 1840's, the electric telegraph was developed by a number of pioneers, of whom the most influential was Samuel Morse. He persuaded Congress in 1842 to fund an experimental line from Washington to Baltimore; this so impressed people that serious commercial investment started, and by the end of that decade there were 12,000 miles of line operated by 20 companies. This was remarkably like the Internet boom of the late 1990's.

Banks were the first big users of the telegraph, and they decided that they needed technical protection mechanisms to prevent transactions being altered by crooked operators en route. (I discussed the *test key* systems they developed for the purpose in section 5.2.4.) Telegrams were also used to create national markets. For the first time, commodity traders in New York could find out within minutes what prices had been set in auctions in Chicago, and fishing skippers arriving in Boston could find out the price of cod in Gloucester. The history of the period shows that most of the concepts and problems of e-commerce were familiar to the Victorians [1215]. How do you know who you're speaking to? How do you know if they're trustworthy? How do you know whether the goods will be delivered, and whether payments will arrive? The answers found in the nineteenth century involved intermediaries — principally banks who helped business manage risk using instruments such as references, guarantees and letters of credit.

10.2 How Bank Computer Systems Work

Banks were among the first large organizations to use computers for book-keeping. They started in the late 1950s and early 1960s with applications such as check processing, and once they found that even the slow and expensive computers of that era were much cheaper than armies of clerks, they proceeded to automate most of the rest of their back-office operations during the

1960s and 1970s. The 1960s saw banks offering automated payroll services to their corporate customers, and by the 1970s they were supporting business-to-business e-commerce based on *electronic data interchange* (EDI), whereby firms from General Motors to Marks and Spencer built systems that enabled them to link up their computers to their suppliers' so that goods could be ordered automatically. Travel agents built similar systems to order tickets in real time from airlines. ATMs arrived en masse in the 1970s, and online banking systems in the 1980s; web-based banking followed in the 1990s. Yet the fancy front-end systems still rely on traditional back-office automation for maintaining account data and performing settlement.

Computer systems used for bookkeeping typically claim to implement variations on the double-entry theme. But the quality of control is highly variable. The double-entry features may be just a skin in the user interface, while the underlying file formats have no integrity controls. And even if the ledgers are all kept on the same system, someone with root access — or with physical access and a debugging tool — may be able to change the records so that the balancing controls are bypassed. It may also be possible to evade the balancing controls in various ways; staff may notice bugs in the software and take advantage of them. Despite all these problems, the law in most developed countries requires companies to have effective internal controls, and makes the managers responsible for them. Such laws are the main drivers of investment in information security mechanisms, but they also a reason for much wasted investment. So we need to look at the mechanics of electronic bookkeeping in a more detail.

A typical banking system has a number of data structures. There is an *account master file* which contains each customer's current balance together with previous transactions for a period of perhaps ninety days; a number of *ledgers* which track cash and other assets on their way through the system; various *journals* which hold transactions that have been received from teller stations, cash machines, check sorters and so on, but not yet entered in the ledgers; and an *audit trail* that records which staff member did what and when.

The processing software that acts on these data structures will include a suite of overnight batch processing programs, which apply the transactions from the journals to the various ledgers and the account master file. The online processing will include a number of modules which post transactions to the relevant combinations of ledgers. So when a customer pays $100 into his savings account the teller will make a transaction which records a credit to the customer's savings account ledger of $100 while debiting the same amount to the cash ledger recording the amount of money in the drawer. The fact that all the ledgers should always add up to zero provides an important check; if the bank (or one of its branches) is ever out of balance, an alarm will go off and people will start looking for the cause.

The invariant provided by the ledger system is checked daily during the overnight batch run, and means that a programmer who wants to add to his own account balance will have to take the money from some other account, rather than just creating it out of thin air by tweaking the account master file. Just as in a traditional business one has different ledgers managed by different clerks, so in a banking data processing shop there are different programmers in charge of different subsystems. In addition, all code is subjected to scrutiny by an internal auditor, and to testing by a separate test department. Once it has been approved, it will be run on a production machine that does not have a development environment, but only approved object code and data.

10.2.1 The Clark-Wilson Security Policy Model

Although such systems had been in the field since the 1960s, a formal model of their security policy was only introduced in 1987, by Dave Clark and Dave Wilson (the former a computer scientist, and the latter an accountant) [295]. In their model, some data items are constrained so that they can only be acted on by a certain set of transformation procedures.

More formally, there are special procedures whereby data can be input — turned from an *unconstrained data item*, or UDI, into a *constrained data item*, or CDI; *integrity verification procedures* (IVP's) to check the validity of any CDI (e.g., that the books balance); and *transformation procedures* (TPs) which may be thought of in the banking case as transactions which preserve balance. In the general formulation, they maintain the integrity of CDIs; they also write enough information to an append-only CDI (the audit trail) for transactions to be reconstructed. Access control is by means of triples *(subject, TP, CDI)*, which are so structured that a dual control policy is enforced. In the formulation in [27]:

1. the system will have an IVP for validating the integrity of any CDI;
2. the application of a TP to any CDI must maintain its integrity;
3. a CDI can only be changed by a TP;
4. subjects can only initiate certain TPs on certain CDIs;
5. triples must enforce an appropriate separation-of-duty policy on subjects;
6. certain special TPs on UDIs can produce CDIs as output;
7. each application of a TP must cause enough information to reconstruct it to be written to a special append-only CDI;
8. the system must authenticate subjects attempting to initiate a TP;
9. the system must let only special subjects (i.e., security officers) make changes to authorization-related lists.

A number of things bear saying about Clark-Wilson.

First, unlike Bell-LaPadula, Clark-Wilson involves maintaining state. Even disregarding the audit trail, this is usually necessary for dual control as you have to keep track of which transactions have been partially approved — such as those approved by only one manager when two are needed. If dual control is implemented using access control mechanisms, it typically means holding partially approved transactions in a special journal file. This means that some of the user state is actually security state, which in turn makes the trusted computing base harder to define. If it is implemented using crypto instead, such as by having managers attach digital signatures to transactions of which they approve, then there can be problems managing all the partially approved transactions so that they get to a second approver in time.

Second, the model doesn't do everything. It captures the idea that state transitions should preserve an invariant such as balance, but not that state transitions should be correct. Incorrect transitions, such as paying into the wrong bank account, are not prevented by this model.

Third, Clark-Wilson ducks the hardest question, namely: how do we control the risks from dishonest staff? Rule 5 says that 'an appropriate separation of duty policy' must be supported, but nothing about what this means. Indeed, it's very hard to find any systematic description in the accounting literature of how you design internal controls — it's something that auditors tend to learn on the job. Companies' internal controls tend to evolve over time in response to real or feared incidents, whether in the company's own experience or its auditors'. In the next section, I try to distill into a few principles the experience gained from several years working at the coalface in banking and consultancy, and more recently on our university's finance and other committees.

10.2.2 Designing Internal Controls

Over the years, a number of standards have been put forward by the accountancy profession, by stock markets and by banking regulators, about how bookkeeping and internal control systems should be designed. In the USA, for example, there is the *Committee of Sponsoring Organizations* (COSO), a group of U.S. accounting and auditing bodies [318]. This self-regulation failed to stop the excesses of the dotcom era, and following the collapse of Enron there was intervention from U.S. lawmakers in the form of the Sarbanes-Oxley Act of 2002. It protects whistleblowers (the main source of information on serious insider fraud), and its section 404 makes managers responsible for maintaining 'adequate internal control structure and procedures for financial reporting'. It also demands that auditors attest to the management's assessment of these controls and disclose any 'material weaknesses'. CEOs also have to certify the truthfulness of financial statements. There was also the Gramm-Leach-Bliley Act of 1999, which liberalised bank regulation in many respects but

which obliged banks to have security mechanisms to protect information from foreseeable threats in security and integrity. Along with HIPAA in the medical sector, Gramm-Leach-Bliley and Sarbanes-Oxley have driven much of the investment in information security and internal control over the early years of the 21st century. (Other countries have equivalents; in the UK it's the Turnbull Guidance from the Financial Reporting Council.) I'll return to them and look in more detail at the policy aspects in Part III.

In this section, my concern is with the technical aspects. Modern risk management systems typically require a company to identify and assess its risks, and then build controls to mitigate them. A company's risk register might contain many pages of items such as 'insider makes large unauthorised bank transaction'. Some of these will be mitigated using non-technical measures such as insurance, but others will end up in your lap. So how do you engineer away a problem like this?

There are basically two kinds of separation of duty policy: dual control and functional separation.

In dual control, two or more staff members must act together to authorize a transaction. The classic military example is in nuclear command systems, which may require two officers to turn their keys simultaneously in consoles that are too far apart for any single person to reach both locks. I'll discuss nuclear command and control further in a later chapter. The classic civilian example is when a bank issues a letter of guarantee, which will typically undertake to carry the losses should a loan made by another bank go sour. Guarantees are particularly prone to fraud; if you can get bank A to guarantee a loan to your business from bank B, then bank B is supervising your account while bank A's money is at risk. A dishonest businessmen with a forged or corruptly-obtained guarantee can take his time to plunder the loan account at bank B, with the alarm only being raised when he absconds and bank B asks bank A for the money. If a single manager could issue such an instrument, the temptation would be strong. I'll discuss this further in section 10.3.2.

With functional separation of duties, two or more different staff members act on a transaction at different points in its path. The classic example is corporate purchasing. A manager takes a purchase decision and tells the purchasing department; a clerk there raises a purchase order; the store clerk records the goods' arrival; an invoice arrives at accounts; the accounts clerk correlates it with the purchase order and the stores receipt and raises a check; and the accounts manager signs the check.

However, it doesn't stop there. The manager now gets a debit on her monthly statement for that internal account, her boss reviews the accounts to make sure the division's profit targets are likely to be met, the internal audit department can descend at any time to audit the division's books, and when the external auditors come in once a year they will check the books of a randomly selected

sample of departments. Finally, when frauds are discovered, the company's lawyers may make vigorous efforts to get the money back.

So the model can be described as *prevent — detect — recover*. The level of reliance placed on each of these three legs will depend on the application. Where detection may be delayed for months or years, and recovery may therefore be very difficult — as with corrupt bank guarantees — it is prudent to put extra effort into prevention, using techniques such as dual control. Where prevention is hard, you should see to it that detection is fast enough, and recovery vigorous enough, to provide a deterrent effect. The classic example here is that bank tellers can quite easily take cash, so you need to count the money every day and catch them afterwards.

Bookkeeping and management control are not only one of the earliest security systems; they also have given rise to much of management science and civil law. They are entwined with a company's business processes, and exist in its cultural context. In Swiss banks, there are two managers' signatures on almost everything, while Americans are much more relaxed. In most countries' banks, staff get background checks, can be moved randomly from one task to another, and are forced to take holidays at least once a year. This would not be acceptable in the typical university — but in academia the opportunities for fraud are much less.

Designing an internal control system is hard because it's a highly interdisciplinary problem. The financial controllers, the personnel department, the lawyers, the auditors and the systems people all come at the problem from different directions, offer partial solutions, fail to understand each other's control objectives, and things fall down the hole in the middle. Human factors are very often neglected, and systems end up being vulnerable to helpful subordinates or authoritarian managers who can cause dual control to fail. It's important not just to match the controls to the culture, but also motivate people to use them; for example, in the better run banks, management controls are marketed to staff as a means of protecting them against blackmail and kidnapping.

Security researchers have so far focused on the small part of the problem which pertains to creating dual control (or in general, where there are more than two principals, *shared control*) systems. Even this is not at all easy. For example, rule 9 in Clark-Wilson says that security officers can change access rights — so what's to stop a security officer creating logons for two managers and using them to send all the bank's money to Switzerland?

In theory you could use cryptography, and split the signing key between two or more principals. In a Windows network, the obvious way to manage things is to put users in separately administered domains. With a traditional banking system using the mainframe operating system MVS, you can separate duties between the sysadmin and the auditor; the former can do anything he wishes, except finding out which of his activities the latter is monitoring [159]. But in real life, dual control is hard to do end-to-end because there are

many system interfaces that provide single points of failure, and in any case split-responsibility systems administration is tedious.

So the practical answer is that most bank sysadmins are in a position to do just this type of fraud. Some have tried, and where they fall down is when the back-office balancing controls set off the alarm after a day or two and money laundering controls stop him getting away with very much. I'll discuss this further in section 10.3.2. The point to bear in mind here is that serial controls along the *prevent — detect — recover* model are usually more important than shared control. They depend ultimately on some persistent state in the system and are in tension with programmers' desire to keep things simple by making transactions atomic.

There are also tranquility issues. For example, could an accountant knowing that he was due to be promoted to manager tomorrow end up doing both authorizations on a large transfer? A technical fix for this might involve a Chinese Wall mechanism supporting a primitive '*X* may do *Y* only if he hasn't done *Z*' ('A manager can confirm a payment only if his name doesn't appear on it as the creator'). So we would end up with a number of exclusion and other rules involving individuals, groups and object labels; once the number of rules becomes large (as it will in a real bank) we would need a systematic way of examining this rule set and verifying that it doesn't have any horrible loopholes.

In the medium term, banking security policy — like medical security policy — may find its most convenient expression in using role based access control, although this will typically be implemented in banking middleware rather than in an underlying platform such as Windows or Linux. Real systems will need to manage separation-of-duty policies with both parallel elements, such as dual control, and serial elements such as functional separation along a transaction's path. This argues for the access control mechanisms being near to the application. But then, of course, they are likely to be more complex, proprietary, and not so well studied as the mechanisms that come with the operating system.

One really important aspect of internal control in banking — and in systems generally — is to minimise the number of 'sysadmins', that is, of people with complete access to the whole system and the ability to change it. For decades now, the standard approach has been to keep development staff quite separate from live production systems. A traditional bank in the old days would have two mainframes, one to run the live systems, with the other being a backup machine that was normally used for development and testing. Programmers would create new software that would be tested by a test department and subject to source code review by internal auditors; once approved this would be handed off to a change management department that would install it in the live system at the next upgrade. The live system would be run by an operations

team with no access to compilers, debuggers or other tools that would let them alter live code or data.

In theory this prevents abuse by programmers, and in practice it can work fairly well. However there are leaks. First, there are always some sysadmins who need full access in order to do their jobs; and second, there are always emergencies. The ATM system goes down at the weekend, and the ATM team's duty programmer is given access to the live system from home so she can fix the bug. You audit such accesses as well as you can, but it's still inevitable that your top sysadmins will be so much more knowledgeable than your auditors that they could do bad things if they really wanted to. Indeed, at banks I've helped with security, you might find that there are thirty or forty people whom you just have to trust — the CEO, the chief dealer, the top sysadmins and a number of others. It's important to know who these people are, and to minimise their numbers. Pay them well — and watch discreetly to see if they start spending even more.

A final remark on dual control is that it's often not adequate for transactions involving more than one organization, because of the difficulties of dispute resolution: 'My two managers say the money was sent!' 'But my two say it wasn't!'

10.2.3 What Goes Wrong

Theft can take a variety of forms, from the purely opportunist to clever insider frauds; but the experience is that most thefts from the average company are due to insiders. There are many surveys; a typical one, by accountants Ernst and Young, reports that 82% of the worst frauds were committed by employees; nearly half of the perpetrators had been there over five years and a third of them were managers [1162].

Typical computer crime cases include:

- Paul Stubbs, a password reset clark at HSBC, conspired with persons unknown to change the password used by AT&T to access their bank account with HSBC. The new password was used to transfer £11.8 million — over $20 million — to offshore companies, from which it was not recovered. Stubbs was a vulnerable young man who had been employed as a password reset clerk after failing internal exams; the court took mercy on him and he got away with five years [975]. It was alleged that an AT&T employee had conspired to cover up the transactions, but that gentleman was acquitted.
- A bank had a system of suspense accounts, which would be used temporarily if one of the parties to a transaction could not be identified (such as when an account number was entered wrongly on a funds transfer). This was a workaround added to the dual control system to deal with

transactions that got lost or otherwise couldn't be balanced immediately. As it was a potential vulnerability, the bank had a rule that suspense accounts would be investigated if they were not cleared within three days. One of the clerks exploited this by setting up a scheme whereby she would post a debit to a suspense account and an equal credit to her boyfriend's account; after three days, she would raise another debit to pay off the first. In almost two years she netted hundreds of thousands of dollars. (The bank negligently ignored a regulatory requirement that all staff take at least ten consecutive days' vacation no more than fifteen months from the last such vacation.) In the end, she was caught when she could no longer keep track of the growing mountain of bogus transactions.

- A clerk at an education authority wanted to visit relatives in Australia, and in order to get some money she created a fictitious school, complete with staff whose salaries were paid into her own bank account. It was only discovered by accident when someone noticed that different records gave the authority different numbers of schools.
- A bank clerk in Hastings, England, noticed that the branch computer system did not audit address changes. He picked a customer who had a lot of money in her account and got a statement only once a year; he then changed her address to his, issued a new ATM card and PIN, and changed her address back to its original value. He stole £8,600 from her account, and when she complained she was not believed: the bank maintained that its computer systems were infallible, and so the withdrawals must have been her fault. The matter was only cleared up when the clerk got an attack of conscience and started stuffing the cash he'd stolen in brown envelopes through the branch's letter box at night. As people don't normally give money to banks, the branch manager finally realized that something was seriously wrong.

Volume crime — such as card fraud — often depends on liability rules. Where banks can tell customers who complain of fraud to get lost (as in much of Europe), bank staff know that complaints won't be investigated properly or at all, and get careless. Things are better in the USA where Regulation E places the onus of proof in disputed transaction cases squarely on the bank. I'll discuss this in detail in section 10.4 below.

All the really large frauds — the cases over a billion dollars — have involved lax internal controls. The collapse of Barings Bank is a good example: managers failed to control rogue trader Nick Leeson, blinded by greed for the bonuses his apparent trading profits earned them. The same holds true for other financial sector frauds, such as the Equity Funding scandal, in which an insurance company's management created thousands of fake people on their computer system, insured them, and sold the policies on to reinsurers; and frauds in

other sectors such as Robert Maxwell's looting of the Daily Mirror newspaper pension funds in Britain. (For a collection of computer crime case histories, see Parker [1005].) Either the victim's top management were grossly negligent, as in the case of Barings, or perpetrated the scam, as with Equity Funding and Maxwell.

The auditors are also a problem. On the one hand, they are appointed by the company's managers and are thus extremely bad at detecting frauds in which the managers are involved; so the assurance that shareholders get is less than many might have thought. (The legal infighting following the collapse of Enron destroyed its auditors Arthur Andersen and thus reduced the 'big five' audit firms to the 'big four'; now auditors go out of their way to avoid liability for fraud.) Second, there were for many years huge conflicts of interest, as accountants offered cheap audits in order to get their foot in the door, whereupon they made their real money from systems consultancy. (This has been greatly restricted since Enron.) Third, the big audit firms have their own list of favourite controls, which often bear little relationship to the client's real risks, and may even make matters worse. For example, our university's auditors nag us every year to get all our staff to change their passwords every month. This advice is wrong, for reasons explained in Chapter 2 — so every year we point this out and challenge them to justify their advice. But they seem incapable of learning, and they have no incentive to: they can be expected to nitpick, and to ignore any evidence that a particular nit is unhelpful until long after the evidence has become overwhelming. While failing to disclose a material weakness could get them into trouble, at least in the USA, the nitpicking has turned into a bonanza for them. It's reckoned that the auditors' gold-plating of the Sarbanes-Oxley requirements is costing the average U.S. listed company $2.4m a year in audit fees, plus 70,000 hours of internal work to ensure compliance; the total cost of SOX could be as much as $1.4 trillion [412]. (My own advice, for what it's worth, is to never use a big-four accountant; smaller firms are cheaper, and a study done by my student Tyler Moore failed to find any evidence that companies audited by the Big Four performed better on the stock market.)

Changing technology also has a habit of eroding controls, which therefore need constant attention and maintenance. For example, thanks to new systems for high-speed processing of bank checks, banks in California stopped a few years ago from honoring requests by depositors that checks have two signatures. Even when a check has imprinted on it 'Two Signatures Required', banks will honor that check with only one signature [1086]. This might seem to be a problem for the customer's security rather than the bank's, but bank checks can also be at risk and if something goes wrong even with a merchant transaction then the bank might still get sued.

The lessons to be learned include:

- it's not always obvious which transactions are security sensitive;
- maintaining a working security system can be hard in the face of a changing environment;
- if you rely on customer complaints to alert you to fraud, you had better listen to them;
- there will always be people in positions of relative trust who can get away with a scam for a while;
- no security policy will never be completely rigid. There will always have to be workarounds for people to cope with real life;
- these workarounds naturally create vulnerabilities. So the lower the transaction error rate, the better.

There will always be residual risks. Managing these residual risks remains one of the hardest and most neglected of jobs. It means not just technical measures, such as involving knowledgeable industry experts, auditors and insurance people in the detailed design, and iterating the design once some loss history is available. It also means training managers, auditors and others to detect problems and react to them appropriately. I'll revisit this in Part III.

The general experience of banks in the English-speaking world is that some 1% of staff are sacked each year. The typical offence is minor embezzlement with a loss of a few thousand dollars. No-one has found an effective way of predicting which staff will go bad; previously loyal staff can be thrown off the rails by shocks such as divorce, or may over time develop a gambling or alcohol habit. Losing a few hundred tellers a year is simply seen as a cost of doing business. What banks find very much harder to cope with are incidents in which senior people go wrong — indeed, in several cases within my experience, banks have gone to great lengths to avoid admitting that a senior insider was bent. And risks that managers are unwilling to confront, they are often unable to control. No-one at Barings even wanted to think that their star dealer Leeson might be a crook; and pop went the bank.

Finally, it's not enough, when doing an audit or a security investigation, to merely check that the books are internally consistent. It's also important to check that the correspond to external reality. This was brought home to the accounting profession in 1938 with the collapse of McKesson and Robbins, a large, well-known drug and chemical company with reported assets of $100m[1]. It turned out that 20% of the recorded assets and inventory were nonexistent. The president, Philip Musica, turned out to be an impostor with

[1] In 2007 dollars, that's $1.4bn if you deflate by prices, $3bn if you deflate by unskilled wages and over $15bn by share of GDP.

a previous fraud conviction; with his three brothers, he inflated the firm's figures using a fake foreign drug business involving a bogus shipping agent and a fake Montreal bank. The auditors, who had accepted the McKesson account without making enquiries about the company's bosses, had failed to check inventories, verify accounts receivable with customers, or think about separation of duties within the company [1082]. The lessons from that incident clearly weren't learned well enough, as the same general things continue to happen regularly and on all sorts of scales from small firms and small branches of big ones, to the likes of Enron.

So if you ever have responsibility for security in a financial (or other) firm, you should think hard about which of your managers could defraud your company by colluding with customers or suppliers. Could a branch manager be lending money to a dodgy business run by his cousin against forged collateral? Could he have sold life-insurance policies to nonexistent people and forged their death certificates? Could an operations manager be taking bribes from a supplier? Could one of your call-center staff be selling customer passwords to the Mafia? Lots of things can and do go wrong; you have to figure out which of them matter, and how you get to find out. Remember: a trusted person is one who can damage you. Who can damage you, and how? This is the basic question that a designer of internal controls must be constantly asking.

10.3 Wholesale Payment Systems

When people think of electronic bank fraud, they often envisage a Hollywood scene in which crafty Russian hackers break a bank's codes and send multi-million dollar wire transfers to tax havens. Systems for transferring money electronically are indeed an occasional target of sophisticated crime, and have been for a century and a half, as I noted earlier in section 5.2.4 when I discussed test key systems.

By the early 1970s, bankers started to realise that this worthy old Victorian system was due for an overhaul.

First, most test-key systems were vulnerable in theory at least to cryptanalysis; someone who observed a number of transactions could gradually work out the key material.

Second, although the test key tables were kept in the safe, there was nothing really to stop staff members working out tests for unauthorised messages at the same time as a test for an authorised message. In theory, you might require that two staff members retrieve the tables from the safe, sit down at a table facing each other and perform the calculation. However, in practice people would work sequentially in a corner (the tables were secret, after all) and even if you could compel them to work together, a bent employee might mentally compute the test on an unauthorized message while overtly computing the

test on an authorized one. So, in reality, test key schemes didn't support dual control. Having tests computed by one staff member and checked by another doubled the risk rather than reducing it. (There are ways to do dual control with manual authenticators, such as by getting the two staff members to work out different codes using different tables; and there are other techniques, used in the control of nuclear weapons, which I'll discuss in 13.4.)

Third, there was a big concern with cost and efficiency. There seemed little point in having the bank's computer print out a transaction in the telex room, having a test computed manually, then composing a telex to the other bank, then checking the test, and then entering it into the other bank's computer. Errors were much more of a problem than frauds, as the telex operators introduced typing errors. Customers who got large payments into their accounts in error sometimes just spent the money, and in one case an erroneous recipient spent some of his windfall on clever lawyers, who helped him keep it. This shocked the industry. Surely the payments could flow directly from one bank's computer to another?

10.3.1 SWIFT

The Society for Worldwide International Financial Telecommunications (SWIFT) was set up in the 1970s by a consortium of banks to provide a more secure and efficient means than telex of sending payment instructions between member banks. It can be thought of as an email system with built-in encryption, authentication and non-repudiation services. It's important not just because it's used to ship trillions of dollars round the world daily, but because its design has been copied in systems processing many other kinds of intangible asset, from equities to bills of lading.

The design constraints are interesting. The banks did not wish to trust SWIFT, in the sense of enabling dishonest employees there to forge transactions. The authenticity mechanisms had to be independent of the confidentiality mechanisms, since at the time a number of countries (such as France) forbade the civilian use of cryptography for confidentiality. The non-repudiation functions had to be provided without the use of digital signatures, as these hadn't been invented yet. Finally, the banks had to be able to enforce Clark-Wilson type controls over interbank transactions. (Clark-Wilson also hadn't been invented yet but its components, dual control, balancing, audit and so on, were well enough established.)

The SWIFT design is summarized in Figure 10.2. Authenticity of messages was assured by computing a message authentication code (MAC) at the sending bank and checking it at the receiving bank. The keys for this MAC used to be managed end-to-end: whenever a bank set up a relationship overseas, the senior manager who negotiated it would exchange keys with his opposite number, whether in a face-to-face meeting or afterwards by post to each others' home addresses. There would typically be two key components

to minimize the risk of compromise, with one sent in each direction (since even if a bank manager's mail is read in his mailbox by a criminal at one end, it's not likely to happen at both). The key was not enabled until both banks confirmed that it had been safely received and installed.

This way, SWIFT had no part in the message authentication; so long as the authentication algorithm in use was sound, none of their staff could forge a transaction. The authentication algorithm used is supposed to be a trade secret; but as banks like their security mechanisms to be international standards, a natural place to look might be the algorithm described in ISO 8731 [1094]. In this way, they got the worst of all possible worlds: the algorithm was fielded without the benefit of public analysis but got it later once it was expensive to change! An attack was found on the ISO 8731 message authentication algorithm and published in [1040] but, fortunately for the industry, it takes over 100,000 messages to recover a key — which is too large for a practical attack on a typical system that is used prudently.

Although SWIFT itself was largely outside the trust perimeter for message authentication, it did provide a non-repudiation service. Banks in each country sent their messages to a *Regional General Processor* (RGP) which logged them and forwarded them to SWIFT, which also logged them and sent them on to the recipient via the RGP in his country, which also logged them. The RGPs were generally run by different facilities management firms. Thus a bank (or a crooked bank employee) wishing to dishonestly repudiate a done transaction would have to subvert not just the local SWIFT application and its surrounding controls, but also two independent contractors in different countries. Note that the repudiation control from multiple logging is better than the integrity control. A bent bank wanting to claim that a transaction had been done when it hadn't could always try to insert the message between the other bank and their RGP; while a bent bank employee would probably just insert a bogus incoming message directly into a local system. So logs can be a powerful way of making repudiation difficult, and are much easier for judges to understand than cryptography.

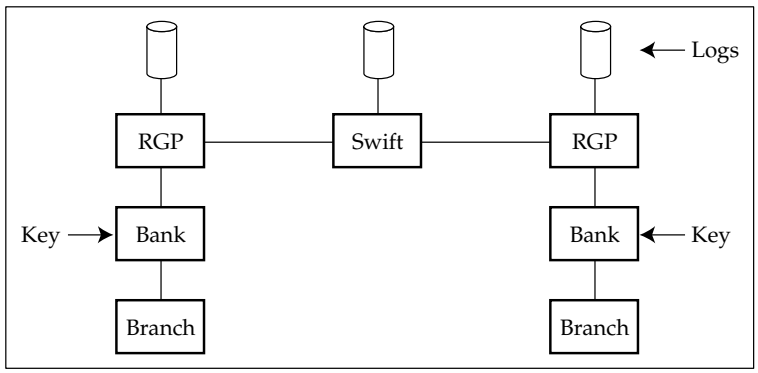


Figure 10.2: Architecture of SWIFT

Confidentiality depended on line encryption devices between the banks and the RGP node, and between these nodes and the main SWIFT processing sites. Key management was straightforward at first. Keys were hand carried in EEPROM cartridges between the devices at either end of a leased line. In countries where confidentiality was illegal, these devices could be omitted without impairing the authenticity and non-repudiation mechanisms.

Dual control was provided either by the use of specialized terminals (in small banks) or by mainframe software packages which could be integrated with a bank's main production system. The usual method of operation is to have three separate staff to do a SWIFT transaction: one to enter it, one to check it, and one to authorize it. (As the checker can modify any aspect of the message, this really only gives dual control, not triple control — and the programmers who maintain the interface can always attack the system there). Reconciliation was provided by checking transactions against daily statements received electronically from correspondent banks. This meant that someone who managed to get a bogus message into the system would sound an alarm within two or three days.

10.3.2 What Goes Wrong

SWIFT I ran for twenty years without a single report of external fraud. In the mid 1990s, it was enhanced by adding public key mechanisms; MAC keys are now shared between correspondent banks using public key cryptography and the MACs themselves may be further protected by a digital signature. The key management mechanisms have been ensconced as ISO standard 11166, which in turn has been used in other systems (such as CREST, which is used by banks and stockbrokers to register and transfer UK stocks and shares). There has been some debate over the security of this architecture [73, 1094]. Quite apart from the centralization of trust brought about by the adoption of public key cryptography — in that the central certification authority can falsely certify a key as belonging to a bank when it doesn't — CREST adopted 512-bit public keys, and these are too short: as I mentioned in the chapter on cryptology, at least one RSA public key of this length has been factored surreptitiously by a group of students [24].

However the main practical attacks on such systems have not involved the payment mechanisms themselves. The typical attack comes from a bank programmer inserting a bogus message into the processing queue. It usually fails because he does not understand the other controls in the system, or the procedural controls surrounding large transfers. For example, banks maintain accounts with each other, so when bank A sends money to a customer of bank B it actually sends an instruction 'please pay this customer the following sum out of our account with you'. As these accounts have both balances and credit limits, large payments aren't processed entirely automatically but need intervention from the dealing room to ensure that the needed currency or

credit line is made available. So transfers over a million dollars or so tend to need managerial interventions of which technical staff are ignorant; and there are also filters that look for large transactions so that the bank can report them to the money-laundering authorities if need be. There is also the common-sense factor, in that anyone who opens a bank account, receives a large incoming wire transfer and then starts frantically moving money out again used to need a very convincing excuse. (Common sense has become less of a backstop since 9/11 as customer due diligence and anti-money-laundering rules have become both formalised and onerous; bank staff rely more on box-ticking, which has made life easier for the bad guys [55].) In any case, the programmer who inserts a bogus transaction into the system usually gets arrested when he turns up to collect the cash. If your life's goal is a career in bank fraud, you're better off getting an accounting or law degree and working in a loans office rather than messing about with computers.

Other possible technical attacks, such as inserting Trojan software into the PCs used by bank managers to initiate transactions, wiretapping the link from the branch to the bank mainframe, subverting the authentication protocol used by bank managers to log on, and even inserting a bogus transaction in the branch LAN causing it to appear on the relevant printer — would also run up against the business-process controls. In fact, most large scale bank frauds which 'worked' have not used technical attacks but exploited procedural vulnerabilities.

- The classic example is a letter of guarantee. It is common enough for a company in one country to guarantee a loan to a company in another. This can be set up as a SWIFT message, or even a paper letter. But as no cash changes hands at the time, the balancing controls are inoperative. If a forged guarantee is accepted as genuine, the 'beneficiary' can take his time borrowing money from the accepting bank, laundering it, and disappearing. Only when the victim bank realises than the loan has gone sour and tries to call in the guarantee is the forgery discovered.
- An interesting fraud of a slightly different type took place in 1986 between London and Johannesburg. At that time, the South African government operated two exchange rates, and in one bank the manager responsible for deciding which rate applied to each transaction conspired with a rich man in London. They sent money out to Johannesburg at an exchange rate of seven Rand to the Pound, and back again the following day at four. After two weeks of this, the authorities became suspicious, and the police came round. On seeing them in the dealing room, the manager fled without stopping to collect his jacket, drove over the border to Swaziland, and flew via Nairobi to London. There, he boasted to the press about how he had defrauded the wicked apartheid system. As the UK has no exchange control, exchange control

fraud isn't an offence and so he couldn't be extradited. The conspirators got away with millions, and the bank couldn't even sue them.

- Perhaps the best known money transfer fraud occurred in 1979 when Stanley Rifkin, a computer consultant, embezzled over ten million dollars from Security Pacific National Bank. He got round the money laundering controls by agreeing to buy a large shipment of diamonds from a Russian government agency in Switzerland. He got the transfer into the system by observing an authorization code used internally when dictating transfers to the wire transfer department, and simply used it over the telephone (a classic example of dual control breakdown at a system interface). He even gave himself extra time to escape by doing the deal just before a U.S. bank holiday. Where he went wrong was in not planning what to do after he collected the stones. If he'd hid them in Europe, gone back to the USA and helped investigate the fraud, he might well have got away with it; as it was, he ended up on the run and got caught.

The system design lesson is unchanged: one must always be alert to things which defeat the dual control philosophy. However, as time goes on we have to see it in a broader context. Even if we can solve the technical problems of systems administration, interfaces and so on, there's still the business process problem of what we control — quite often critical transactions don't appear as such at a casual inspection.

10.4 Automatic Teller Machines

Another set of lessons about the difficulties and limitations of dual control emerges from studying the security of *automatic teller machines* (ATMs). ATMs, also known as cash machines, have been one of the most influential technological innovations of the 20th century.

ATMs were the first large-scale retail transaction processing systems. They were devised in 1938 by the inventor Luther Simjian, who also thought up the teleprompter and the self-focussing camera. He persuaded Citicorp to install his 'Bankamat' machine in New York in 1939; they withdrew it after six months, saying 'the only people using the machines were a small number of prostitutes and gamblers who didn't want to deal with tellers face to face' [1168]. Its commercial introduction dates to 1967, when a machine made by De La Rue was installed by Barclays Bank in Enfield, London. The world installed base is now thought to be about 1,500,000 machines. The technology developed for them is now also used in card payment terminals in shops. Modern block ciphers were first used on a large scale in ATM networks: they are used to generate and verify PINs in secure hardware devices located within the ATMs and at bank computer centres. This technology, including

block ciphers, tamper-resistant hardware and the supporting protocols, ended up being used in many other applications from postal franking machines to lottery ticket terminals. In short, ATMs were the 'killer app' that got modern commercial cryptology and retail payment technology off the ground.

10.4.1 ATM Basics

Most ATMs operate using some variant of a system developed by IBM for its 3624 series cash machines in the late 1970s. The card contains the customer's primary account number, *PAN*. A secret key, called the 'PIN key', is used to encrypt the account number, the decimalize it and truncate it. The result of this operation is called the 'natural PIN'; an offset can be added to it in order to give the PIN which the customer must enter. The offset has no real cryptographic function; it just enables customers to choose their own PIN. An example of the process is shown in Figure 10.3.

In the first ATMs to use PINs, each ATM contained a copy of the PIN key and each card contained the offset as well as the primary account number. Each ATM could thus verify all customer PINs. Early ATMs also operated offline; if your cash withdrawal limit was $500 per week, a counter was kept on the card. In recent years networks have become more dependable and ATMs have tended to operate online only, which simplifies the design; the cash counters and offsets have vanished from magnetic strips and are now kept on servers. In the last few years, magnetic strips have been supplemented with smartcard chips in some countries, especially in Europe; I will describe the smartcard systems later. However the basic principle remains: PINs are generated and protected using cryptography.

Dual control is implemented in this system using tamper-resistant hardware. A cryptographic processor, also known as a *security module*, is kept in the bank's server room and will perform a number of defined operations on customer PINs and on related keys in ways that enforce a dual-control policy. This includes the following.

1. Operations on the clear values of customer PINs, and on the keys needed to compute them or used to protect them, are all done in tamper-resistant

Account number *PAN*:	`8807012345691715`
PIN key *KP*:	`FEFEFEFEFEFEFEFE`
Result of DES $\{PAN\}_{KP}$:	`A2CE126C69AEC82D`
$\{N\}_{KP}$ decimalized:	`0224126269042823`
Natural PIN:	`0224`
Offset:	`6565`
Customer PIN:	`6789`

Figure 10.3: IBM method for generating bank card PINs

hardware and the clear values are never made available to any single member of the bank's staff.

2. Thus, for example, the cards and PINs are sent to the customer via separate channels. The cards are personalized in a facility with embossing and mag strip printing machinery, and the PIN mailers are printed in a separate facility containing a printer attached to a security module.
3. A *terminal master key* is supplied to each ATM in the form of two printed components, which are carried to the branch by two separate officials, input at the ATM keyboard, and combined to form the key. Similar procedures (but with three officials) are used to set up master keys between banks and network switches such as VISA.
4. If ATMs are to perform PIN verification locally, then the PIN key is encrypted under the terminal master key and then sent to the ATM.
5. If the PIN verification is to be done centrally over the network, the PIN is encrypted under a key set up using the terminal master key. It will then be sent from the ATM to a central security module for checking.
6. If the bank's ATMs are to accept other banks' cards, then its security modules use transactions that take a PIN encrypted under an ATM key, decrypt it and re-encrypt it for its destination, such as using a key shared with VISA. This *PIN translation* function is done entirely within the hardware security module, so that clear values of PINs are never available to the bank's programmers. VISA will similarly decrypt the PIN and re-encrypt it using a key shared with the bank that issued the card, so that the PIN can be verified by the security module that knows the relevant PIN key.

During the 1980s and 1990s, hardware security modules became more and more complex, as ever more functionality got added to support more complex financial applications from online transactions to smartcards. An example of a leading product is the IBM 4758 — this also has the virtue of having its documentation available publicly online for study (see [641] for the command set and [1195] for the architecture and hardware design). We'll discuss this later in the chapter on tamper resistance.

But extending the dual control security policy from a single bank to tens of thousands of banks worldwide, as modern ATM networks do, was not completely straightforward.

- When people started building ATM networks in the mid 1980s, many banks used software encryption rather than hardware security modules to support ATMs. So in theory, any bank's programmers might get access to the PINs of any other bank's customers. The remedy was to push through standards for security module use. In many countries

(such as the USA), these standards were largely ignored; but even where they were respected, some banks continued using software for transactions involving their own customers. So some keys (such as those used to communicate with ATMs) had to be available in software too, and knowledge of these keys could be used to compromise the PINs of other banks' customers. (I'll explain this in more detail later.) So the protection given by the hardware TCB was rarely complete.

- It is not feasible for 10,000 banks to share keys in pairs, so each bank connects to a switch provided by an organization such as VISA or Cirrus, and the security modules in these switches translate the traffic. The switches also do accounting, and enable banks to settle their accounts for each day's transactions with all the other banks in the system by means of a single electronic debit or credit. The switch is highly trusted, and if something goes wrong there the consequences can be severe. In one case, there turned out to be not just security problems but also dishonest staff. The switch manager ended up a fugitive from justice, and the bill for remediation was in the millions. In another case, a Y2K-related software upgrade at a switch was bungled, with the result that cardholders in one country found that for a day or two they could withdraw money even if their accounts were empty. This also led to a very large bill.
- Corners are cut to reduce the cost of dealing with huge transaction volumes. For example, it is common for authentication of authorization responses to be turned off. So anyone with access to the network can cause a given ATM to accept any card presented to it, by simply replaying a positive authorization response. Network managers claim that should a fraud ever start, then the authentication can always be turned back on. This might seem reasonable; attacks involving manipulated authorization responses are very rare. Similarly, after UK banks put smartcard chips into bank cards, some of them kept in accepting magnetic-strip transactions, so that a card with a broken chip would still work so long as the magnetic strip could be read. But such shortcuts — even when apparently reasonable on grounds of risk and cost — mean that a bank which stonewalls customer complaints by saying its ATM network is secure, and so the transaction must be the customer's fault, is not telling the truth. This may lay the bank and its directors open to fraud charges. What's more, changing the network's modus operandi suddenly in response to a fraud can be difficult; it can unmask serious dependability problems or lead to unacceptable congestion. This brings home the late Roger Needham's saying that 'optimization is the process of taking something which works, and replacing it by something which doesn't quite but is cheaper'.

There are many other ways in which ATM networks can be attacked in theory, and I'll discuss a number of them later in the context of interface security: the design of the hardware security modules that were in use for decades was so poor that programmers could extract PINs and keys simply by issuing suitable sequences of commands to the device's interface with the server, without having to break either the cryptographic algorithms or the hardware tamper-resistance. However, one of the interesting things about these systems is that they have now been around long enough, and have been attacked enough by both insiders and outsiders, to give us a lot of data points on how such systems fail in practice.

10.4.2 What Goes Wrong

ATM fraud is an interesting study as this is a mature system with huge volumes and a wide diversity of operators. There have been successive waves of ATM fraud, which have been significant since the early 1990s. In each wave, a set of vulnerabilities was exploited and then eventually fixed; but the rapidly growing scale of payment card operations opened up new vulnerabilities. There is a fascinating interplay between the technical and regulatory aspects of protection.

The first large wave of fraud lasted from perhaps 1990–96 and exploited the poor implementation and management of early systems. In the UK, one prolific fraudster, Andrew Stone, was convicted three times of ATM fraud, the last time getting five and a half years in prison. He got involved in fraud when he discovered by chance the 'encryption replacement' trick I discussed in the chapter on protocols: he changed the account number on his bank card to his wife's and found by chance that he could take money out of her account using his PIN. In fact, he could take money out of any account at that bank using his PIN. This happened because his bank (and at least two others) wrote the encrypted PIN to the card's magnetic strip without linking it to the account number in any robust way (for example, by using the 'offset' method described above). His second method was 'shoulder surfing': he'd stand in line behind a victim, observe the entered PIN, and pick up the discarded ATM slip. Most banks at the time printed the full account number on the slip, and a card would work with no other correct information on it.

Stone's methods spread via people he trained as his accomplices, and via a 'Howto' manual he wrote in prison. Some two thousand victims of his (and other) frauds banded together to bring a class action against thirteen banks to get their money back; the banks beat this on the technical legal argument that the facts in each case were different. I was an expert in this case, and used it write a survey of what went wrong [33] (there is further material in [34]). The fraud spread to the Netherlands, Italy and eventually worldwide, as criminals

learned a number of simple hacks. Here I'll summarize the more important and interesting lessons we learned.

The engineers who designed ATM security systems in the 1970s and 1980s (of whom I was one) had assumed that criminals would be relatively sophisticated, fairly well-informed about the system design, and rational in their choice of attack methods. In addition to worrying about the many banks which were slow to buy security modules and implementation loopholes such as omitting authentication codes on authorization responses, we agonized over whether the encryption algorithms were strong enough, whether the tamper-resistant boxes were tamper-resistant enough, and whether the random number generators used to manufacture keys were random enough. We knew we just couldn't enforce dual control properly: bank managers considered it beneath their dignity to touch a keyboard, so rather than entering the ATM master key components themselves after a maintenance visit, most of them would just give both key components to the ATM engineer. We wondered whether a repairman would get his hands on a bank's PIN key, forge cards in industrial quantities, close down the whole system, and wreck public confidence in electronic banking.

The great bulk of the actual 'phantom withdrawals', however, appeared to have one of the following three causes:

- Simple processing errors account for a lot of disputes. With U.S. customers making something like 5 billion ATM withdrawals a year, even a system that only makes one error per hundred thousand transactions will give rise to 50,000 disputes a year. In practice the error rate seems to lie somewhere between 1 in 10,000 and 1 in 100,000. One source of errors we tracked down was that a large bank's ATMs would send a transaction again if the network went down before a confirmation message was received from the mainframe; periodically, the mainframe itself crashed and forgot about open transactions. We also found customers whose accounts were debited with other customers' transactions, and other customers who were never debited at all for their card transactions. (We used to call these cards 'directors' cards' and joked that they were issued to bank directors.)
- Thefts from the mail were also huge. They are reckoned to account for 30% of all UK payment card losses, but most banks' postal control procedures have always been dismal. For example, when I moved to Cambridge in February 1992 I asked my bank for an increased card limit: the bank sent not one, but two, cards and PINs through the post. These cards arrived only a few days after intruders had got hold of our apartment block's mail and torn it up looking for valuables. It turned out that this bank did not have the systems to deliver a card by registered post. (I'd asked them to send the card to the branch for me to collect but the

branch staff had simply re-addressed the envelope to me.) Many banks now make you phone a call center to activate a card before you can use it. This made a dent in the fraud rates.

- Frauds by bank staff appeared to be the third big cause of phantoms. We mentioned the Hastings case in section 10.2.3 above; there are many others. For example, in Paisley, Scotland, an ATM repairman installed a portable computer inside an ATM to record customer card and PIN data and then went on a spending spree with forged cards. In London, England, a bank stupidly used the same cryptographic keys in its live and test systems; maintenance staff found out that they could work out customer PINs using their test equipment, and started offering this as a service to local criminals at £50 a card. Insider frauds were particularly common in countries like Britain where the law generally made the customer pay for fraud, and rarer in countries like the USA where the bank paid; British bank staff knew that customer complaints wouldn't be investigated carefully, so they got lazy, careless, and sometimes bent.

These failures are all very much simpler and more straightforward than the ones we'd worried about. In fact, the only fraud we had worried about and that actually happened to any great extent was on offline processing. In the 1980s, many ATMs would process transactions while the network was down, so as to give 24-hour service; criminals — especially in Italy and later in England too — learned to open bank accounts, duplicate the cards and then 'jackpot' a lot of ATMs overnight when the network was down [775]. This forced most ATM operations to be online-only by 1994.

However, there were plenty of frauds that happened in quite unexpected ways. I already mentioned the Utrecht case in section 3.5, where a tap on a garage point-of-sale terminal was used to harvest card and PIN data; and Stone's 'encryption replacement' trick. There were plenty more.

- Stone's shoulder-surfing trick of standing in an ATM queue, observing a customer's PIN, picking up the discarded ticket and copying the data to a blank card, was not in fact invented by him. It was first reported in New York in the mid 1980s; and it was still working in the Bay Area in the mid 1990s. By then it had been automated; Stone (and Bay area criminals) used video cameras with motion sensors to snoop on PINs, whether by renting an apartment overlooking an ATM or even parking a rented van there. Visual copying is easy to stop: the standard technique nowadays is to print only the last four digits of the account number on the ticket, and there's also a three-digit 'card verification value' (CVV) on the magnetic strip that should never be printed. Thus even if the villain's camera is good enough to read the account number and expiry date from the front of the card, a working copy can't be made. (The CVV

is like the three-digit security code on the signature strip, but different digits; each is computed from the account number and expiry date by encrypting them with a suitable key). Surprisingly, it still happens; I have a letter from a UK bank dated May 2007 claiming that a series of terrorist-related ATM frauds were perpetrated using closed-circuit TV. This amounts to an admission that the CVV is not always checked.

- There were some losses due to programming errors by banks. One small institution issued the same PIN to all its customers; another bank's cash machines had the feature that when a telephone card was entered at an ATM, it believed that the previous card had been inserted again. Crooks stood in line, observed customers' PINs, and helped themselves.
- There were losses due to design errors by ATM vendors. One model, common in the 1980s, would output ten banknotes from the lowest denomination non-empty cash drawer, whenever a certain fourteen digit sequence was entered at the keyboard. One bank printed this sequence in its branch manual, and three years later there was a sudden spate of losses. All the banks using the machine had to rush out a patch to disable the test transaction. And despite the fact that I documented this in 1993, and again in the first edition of this book in 2001, similar incidents are still reported in 2007. Some makes of ATM used in stores can be reprogrammed into thinking that they are dispensing $1 bills when in fact they're dispensing twenties; it just takes a default master password that is printed in widely-available online manuals. Any passer-by who knows this can stroll up to the machine, reset the bill value, withdraw $400, and have his account debited with $20. The store owners who lease the machines are not told of the vulnerability, and are left to pick up the tab [1037].
- Several banks thought up check-digit schemes to enable PINs to be checked by offline ATMs without having to give them the bank's PIN key. For example, customers of one British bank get a credit card PIN with digit one plus digit four equal to digit two plus digit three, and a debit card PIN with one plus three equals two plus four. Crooks found they could use stolen cards in offline devices by entering a PIN such as 4455.
- Many banks' operational security procedures were simply dire. In August 1993, my wife went into a branch of our bank with a witness and told them she'd forgotten her PIN. The teller helpfully printed her a new PIN mailer from a printer attached to a PC behind the counter — just like that! It was not the branch where our account is kept. Nobody knew her and all the identification she offered was our bank card and her checkbook. When anyone who's snatched a handbag can walk in off

the street and get a PIN for the card in it at any branch, no amount of encryption technology will do much good. (The bank in question has since fallen victim to a takeover.)

- A rapidly growing modus operandi in the early 1990s was to use false terminals to collect card and PIN data. The first report was from the USA in 1988; there, crooks built a vending machine which would accept any card and PIN, and dispense a packet of cigarettes. In 1993, two villains installed a bogus ATM in the Buckland Hills Mall in Connecticut [667, 962]. They had managed to get a proper ATM and a software development kit for it — all bought on credit. Unfortunately for them, they decided to use the forged cards in New York, where cash machines have hidden video cameras, and as they'd crossed a state line they ended up getting long stretches in Club Fed.

So the first thing we did wrong when designing ATM security systems in the early to mid 1980s was to worry about criminals being clever, when we should rather have worried about our customers — the banks' system designers, implementers and testers — being stupid. Crypto is usually only part of a very much larger system. It gets a lot of attention because it's mathematically interesting; but as correspondingly little attention is paid to the 'boring' bits such as training, usability, standards and audit, it's rare that the bad guys have to break the crypto to compromise a system. It's also worth bearing in mind that there are so many users for large systems such as ATM networks that we must expect the chance discovery and exploitation of accidental vulnerabilities which were simply too obscure to be caught in testing.

The second thing we did wrong was to not figure out what attacks could be industrialised, and focus on those. In the case of ATMs, the false-terminal attack is the one that made the big time. The first hint of organised crime involvement was in 1999 in Canada, where dozens of alleged Eastern European organized-crime figures were arrested in the Toronto area for deploying doctored point-of-sale terminals [85, 152]. The technology has since become much more sophisticated; 'skimmers' made in Eastern Europe are attached to the throats of cash machines to read the magnetic strip and also capture the PIN using a tiny camera. I'll discuss these in more detail in the next section. Despite attempts to deal with false-terminal attacks by moving from magnetic strip cards to smartcards, they have become pervasive. They will be difficult and expensive to eliminate.

10.4.3 Incentives and Injustices

In the USA, the banks have to carry the risks associated with new technology. This was decided in a historic precedent, Judd versus Citibank, in which bank customer Dorothy Judd claimed that she had not made some disputed withdrawals and Citibank said that as its systems were secure, she must have done.

The judge ruled that Citibank's claim to infallibility was wrong in law, as it put an unmeetable burden of proof on her, and gave her her money back [674]. The U.S. Federal Reserve incorporated this into 'Regulation E', which requires banks to refund all disputed transactions unless they can prove fraud by the customer [440]. This has led to some minor abuse — misrepresentations by customers are estimated to cost the average U.S. bank about $15,000 a year — but this is an acceptable cost (especially as losses from vandalism are typically three times as much) [1362].

In other countries — such as the UK, Germany, the Netherlands and Norway — the banks got away for many years with claiming that their ATM systems were infallible. Phantom withdrawals, they maintained, could not possibly exist and a customer who complained of one must be mistaken or lying. This position was demolished in the UK when Stone and a number of others started being jailed for ATM fraud, as the problem couldn't be denied any more. Until that happened, however, there were some rather unpleasant incidents which got banks a lot of bad publicity [34]. The worst was maybe the Munden case.

John Munden was one of our local police constables, based in Bottisham, Cambridgeshire; his beat included the village of Lode where I lived at the time. He came home from holiday in September 1992 to find his bank account empty. He asked for a statement, found six withdrawals for a total of £460 which he did not recall making, and complained. His bank responded by having him prosecuted for attempting to obtain money by deception. It came out during the trial that the bank's system had been implemented and managed in a ramshackle way; the disputed transactions had not been properly investigated; and all sorts of wild claims were made by the bank, such as that their ATM system couldn't suffer from bugs as its software was written in assembler. Nonetheless, it was his word against the bank's. He was convicted in February 1994 and sacked from the police force.

This miscarriage of justice was overturned on appeal, and in an interesting way. Just before the appeal was due to be heard, the prosecution served up a fat report from the bank's auditors claiming that the system was secure. The defense demanded equal access to the bank's systems for its own expert. The bank refused and the court therefore disallowed all its computer evidence — including even its bank statements. The appeal succeeded, and John got reinstated. But this was only in July 1996 — he'd spent the best part of four years in limbo and his family had suffered terrible stress. Had the incident happened in California, he could have won enormous punitive damages — a point bankers should ponder as their systems become global and their customers can be anywhere.[2]

[2]Recently the same drama played itself out again when Jane Badger, of Burton-on-Trent, England, was prosecuted for complaining about phantom withdrawals. The case against her collapsed in January 2008. The bank, which is called Egg, is a subsidiary of Citicorp.

The lesson to be drawn from such cases is that dual control is not enough. If a system is to provide evidence, then it must be able to withstand examination by hostile experts. In effect, the bank had used the wrong security policy. What they really needed wasn't dual control but *non-repudiation*: the ability for the principals in a transaction to prove afterwards what happened. This could have been provided by installing ATM cameras; although these were available (and are used in some U.S. states), they were not used in Britain. Indeed, during the 1992–4 wave of ATM frauds, the few banks who had installed ATM cameras were pressured by the other banks into withdrawing them; camera evidence could have undermined the stance that the banks took in the class action that their systems were infallible.

One curious thing that emerged from this whole episode was that although U.S. banks faced a much fiercer liability regime, they actually spent less on security that UK banks did, and UK banks suffered more fraud. This appears to have been a moral-hazard effect, and was one of the anomalies that sparked interest in security economics. Secure systems need properly aligned incentives.

10.5 Credit Cards

The second theme in consumer payment systems is the credit card. For many years after their invention in the 1950s, credit cards were treated by most banks as a loss leader with which to attract high-value customers. Eventually, in most countries, the number of merchants and cardholders reached critical mass and the transaction volume suddenly took off. In Britain, it took almost twenty years before most banks found the business profitable; then all of a sudden it was extremely profitable. Payment systems have strong network externalities, just like communications technologies or computer platforms: they are two-sided markets in which the service provider must recruit enough merchants to appeal to cardholders, and vice versa. Because of this, and the huge investment involved in rolling out a new payment system to tens of thousands of banks, millions of merchants and billions of customers worldwide, any new payment mechanism is likely to take some time to get established. (The potentially interesting exceptions are where payment is bundled with some other service, such as with Google Checkout.)

Anyway, when you use a credit card to pay for a purchase in a store, the transaction flows from the merchant to his bank (the acquiring bank) which pays him after deducting a *merchant discount* of typically 4–5%. If the card was issued by a different bank, the transaction now flows to a switching center run by the brand (such as VISA) which takes a commission and passes it to the issuing bank for payment. Daily payments between the banks and the brands settle the net cash flows. The issuer also gets a slice of the merchant discount, but makes most of its money from extending credit to cardholders at rates that are usually much higher than the interbank rate.

10.5.1 Fraud

The risk of fraud using stolen cards was traditionally managed by a system of *hot card lists* and merchant *floor limits*. Each merchant gets a local hot card list — formerly on paper, now stored in his terminal — plus a limit set by their acquiring bank above which they have to call for authorization. The call center, or online service, which he uses for this has access to a national hot card list; above a higher limit, they will contact VISA or MasterCard which has a complete list of all hot cards being used internationally; and above a still higher limit, the transaction will be checked all the way back to the card issuer. Recently, the falling cost of communications has led to many transactions being authorised all the way back to the issuer, but there are still extensive fallback processing capabilities. This is because maintaining 99.9999% availability on a network, plus the capacity to handle peak transaction volumes on the Wednesday before Thanksgiving and the Saturday just before Christmas, still costs a whole lot more than the fraud from occasional offline and stand-in processing.

The introduction of *mail order and telephone order* (MOTO) transactions in the 1970s meant that the merchant did not have the customer present, and was not able to inspect the card. What was to stop someone ordering goods using a credit card number he'd picked up from a discarded receipt?

Banks managed the risk by using the expiry date as a password, lowering the floor limits, increasing the merchant discount and insisting on delivery to a cardholder address, which is supposed to be checked during authorization. But the main change was to shift liability so that the merchant bore the full risk of disputes. If you challenge an online credit card transaction (or in fact any transaction made under MOTO rules) then the full amount is immediately debited back to the merchant, together with a significant handling fee. The same procedure applies whether the debit is a fraud, a dispute or a return.

A recent development has been the 'Verified by VISA' program under which merchants can refer online credit-card transactions directly to the issuing bank, which can then authenticate the cardholder using its preferred method. The incentive for the merchant is that the transaction is then treated as a cardholder-present one, so the merchant is no longer at risk. The problem with this is that the quality of authentication offered by participating banks varies wildly. At the top of the scale are banks that use two-channel authentication: when you buy online you get a text message saying something like 'If you really want to pay Amazon.com $76.23, enter the code 4697 in your browser now'. At the bottom end are banks that ask you to enter your ATM PIN into the browser directly — thereby making their customers wide-open targets for particularly severe phishing attacks. There is a clear disincentive for the cardholder, who may now be held liable in many countries regardless of the quality of the local authentication methods.

Of course, even if you have the cardholder physically present, this doesn't guarantee that fraud will be rare. For many years, most fraud was done in person with stolen cards, and stores which got badly hit tended to be those selling goods that can be easily fenced, such as jewelry and consumer electronics. Banks responded by lowering their floor limits. More recently, as technical protection mechanisms have improved, there has been an increase in scams involving cards that were never received by genuine customers. This *pre-issue fraud* can involve thefts from the mail of the many 'pre-approved' cards which arrive in junk mail, or even applications made in the names of people who exist and are creditworthy, but are not aware of the application ('identity theft'). These attacks on the system are intrinsically hard to tackle using purely technical means.

10.5.2 Forgery

In the early 1980's, electronic terminals were introduced through which a sales clerk could swipe a card and get an authorization automatically. But the sales draft was still captured from the embossing, so crooks figured out how to re-encode the magnetic strip of a stolen card with the account number and expiry date of a valid card, which they often got by fishing out discarded receipts from the trash cans of expensive restaurants. A re-encoded card would authorize perfectly, but when the merchant submitted the draft for payment, the account number didn't match the authorization code (a six digit number typically generated by encrypting the account number, date and amount). So the merchants didn't get paid and raised hell.

Banks then introduced *terminal draft capture* where a sales draft is printed automatically using the data on the card strip. The crooks' response was a flood of forged cards, many produced by Triad gangs: between 1989 and 1992, magnetic strip counterfeiting grew from an occasional nuisance into half the total fraud losses [7]. VISA's response was *card verification values* (CVVs) — these are three-digit MACs computed on the card strip contents (account number, version number, expiry date) and written at the end of the strip. They worked well initially; in the first quarter of 1994, VISA International's fraud losses dropped by 15.5% while Mastercard's rose 67% [269]. So Mastercard adopted similar checksums too.

The crooks moved to *skimming* — operating businesses where genuine customer cards were swiped through an extra, unauthorized, terminal to grab a copy of the magnetic strip, which would then be re-encoded on a genuine card. The banks' response was intrusion detection systems, which in the first instance tried to identify criminal businesses by correlating the previous purchase histories of customers who complained.

In the late 1990's, credit card fraud rose sharply due to another simple innovation in criminal technology: the operators of the crooked businesses

which skim card data absorb the cost of the customer's transaction rather than billing it. You have a meal at a Mafia-owned restaurant, offer a card, sign the voucher, and fail to notice when the charge doesn't appear on your bill. Perhaps a year later, there is suddenly a huge bill for jewelry, electrical goods or even casino chips. By then you've completely forgotten about the meal, and the bank never had a record of it [501].

In the early 2000's, high-tech criminals became better organised as electronic crime became specialised. Phishing involved malware writers, botnet herders, phishing site operators and cash-out specialists, linked by black markets organised in chat rooms. This has spilled over from targeting online transactions to attacks on retail terminals. Fake terminals, and terminal tapping devices, used in the USA and Canada simply record mag-strip card and PIN data, which are used to make card clones for use in ATMs. In the Far East, wiretaps have been used to harvest card data wholesale [792]. Things are more complex in Europe which has introduced smartcards, but there are now plenty of devices that copy the EMV standard smartcards to mag-strip cards that are used in terminals that accept mag-strip transactions. Some of them use vulnerabilities in the EMV protocol, and so I'll come back to discuss them after I've described bank smartcard use in the next section.

10.5.3 Automatic Fraud Detection

There has been a lot of work since the mid-1990s on more sophisticated financial intrusion detection. Some generic systems do abuse detection using techniques such as neural networks, but it's unclear how effective they are. When fraud is down one year, it's hailed as a success for the latest fraud spotting system [101], while when the figures are up a few years later the vendors let the matter pass quietly [1191].

More convincing are projects undertaken by specific store chains that look for known patterns of misuse. For example, an electrical goods chain in the New York area observed that offender profiling (by age, sex, race and so on) was ineffective, and used purchase profiling instead to cut fraud by 82% in a year. Their technique involved not just being suspicious of high value purchases, but training staff to be careful when customers were careless about purchases and spent less than the usual amount of time discussing options and features. These factors can be monitored online too, but one important aspect of the New York success is harder for a web site: employee rewarding. Banks give a $50 reward per bad card captured, which many stores just keep — so their employees won't make an effort to spot cards or risk embarrassment by confronting a customer. In New York, some store staff were regularly earning a weekly bonus of $150 or more [840].

With online shopping, the only psychology the site designer can leverage is that of the villain. It has been suggested that an e-commerce site should have

an unreasonably expensive 'platinum' option that few genuine customers will want to buy. This performs two functions. First, it helps you to do basic purchase profiling. Second, it fits with the model of *Goldilocks pricing* developed by economists Hal Shapiro and Carl Varian, who point out that the real effect of airlines' offering first class fares is to boost sales of business class seats to travelers who can now convince their bosses (or themselves) that they are being 'economical' [1159]. Another idea is to have a carefully engineered response to suspect transactions: if you just say 'bad card, try another one' then the fraudster probably will. You may even end up being used by the crooks as an online service that tells them which of their stolen cards are on the hot list, and this can upset your bank (even though the banks are to blame for the system design). A better approach is claim that you're out of stock, so the bad guy will go elsewhere [1199].

As for electronic banking, it has recently become important to make intrusion detection systems work better with lower-level mechanisms. A good example is the real-time man-in-the-middle attack. After banks in the Netherlands handed out password calculators to their online banking customers, the response of the phishermen was to phish in real time: the mark would log on to the phishing site, which would then log on to the bank site and relay the challenge for the mark to enter into his calculator. The quick fix for this is to look out for large numbers of logons coming from the same IP address. It's likely to be a long struggle, of course; by next year, the bad guys may be using botnets to host the middleman software.

10.5.4 The Economics of Fraud

There's a lot of misinformation about credit card fraud, with statistics quoted selectively to make points. In one beautiful example, VISA was reported to have claimed that card fraud was up, and that card fraud was down, on the same day [598].

But a consistent pattern of figures can be dug out of the trade publications. The actual cost of credit card fraud during the 1990s was about 0.15% of all international transactions processed by VISA and Mastercard [1087], while national rates varied from 1% in America through 0.2% in UK to under 0.1% in France and Spain. The prevailing business culture has a large effect on the rate. U.S. banks, for example, are much more willing to send out huge junk mailings of pre-approved cards to increase their customer base, and write off the inevitable pre-issue fraud as a cost of doing business. In other countries, banks are more risk-averse.

The case of France is interesting, as it seems at first sight to be an exceptional case in which a particular technology has brought real benefits. French banks introduced chip cards for all domestic transactions in the late 1980's, and this reduced losses from 0.269% of turnover in 1987 to 0.04% in 1993 and 0.028%

in 1995. However, there is now an increasing amount of cross-border fraud. French villains use foreign magnetic stripe cards — particularly from the UK [498, 1087] — while French chip cards are used at merchants in non-chip countries [270]. But the biggest reduction in Europe was not in France but in Spain, where the policy was to reduce all merchant floor limits to zero and make all transactions online. This cut their losses from 0.21% of turnover in 1988 to 0.008% in 1991 [110].

The lessons appear to be that first, card fraud is cyclical as new defences are introduced and the villains learn to defeat them; and second, that the most complicated and expensive technological solution doesn't necessarily work best in the field. In fact, villains get smarter all the time. After the UK moved from magnetic strip cards to chipcards in 2005, it took less than eighteen months for the crooks to industrialise the process of moving stolen card data abroad: by 2007, as I'll discuss shortly.

10.5.5 Online Credit Card Fraud — the Hype and the Reality

Turning now from traditional credit card fraud to the online variety, I first helped the police investigate an online credit card fraud in 1987. In that case, the bad guy got a list of hot credit card numbers from his girlfriend who worked in a supermarket, and used them to buy software from companies in California, which he downloaded to order for his customers. This worked because hot card lists at the time carried only those cards which were being used fraudulently in that country; it also guaranteed that the bank would not be able to debit an innocent customer. As it happens, the criminal quit before there was enough evidence to nail him. A rainstorm washed away the riverbank opposite his house and exposed a hide which the police had built to stake him out.

From about 1995, there was great anxiety at the start of the dotcom boom that the use of credit cards on the Internet would lead to an avalanche of fraud, as 'evil hackers' intercepted emails and web forms and harvested credit card numbers by the million. These fears drove Microsoft and Netscape to introduce SSL/TLS to encrypt credit card transactions en route from browsers to web servers. (There was also a more secure protocol, SET, in which the browser would get a certificate from the card-issuing bank and would actually sign the transaction; this failed to take off as the designers didn't get the incentives right.)

The hype about risks to credit card numbers was overdone. Intercepting email is indeed possible but it's surprisingly difficult in practice — so much so that governments had to bully ISPs to install snooping devices on their networks to make court-authorized wiretaps easier [187]. I'll discuss this further in Part III. The actual threat is twofold. First, there's the growth of

phishing since 2004; there (as I remarked in Chapter 2) the issue is much more psychology than cryptography. TLS per se doesn't help, as the bad guys can also get certificates and encrypt the traffic.

Second, most of the credit card numbers that are traded online got into bad hands because someone hacked a merchant's computer. VISA had had rules for many years prohibiting merchants from storing credit card data once the transaction had been processed, but many merchants simply ignored them. From 2000, VISA added new rules that merchants had to install a firewall, keep security patches up-to-date, encrypt stored and transmitted data and regularly update antivirus software [1262]. These were also not enforced. The latest set of rules, the Payment Card Industry Data Security Standard, are a joint effort by VISA and Mastercard, and supported by the other brands too; they say much the same things, and we'll have to wait and see whether the enforcement is any better. 'PCI', as the new system's called, certainly seems to be causing some pain; in October 2007, the U.S. National Retail Federation asked credit card companies to stop forcing retailers to store credit card data at all (at present they are supposed to store card numbers temporarily in case of chargebacks) [1296].

The real incentives facing merchants are, first, the cost of disputes, and second, the security-breach disclosure laws that are (in 2007) in force in 34 U.S. states and that are contemplated as a European Directive. Disclosure laws have had a very definite effect in the USA as the stock prices of companies suffering a breach can fall several percent. As for disputes, consumer protection laws in many countries make it easy to repudiate a transaction. Basically all the customer has to do is call the credit card company and say 'I didn't authorize that' and the merchant is saddled with the bill. This was workable in the days when almost all credit card transactions took place locally and most were for significant amounts. If a customer fraudulently repudiated a transaction, the merchant would pursue them through the courts and harrass them using local credit reference agencies. In addition, the banks' systems are often quite capable of verifying local cardholder addresses.

But the Internet differs from the old mail order/telephone order regime in that many transactions are international, amounts are small, and verifying overseas addresses via the credit card system is problematic. Often all the call center operator can do is check that the merchant seems confident when reading an address in the right country. So the opportunity for repudiating transactions — and getting away with it — is hugely increased. There are particularly high rates of repudiation of payment to porn sites. No doubt some of these disputes happen when a transaction made under the influence of a flush of hormones turns up on the family credit card bill and the cardholder has to repudiate it to save his marriage; but many are the result of blatant fraud by operators. A common scam was to offer a 'free tour' of the site and demand a credit card number, supposedly to verify that the user was over 18,

and then bill him anyway. Some sites billed other consumers who have never visited them at all [620]. Even apparently large and 'respectable' web sites like `playboy.com` were criticised for such practices, and at the bottom end of the porn industry, things are atrocious.

The main brake on wicked websites is the credit-card chargeback. A bank will typically charge the merchant $100–200 in fees for each of them, as well as debiting the transaction amount from his account. So if more than a small percentage of the transactions on your are challenged by customers, your margins will be eroded. If chargebacks go over perhaps 10%, your bank may withdraw your card acquisition service. This has happened to a number of porn sites; a more prosaic example was the collapse of sportswear merchant `boo.com` because they had too many returns: their business model assumed a no-quibble exchange or refund policy, similar to those operated by high-street discount clothing stores. Yet more of their shipments than they'd expected were the wrong size, or the wrong colour, or just didn't appeal to the customers. Refunds are cheaper than chargebacks, but still, the credit card penalties broke the company [1199]. Chargebacks also motivate merchants to take case — to beware of odd orders (e.g. for four watches), orders from dodgy countries, customers using free email services, requests for expedited delivery, and so on. But leaving the bulk of the liability for mail-order transactions with them is suboptimal: the banks know much more about fraud patterns.

This history suggests that purely technological fixes may not be easy, and that the most effective controls will be at least partly procedural. Some card issuers offer credit card numbers that can be used once only; as they issue them one at a time to customers via their web site, this also helps drive lots of traffic to their advertisers [324]. Other banks have found that they get better results by investing in address verification [102]. However the big investment in the last few years has been in new card technologies, with Europe replacing both credit cards and debit cards with smartcards complying with the EMV 'chip and PIN' standard, while U.S. banks are starting to roll out bank cards based on RFID.

10.6 Smartcard-Based Banking

In the 1960s and 70s, various people proposed putting integrated circuits in bank cards. The Germans consider the smartcard to have been invented by Helmut Gröttrup and Jürgen Dethloff in 1968, when they proposed and patented putting a custom IC in a card 1968; the French credit Roland Moreno, who proposed putting memory chips in cards in 1973, and Michel Ugon who proposed adding a microprocessor in 1977. The French company Honeywell-Bull patented a chip containing memory, a microcontroller and everything else needed to do transactions in 1982; they started being used in French

pay phones in 1983; and in banking from the mid-1980s, as discussed in section 10.5.4 above.

Smartcards were marketed from the beginning as the French contribution to the information age, and the nascent industry got huge government subsidies. In the rest of the world, progress was slower. There were numerous pilot projects in which smartcards were tried out with different protocols, and in different applications. I already mentioned the COPAC system at 3.8.1; we developed this in 1991–2 for use in countries with poor telecommunications, and it sold best in Russia. Norway's commercial banks started issuing smartcards in 1986 but its savings banks refused to; when the central bank pressured the banks to unite on a common technology, mag stripe won and smartcards were withdrawn in 1995. Britain's NatWest Bank developed the Mondex electronic purse system in the early 90s, piloted it in Swindon, then sold it to Mastercard. There was a patent fight between VISA (which had bought the COPAC rights) and Mastercard. The Belgian banks implemented an electronic purse called 'Proton' for low-value payments to devices like parking meters; the Germans followed with 'Geldkarte' which became the European standard EN1546 and is now also available as the 'Moneo' electronic purse in France.

Offline systems such as Mondex had problems dealing with broken cards. If the back-end system doesn't do full balancing, then when a customer complains that a card has stopped working, all the bank can do is either to refund the amount the customer claims was on the card, or tell her to get lost; so most modern systems do balancing, which means they aren't as cheap to operate as one might have hoped. All this was good learning experience. But for a payment card to be truly useful, it has to work internationally — and especially so in Europe with many small countries jammed up close together, where even a one-hour shopping trip in the car may involve international travel. So the banks finally got together with their suppliers and hammered out a standard.

10.6.1 EMV

The EMV standards are named after the participating institutions Europay, Mastercard and VISA (Europay developed the Belgian Proton card). As of 2007, several hundred million European cardholders now have debit and credit cards that conform to this standard, and can be used more or less interoperably in the UK, Ireland, France, Germany and other participating countries. In English speaking countries such as the UK and Ireland, EMV has been branded as 'chip and PIN' (although the standards do also support signature-based transactions). The standards' proponents hope that they will become the worldwide norm for card payments, although this is not quite a done deal: Japan and increasingly the USA are adopting RFID standards for contactless payment, which I'll discuss in the next section. Anyway, in much

of the world, the EMV standards act as a 'fraud bulldozer', moving around the payment-systems landscape so that some types of fraud become less common and others more so.

The EMV protocol documents [429] are not so much a single protocol as a suite of protocols. The VISA version of the protocols alone come to more than 3,600 pages, and these are only the compatibility aspects — there are further documents specific to individual banks. Specifications this complex cannot be expected to be bug-free, and I'll describe some of the bugs in the following sections. The most obvious problem is that the documents allow many options, some of which are dangerous, either individually or in combination. So EMV can be thought of as a construction kit for building payment systems, with which one can build systems that are quite secure, or very insecure, depending on how various parameters are set, what sort of fallback modes are invoked on failure, and what other systems are hooked up.

In order to understand this, we need to look briefly at the EMV mechanisms. Each customer card contains a smartcard chip with the capability to verify a PIN and authenticate a transaction. The cards come in two types: low-cost cards that do only symmetric cryptography and use a set of protocols known as static data authentication (SDA); and more expensive cards that can generate digital signatures, supporting protocols called dynamic data authentication (DDA) and combined data authentication (CDA).

10.6.1.1 Static Data Authentication

SDA is the default EMV protocol, and it works as follows. The customer puts her card into the 'chip and PIN' terminal to which it sends a digital certificate, account number and the other data found on the old-fashioned magnetic strip, plus a digital signature from the card-issuing bank (the bank chooses which data items to sign). The terminal verifies the signature and the merchant enters the payment amount; the terminal solicits the PIN; the customer enters it; and it's sent in clear to the card. If the PIN is accepted, the card tells the terminal that all's well and generates a MAC, called an 'application data cryptogram', on the supplied data (merchant ID, amount, serial number, nonce and so on). The key used to compute this MAC is shared between the card and the customer's bank, and so it can only be verified by the issuing bank. (The bank could thus use any algorithm it liked, but the default is DES-CBC-MAC with triple-DES for the last block.) Also, the only way the terminal can check that the transaction is genuine is by going online and getting an acknowledgement. As this isn't always convenient, some merchants have a 'floor limit' below which offline transactions are permitted.

This protocol has a number of vulnerabilities that are by now well known. The most commonly-exploited one is backwards compatibility with magnetic strip cards: as the certificate contains all the information needed to forge a

mag-strip card, and as the introduction of chip and PIN means that people now enter PINs everywhere rather than just at ATMs, a number of gangs have used assorted sniffers to collect card data from terminals and collected money using mag-strip forgeries. Many ATMs and merchant terminals even in the EMV adopter countries will fall back to mag-strip processing for reliability reasons, and there are many countries — from the USA to Thailand — that haven't adopted EMV at all. There are two flavours of attack: where the PIN is harvested along with the card details, and where it's harvested separately.

First, where the card reader and the PIN pad are separate devices, then a wiretap between them will get PINs as well as card data. Since 2005 there have been reports of sniffing devices, made in Eastern Europe, that have been found in stores in Italy; they harvest the card and PIN data and send it by SMS to the villains who installed them. This may be done under cover of a false-pretext 'maintenance' visit or by corrupt store staff. There have also been reports of card cloning at petrol stations after PIN pads were replaced with tampered ones; although these cases are waiting to come to trial as I write, I investigated the tamper-resistance of PIN pads with two colleagues and we found that the leading makes were very easy to compromise.

For example, the Ingenico i3300, one of the most widely-deployed terminals in the UK in 2007, suffers from a series of design flaws. Its rear has a user-accessible compartment, shown in Figure 10.4, for the insertion of optional extra components. This space is not designed to be tamper-proof, and when covered it cannot be inspected by the cardholder even if she handles the device. This compartment gives access to the bottom layer of the circuit board. This does not give direct access to sensitive data — but, curiously, the designers opted to provide the attacker 1 mm diameter holes (used for positioning the optional components) and vias through the circuit board. From there, a simple metal hook can tap the serial data line. We found that a 1 mm diameter via, carrying the serial data signal, is easily accessed using a bent paperclip. This can be inserted through a hole in the plastic surrounding the internal compartment, and does not leave any external marks. The effect is that the attacker can design a small wiretap circuit that sits invisibly inside the terminal and gathers both card and PIN data. This circuit can be powered from the terminal itself and could contain a small mobile phone to SMS the booty to its makers.

Britain had an epidemic of fraud in 2006–7 apparently involving sniffer devices inserted into the wiring between card terminals and branch servers in petrol stations in the UK. As the card readers generally have integral PIN pads in this application, the PINs may be harvested by eye by petrol-station staff, many of whom are Tamils who arrived as refugees from the civil war in Sri Lanka. It's said that the Tamil Tigers — a terrorist group — intimidates them into participating. This was discovered when Thai police caught men in Phuket with 5,000 forged UK debit and credit cards, copied on to 'white plastic'.



Figure 10.4: A rigid wire is inserted through a hole in the Ingenico's concealed compartment wall to intercept the smartcard data. The front of the device is shown on the top right.

Attacks exploiting the fact that the MAC can't be read by the merchant include 'yescards'. These are cards programmed to accept any PIN (hence the name) and to participate in the EMV protocol using an externally-supplied certificate, returning random values for the MAC. A villain with a yescard and access to genuine card certificates — perhaps through a wiretap on a merchant terminal — can copy a cert to a yescard, take it to a merchant with a reasonable floor limit, and do a transaction using any PIN. This attack has been reported in France and suspected in the UK [122]; it's pushing France towards a move from SDA to DDA. However, most such frauds in Britain still use magnetic strip fallback: many ATMs and merchants use the strip if the chip is not working.

Another family of problems with EMV has to do with authentication methods. Each card, and each terminal, has a list of preferred methods, which might say in effect: 'first try online PIN verification, and if that's not supported use local cleartext PIN verification, and if that's not possible then you don't need to authenticate the customer at all'. It might at first sight be surprising that 'no authentication' is ever an option, but it's frequently there, in order to support devices such as parking ticket vending machines that don't have

PIN pads. One glitch is that the list of authentication methods isn't itself authenticated, so a bad man might manipulate it in a false-terminal or relay attack. Another possibility is to have two cards: your own card, for which you know the PIN, and a stolen card for which you don't, slotted into a device that lets you switch between them. You present the first card to the terminal and verify the PIN; you then present the transaction to the stolen card with the verification method changed to 'no authentication'. The stolen card computes the MAC and gets debited. The bank then maintains to the victim that as his chip was read and a PIN was used, he's liable for the money.

One countermeasure being contemplated is to insert the verification method into the transaction data; another is to reprogram cards to remove 'no authentication' from the list of acceptable options. If your bank takes the latter option, you'd better keep some change in your car ashtray! Yet another is to reprogram customer cards so that 'no authentication' works only up to some limit, say $200.

The fact that banks can now reprogram customers' cards in the field is also novel. The mechanism uses the shared key material and kicks in when the card's at an online terminal such as an ATM. One serious bug we discovered is that the encryption used to protect these messages is so poorly implemented that bank insiders can easily extract the keys from the hardware security modules [12]. I'll discuss this kind of vulnerability at greater length when we dive into the thicket of API security. Remote reprogrammability was pioneered by the pay-TV stations in their wars against pirates who cloned their smartcards; it can be a powerful tool, but it can also be badly misused. For example, it opens the possibility of a disgruntled insider launching a service-denial attack that rapidly wipes out all a bank's customers' cards.

However, such bankers' nightmares aside, the practical security of EMV depends to a great extent on implementation details such as the extent to which fallback magnetic-strip processing is available in local ATMs, the proportion of local shops open to various kinds skimmer attacks (whether because of personnel vulnerability factors or because there are many store chains using separate card readers and PIN pads), and — as always — incentives. Do the banks carry the can for fraud as in the USA, which makes them take care, or are they able to dump the costs on merchants and cardholders, as in much of Europe, which blunts their motivation to insist on high standards? Indeed, in some countries — notably the UK — banks appear to have seen EMV not so much as a fraud-reduction technology but a liability-engineering one. In the old days they generally paid for fraud in signature-based transactions but often blamed the customer for the PIN-based ones ('your PIN was used so you must have been negligent'). The attractions of changing most in-store transactions from signatures to PINs were obvious.

The bottom-line question is, of course, whether it paid for itself. In Britain it hasn't. Fraud rose initially, thanks to the much larger number of cards stolen

from the mail during the changeover period; local fraud has been said to fall since, though this has been achieved with the help of some fairly blatant manipulation of the numbers. For example, bank customers were stopped from reporting card fraud to the police from April 2007; frauds must be reported to the bank. Oh, and the banks have taken over much of the financing of the small police unit that does still investigate card fraud. This helps the government massage the crime statistics downward, and lets the banking industry control such prosecutions as do happen. Meanwhile overseas fraud has rocketed, thanks to the Tamil Tigers and to the vigorous international trade in stolen card numbers. The net effect was that by October 2007 fraud was up 26% on the previous year [83].

10.6.1.2 Dynamic Data Authentication

DDA is a more complex EMV protocol, used in Germany. It differs from SDA in that the cards are capable of doing public-key cryptography: each has an RSA public-private keypair, with the public part embedded in the card certificate. The cryptography is used for two functions. First, when the card is first inserted into the terminal, it's sent a nonce, which it signs. This assures the terminal that the card is present (somewhere). The terminal then sends the transaction data plus the PIN encrypted using the card's public key, and the card returns the application data cryptogram as before.

This provides a small amount of extra protection (though at the cost of a more expensive card — perhaps $1 each in volume rather than 50c). In particular, the PIN doesn't travel in the clear between the PIN pad and the terminal, so the Hungarian skimmer won't work. (The Tamil Tiger attack still works as in that case the shop assistant collected the PIN using the mark 1 eyeball.)

There are still significant vulnerabilities though. Even assuming that the cryptography is sound, that the software's properly written, that the interfaces are well-designed, and that the cards are too expensive to clone, the lack of any hard link between the public-key operation of proving freshness and accepting the PIN, and the shared-key operation of computing the MAC, means that the two-card attack could still be perpetrated.

10.6.1.3 Combined Data Authentication

CDA is the Rolls-Royce of EMV protocols. it's like DDA except that the card also computes a signature on the MAC. This ties the transaction data to the public key and to the fact that a PIN verification was performed (assuming, that is, the bank selected the option of including a PIN-verification flag in in the transaction data).

But the protocol still isn't bulletproof, as the customer has no trustworthy user interface. A wicked merchant could mount a false front over a payment terminal so that the customer would think she was paying $5 for a box of chocolates when in reality she was authorising a payment of $2500. (With over 200 approved terminal types, it's unreasonable to expect customers to tell a genuine terminal from a bogus one.) A bad merchant can also mount a *relay* attack. Two students of mine implemented this as a proof-of-concept for a TV program; a bogus terminal in a café was hooked up via WiFi and a laptop to a bogus card. When a sucker in the café went to pay £5 for his cake to a till operated by one student, his card was connected up to the false card carried by the other, who was lingering in a bookstore waiting to buy a book for £50. The £50 transaction went through successfully [401, 915].

An interesting possibility for relay attacks is to provide deniability in money-laundering. EMV transactions are now routinely used for high-value transactions, such as buying cars and yachts, and as they're between bank accounts directly they attract little attention from the authorities. So a bad man in London wanting to pay $100,000 to a crook in Moscow could simply arrange to buy him a BMW. With relaying, he could get an alibi by making this transaction just after a local one with witnesses; he might take his Member of Parliament out to a meal. If challenged he could claim that the car purchase was a fraud, and the police could have a hard time proving a transaction relay in the face of bank denials that such things happen.

There also are the usual 'social engineering' attacks; for example, a dishonest merchant observes the customer entering the PIN and then steals the card, whether by palming it and giving her back a previously-stolen card issued by the same bank, or by following her and stealing it from her bag (or snatching her bag). Such attacks have happened since the early days of bank cards. They can be automated: a bogus vending machine might retain a card and give back a previously-stolen one; or more pernicious still, use a card in its temporary possession to make a large online purchase. There is a nasty variant for systems that use the same card for online banking: the wicked parking meter goes online and sends all your money to Russia in the few seconds when you thought it was simply debiting you $2.50.

10.6.2 RFID

In the USA, where thanks to the Federal Reserve the incentives facing banks and merchants are less perverted, the banking industry has remained unconvinced that the multibillion-dollar costs of moving to EMV would be justified by any reductions in losses. Rather than moving to EMV, the industry has preferred to skip a generation and wait for the next payment method — so-called 'RFID' or 'contactless' payment cards.

Contactless payment has been a reality for a few years in a number of transport systems from London to Tokyo. When you buy a season ticket you get a 'contactless' smartcard — a device using short-range radio or magnetic signals to communicate with a terminal embedded in the turnstile. The automation allows a greater variety of deals to be sold than the traditional season ticket too; you can pay as you go and top up your credit as you need it. Turning this technology into a general-purpose payment instrument has a number of advantages.

One interesting new development is NFC — near-field communications. NFC is a means of building contactless/RFID communications capability into devices such as mobile phones. This means that your phone can double as your season ticket; at Japanese subway turnstiles, you can just touch your phone on the payment pad in order to get through. Small payments can be processed quickly and automatically, while for larger payments the phone can provide the trustworthy use interface whose lack is such a serious problem for EMV-style payment systems.

There are quite a few problems to be tackled. First, if RFID payment cards can also be used in traditional credit-card systems, then a bad man can harvest credit card numbers, security codes and expiry dates by doing RFID transactions with victims' cards as he brushes past them in the street — or by reading cards that have been sent to customers in the mail, without opening the envelopes [601].

Second, there are practical problems to do with RF propagation: if you have three cards in your wallet and you wave the wallet over a subway turnstile, which of them gets debited? (All of them?)

Third, our old friend the middleperson attack (and his evil twin the forwarding attack) return with a vengeance. When my students implemented the forwarding attack on EMV, they had to spend several weeks building custom electronics for the wicked reader and the bogus card. Once RFID and NFC become pervasive, making equipment is just a programming task, and not even a very tricky one. Any two NFC phones should be able to act in concert as the false terminal and the wicked card. And it appears that no-one's taking ownership of the problem of securing RFID payments; each of the players in the business appears to be hoping that someone else will solve the problem [56].

10.7 Home Banking and Money Laundering

After credit and debit cards, the third thread of electronic banking at the consumer-level is home banking. In 1985, the first such service in the world was offered by the Bank of Scotland, whose customers could use Prestel, a proprietary email system operated by British Telecom, to make payments.

When Steve Gold and Robert Schifreen hacked Prestel — as described in the chapter on passwords — it initially terrified the press and the bankers. They realised that the hackers could easily have captured and altered transactions. But once the dust settled and people thought through the detail, they realised there was little real risk. The system allowed only payments between your own accounts and to accounts you'd previously notified to the bank, such as you gas and electricity suppliers.

This pattern, of high-profile hacks — which caused great consternation but which on sober reflection turned out to be not really a big deal — has continued ever since.

To resume this brief history, the late 1980's and early 1990's saw the rapid growth of call centers, which — despite all the hoopla about the web — still probably remain in 2007 the largest delivery channel for business-to-consumer electronic commerce[3]. The driver was cost cutting: call centres are cheaper than bank branches. Round about 1999, banks rushed to build websites in order to cut costs still further, and in so doing they also cut corners. The bank-end controls, which limited who you could pay and how much, were abolished amid the general euphoria, and as we've seen, the phishermen arrived in earnest from 2004. Phishermen are not the only threat, although they appear to be the main one in the English-speaking world; in Continental Europe, there is some suspicion that keyloggers may be responsible for more account takeovers.

As I mentioned in the chapter on passwords, the main change is the increasing specialisation of gangs involved in financial crime. One firm writes the malware, another herds the botnet; another does the 'Madison Avenue' job of writing the spam; and there are specialists who will accept hot money and launder it. (Note that if it becomes too easy for bent programmers to make contact with capable money launderers, this could have a material effect on the fraud risk faced by systems such as SWIFT. It would undermine our current implicit separation-of-duty policy in that the techies who know how to hack the message queue don't understand how to get money out of the system.)

The hot topic in 2007 is how to stop phishermen getting away with money stolen from compromised bank accounts, and a phisherman faces essentially the same money-laundering problem as a bent bank programmer. Until May 2007, the preferred route was eGold, a company operated from Florida but with a legal domicile in the Caribbean, which offered unregulated electronic payment services. The attraction to the villains was that eGold payments were irreversible; their staff would stonewall bank investigators who were hot on the trail of stolen money. eGold duly got raided by the FBI and indicted.

[3]I'm not aware of any global figures, but, to get some indication, the UK has 6000 call centres employing half a million people; and Lloyds TSB, a large high-street bank, had 16 million accounts of whom most use telephone banking but under 2 million used online banking regularly in 2005.

The villains' second recourse was to send money through banks in Finland to their subsidiaries in the Baltic states and on to Russia; no doubt the Finnish regulators will have cleaned this up by the time this book appears. The third choice was wire-transfer firms like Western Union, and various electronic money services in Russia and the Middle East. I wrote a survey of this for the U.S. Federal Reserve; see [55].

At the time of writing, the favourite modus operandi of the folks who launder money for the phishermen is to recruit *mules* to act as cut-outs when sending money from compromised accounts to Western Union [545]. Mules are attracted by spam offering jobs in which they work from home and earn a commission. They're told they will be an agent for a foreign company; their work is to receive several payments a week, deduct their own commission, and then send the balance onward via Western Union. Money duly arrives in their account and they pay most of it onwards. After a few days, the bank from whose customer the money was stolen notices and reverses out the credit. The poor mule is left with a huge overdraft and finds that he can't get the money back from Western Union. In the English-speaking world, that's just about it; in Germany, mules are also prosecuted and jailed. (Even some German bankers consider this to be harsh, as the typical mule is an elderly working-class person who grew up under the communists in East Germany and doesn't even understand capitalism, let alone the Internet.) As the word gets round, mule recruitment appears to be getting more and more difficult — if we can judge from the rapidly increasing quantities of mule-recruitment spam during the middle of 2007. Note in passing that as the real victims of many phishing attacks are the poor mules, this implies that phishing losses as reported by the banks may be a significant underestimate.

Another thing we've learned from watching the phishermen over the past few years is that the most effective countermeasure isn't improving authentication, but sharpening up asset recovery. Of the £35m lost by UK banks in 2006, over £33m was lost by one bank. Its competitors assure me that the secret of their success is that they spot account takeovers quickly and follow them up aggressively; if money's sent to a mule's account, he may find his account frozen before he can get to Western Union. So the phishermen avoid them. This emphasises once more the importance of sound back-end controls. The authentication mechanisms alone can't do the job; you need to make the audit and intrusion-detection mechanisms work together with them.

Another thing we've learned is that liability dumping is not just pervasive but bad for security. The rush to online banking led many financial institutions to adopt terms and conditions under which their records of an electronic transaction are definitive; this conflicts with consumer law and traditional banking practice [201]. Unfortunately, the EU's 2007 Payment Services Directive allows all European banks to set dispute resolution procedures in their terms and conditions, and undermines the incentive to deal with the problems. The

ability of banks to blame their customers for fraud has also led to many sloppy practices. In the UK, when it turned out that people who'd accessed electronic services at Barclays Bank via a public terminal could be hacked by the next user pressing the 'back' button on the browser, they tried to blame customers for not clearing their web caches [1249]. (If opposing that in court, I'd have great fun finding out how many of Barclays' branch managers knew what a cache is, and the precise date on which the bank's directors had it brought to their attention that such knowledge is now essential to the proper conduct of retail banking business.)

10.8 Summary

Banking systems are interesting in a number of ways.

Bookkeeping applications give us a mature example of systems whose security is oriented towards authenticity and accountability rather than confidentiality. Their protection goal is to prevent and detect frauds being committed by dishonest insiders. The Clark-Wilson security policy provides a model of how they operate. It can be summarized as: *'all transactions must preserve an invariant of the system, namely that the books must balance (so a negative entry in one ledger must be balanced by a positive entry in another one); some transactions must be performed by two or more staff members; and records of transactions cannot be destroyed after they are committed'*. This was based on time-honoured bookkeeping procedures, and led the research community to consider systems other than variants of Bell-LaPadula.

But manual bookkeeping systems use more than just dual control. Although some systems do need transactions to be authorised in parallel by two or more staff, a separation of duty policy more often works in series, in that different people do different things to each transaction as it passes through the system. Designing bookkeeping systems to do this well is a hard and often neglected problem, that involves input from many disciplines. Another common requirement is non-repudiation — that principals should be able to generate, retain and use evidence about the relevant actions of other principals.

The other major banking application, remote payment, is increasingly critical to commerce of all kinds. In fact, wire transfers of money go back to the middle of the Victorian era. Because there is an obvious motive to attack these systems, and villains who steal large amounts and get caught are generally prosecuted, payment systems are a valuable source of information about what goes wrong. Their loss history teaches us the importance of minimizing the background error rate, preventing procedural attacks that defeat technical controls (such as thefts of ATM cards from the mail), and having adequate controls to deter and detect internal fraud.

Payment systems have also played a significant role in the development and application of cryptology. One innovation was the idea that cryptography could be used to confine the critical part of the application to a trusted computing base consisting of tamper-resistant processors — an approach since used in many other applications.

The recent adoption of various kinds of smartcard-based payment mechanism - EMV in Europe, RFID in the USA and Japan — is changing the fraud landscape. It opens up the possibility of more secure payment systems, but this is not at all guaranteed. In each case, the platform merely provides a toolkit, with which banks and merchants can implement good systems, or awful ones.

Finally, the recent history of attacks on electronic banking systems by means of account takeover — by phishermen, and to a lesser extent using keyloggers — presents a challenge that may over time become deeper and more pervasive than previous challenges. Up till now, banking folks — from the operations guys up to the regulators and down to the system designers — saw the mission as maintaining the integrity of the financial system. We may have to come to terms with a world in which perhaps one customer account in ten thousand or so has been compromised at any given time. Instead, we will have to talk about the resilience of the financial system.

Research Problems

Designing internal controls is still pre-scientific; we could do with tools to help us do it in a more systematic, less error-prone way. Accountants, lawyers, financial market regulators and system engineers all seem to feel that this is someone else's responsibility. This is a striking opportunity to do multidisciplinary research that has the potential to be outstandingly useful.

At a more techie level, we don't even fully understand stateful access control systems, such as Clark-Wilson and Chinese Wall. To what extent does one do more than the other on the separation-of-duty front? How should dual control systems be designed anyway? How much of the authorization logic can we abstract out of application code into middleware? Can we separate policy and implementation to make enterprise-wide policies easier to administer?

As for robustness of cryptographic systems, the usability of security mechanisms, and assurance generally, these are huge topics and still only partially mapped. Robustness and assurance are partially understood, but usability is still a very grey area. There are many more mathematicians active in security research than applied psychologists, and it shows.

Finally, if account takeover is going to become pervasive, and a typical bank has 0.01% of its customer accounts under the control of the Russian mafia at any one time, what are the implications? I said that instead of talking about the integrity of the financial system, we have to talk about its resilience. But what

does this mean? No-one's quite sure what resilience implies in this context. Recent experience suggests that extending the principles of internal control and combining them with aggressive fraud detection and asset recovery could be a good place for engineers to start. But what are the broader implications? Personally I suspect that our regulatory approach to money laundering needs a thorough overhaul. The measures introduced in a panic after the passage of the U.S. Patriot Act have been counterproductive, and perhaps emotions have subsided now to the point that governments can be more rational. But what should we do? Should Western Union be closed down by the Feds, as eGold was? That's probably excessive — but I believe that laundering controls should be less obsessive about identity, and more concerned about asset recovery.

Further Reading

I don't know of any comprehensive book on banking computer systems, although there are many papers on specific payment systems available from the Bank for International Settlements [114]. When it comes to developing robust management controls and business processes that limit the amount of damage that any one staff member could do, there is a striking lack of hard material (especially given the demands of Gramm-Leach-Bliley and Sarbanes-Oxley). There was one academic conference in 1997 [657], and the best book I know of on the design and evolution of internal controls, by Steven Root, predates the Enron saga [1082]. As the interpretation put on these new laws by the big four accountancy firms makes the weather on internal controls, and as their gold-plating costs the economy so much, it is certainly in the public interest for more to be published and discussed. I'll revisit this topic in Part III.

For the specifics of financial transaction processing systems, the cited articles [33, 34] provide a basic introduction. More comprehensive, if somewhat dated, is [354] while [525] describes the CIRRUS network as of the mid-80s. The transcript of Paul Stubbs' trial gives a snapshot of the internal controls in the better electronic banking systems in 2004 [975]; for conspicuous internal control failure, see for example [1038]. The most informative public domain source on the technology — though somewhat heavy going — may be the huge online manuals for standards such as EMV [429] and the equipment that supports it, such as the IBM 4758 and CCA [641].

EXHIBIT D

Operating Systems
R. Stockton Gaines Editor

Password Security: A Case History

Robert Morris and Ken Thompson
Bell Laboratories

This paper describes the history of the design of the password security scheme on a remotely accessed time-sharing system. The present design was the result of countering observed attempts to penetrate the system. The result is a compromise between extreme security and ease of use.

Key Words and Phrases: operating systems, passwords, computer security
CR Categories: 2.41, 4.35

Introduction

Password security on the UNIX (a trademark of Bell Laboratories) time-sharing system [3] is provided by a collection of programs whose elaborate and strange design is the outgrowth of many years of experience with earlier versions. To help develop a secure system, we have had a continuing competition to devise new ways to attack the security of the system (the bad guy) and, at the same time, to devise new techniques to resist the new attacks (the good guy). This competition has been in the same vein as the competition of long standing between manufacturers of armor plate and those of armor-piercing shells. For this reason, the description that follows will trace the history of the password system rather than simply presenting the program in its current state. In this way, the reasons for the design will be made clearer, as the design cannot be understood without also understanding the potential attacks.

Permission to copy without fee all or part of this material is granted provided that the copies are not made or distributed for direct commercial advantage, the ACM copyright notice and the title of the publication and its date appear, and notice is given that copying is by permission of the Association for Computing Machinery. To copy otherwise, or to republish, requires a fee and/or specific permission.

Authors' present address: R. Morris and K. Thompson, Bell Laboratories, 600 Mountain Avenue, Murray Hill, NJ 07974.

© 1979 ACM 0001-0782/79/1100-0594 $00.75.

An underlying goal has been to provide password security at minimal inconvenience to the users of the system. For example, those who want to run a completely open system without passwords, or to have passwords only at the option of the individual users, are able to do so, while those who require all of their users to have passwords gain a high degree of security against penetration of the system by unauthorized users.

The password system must be able not only to prevent any access to the system by unauthorized users (i.e., prevent them from logging in at all), but it must also prevent users who are already logged in from doing things that they are not authorized to do. The so-called "super-user" password on the UNIX system, for example, is especially critical because the super-user has all sorts of permissions and has essentially unlimited access to all system resources.

Password security is of course only one component of overall system security, but it is an essential component. Experience has shown that attempts to penetrate remote-access systems have been astonishingly sophisticated.

Remote-access systems are peculiarly vulnerable to penetration by outsiders as there are threats at the remote terminal, along the communications link, as well as at the computer itself. Although the security of a password encryption algorithm is an interesting intellectual and mathematical problem, it is only one tiny facet of a very large problem. In practice, physical security of the computer, communications security of the communications link, and physical control of the computer itself loom as far more important issues. Perhaps most important of all is control over the actions of ex-employees, since they are not under any direct control and they may have intimate knowledge about the system, its resources, and methods of access. Good system security involves realistic evaluation of the risks not only of deliberate attacks but also of casual authorized access and accidental disclosure.

Prologue

The UNIX system was first implemented with a password file that contained the actual passwords of all the users, and for that reason the password file had to be heavily protected against being either read or written. Although historically, this had been the technique used for remote-access systems, it was completely unsatisfactory for several reasons.

The technique is excessively vulnerable to lapses in security. Temporary loss of protection can occur when the password file is being edited or otherwise modified. There is no way to prevent the making of copies by privileged users. Experience with several earlier remote-access systems showed that such lapses occur with frightening frequency. Perhaps the most memorable such occasion occurred in the early 60s at a time when one of

Plaid Ex1003
Page 269 of 372

the authors (Morris) happened to be using the system. A system administrator on the CTSS system at MIT was editing the password file and another system administrator was editing the daily message that is printed on everyone's terminal on login. Due to a software design error, the temporary editor files of the two users were interchanged and thus, for a time, the password file was printed on every terminal when it was logged in.

Once such a lapse in security has been discovered, everyone's password must be changed, usually simultaneously, at a considerable administrative cost. This is not a great matter, but far more serious is the high probability of such lapses going unnoticed by the system administrators.

Security against unauthorized disclosure of the passwords was, in the last analysis, impossible with this system because, for example, if the contents of the file system are put on to magnetic tape for backup, as they must be, then anyone who has physical access to the tape can read anything on it with no restriction.

Many programs must get information of various kinds about the users of the system, and these programs in general should have no special permission to read the password file. The information which should have been in the password file actually was distributed (or replicated) into a number of files, all of which had to be updated whenever a user was added to or dropped from the system.

The First Scheme

The obvious solution is to arrange that the passwords not appear in the system at all, and it is not difficult to decide that this can be done by encrypting each user's password, putting only the encrypted form in the password file, and throwing away his original password (the one that he typed in). When the user later tries to log in to the system, the password that he types is encrypted and compared with the encrypted version in the password file. If the two match, his login attempt is accepted. Such a scheme was first described in [4, p. 91ff.]. It also seemed advisable to devise a system in which neither the password file nor the password program itself needed to be protected against being read by anyone.

All that was needed to implement these ideas was to find a means of encryption that was very difficult to invert, even when the encryption program is available. Most of the standard encryption methods used (in the past) for encryption of messages are rather easy to invert. A convenient and rather good encryption program happened to exist on the system at the time; it simulated the M-209 cipher machine [1] used by the U.S. Army during World War II. It turned out that the M-209 program was usable, but with a given key, the ciphers produced by this program are trivial to invert. It is a much more difficult matter to find out the key given the cleartext input and the enciphered output of the program. Therefore, the password was used not as the text to be encrypted but as the key, and a constant was encrypted using this key. The encrypted result was entered into the password file.

Attacks on the First Approach

Suppose that the bad guy has available the text of the password encryption program and the complete password file. Suppose also that he has substantial computing capacity at his disposal.

One obvious approach to penetrating the password mechanism is to attempt to find a general method of inverting the encryption algorithm. Very possibly this can be done, but few successful results have come to light, despite substantial efforts extending over a period of more than five years. The results have not proved to be very useful in penetrating systems.

Another approach to penetration is simply to keep trying potential passwords until one succeeds; this is a general cryptanalytic approach called *key search*. Human beings being what they are, there is a strong tendency for people to choose relatively short and simple passwords that they can remember. Given free choice, most people will choose their passwords from a restricted character set (e.g., all lower-case letters), and will often choose words or names. This human habit makes the key search job a great deal easier.

The critical factor involved in key search is the amount of time needed to encrypt a potential password and to check the result against an entry in the password file. The running time to encrypt one trial password and check the result turned out to be approximately 1.25 milliseconds on a PDP-11/70 when the encryption algorithm was recoded for maximum speed. It takes essentially no more time to test the encrypted trial password against all the passwords in an entire password file, or for that matter, against any collection of encrypted passwords, perhaps collected from many installations.

If we want to check all passwords of length n that consist entirely of lower-case letters, the number of such passwords is 26^n. If we suppose that the password consists of printable characters only, then the number of possible passwords is somewhat less than 95^n. (The standard system "character erase" and "line kill" characters are, for example, not prime candidates.) We can immediately estimate the running time of a program that will test every password of a given length with all of its characters chosen from some set of characters. The following table gives estimates of the running time required on a PDP-11/70 to test all possible character strings of length n chosen from various sets of characters: namely, all lower-case letters, all lower-case letters plus digits, all alphanumeric characters, all 95 printable ASCII characters, and finally all 128 ASCII characters.

Plaid Ex1003
Page 270 of 372

n	26 lower-case letters	36 lower-case letters and digits	62 alpha-numeric characters	95 printable characters	all 128 ASCII characters
1	30 msec.	40 msec.	80 msec.	120 msec.	160 msec.
2	800 msec.	2 sec.	5 sec.	11 sec.	20 sec.
3	22 sec.	58 sec.	5 min.	17 min.	44 min.
4	10 min.	35 min.	5 hrs.	28 hrs.	93 hrs.
5	4 hrs.	21 hrs.	318 hrs.	112 days	500 days
6	107 hrs.	760 hrs.	2.2 yrs.	29 yrs.	174 yrs.

One has to conclude that it is no great matter for someone with access to a PDP-11 to test all lower-case alphabetic strings up to length five and, given access to the machine for, say, several weekends, to test all such strings up to six characters in length. By using such a program against a collection of actual encrypted passwords, a substantial fraction of all the passwords will be found.

Another profitable approach for the bad guy is to use the word list from a dictionary or to use a list of names. For example, a large commercial dictionary contains typically about 250,000 words; these words can be checked in about five minutes. Again, a noticeable fraction of any collection of passwords will be found. Improvements and extensions will be (and have been) found by a determined bad guy. Some "good" things to try are:

—The dictionary with the words spelled backwards.
—A list of first names (best obtained from some mailing list). Last names, street names, and city names also work well.
—The above with initial upper-case letters.
—All valid license plate numbers in your state. (This takes about five hours in New Jersey.)
—Room numbers, social security numbers, telephone numbers, and the like.

The authors have conducted experiments to try to determine typical users' habits in the choice of passwords when no constraint is put on their choice. The results were disappointing, except to the bad guy. In a collection of 3,289 passwords gathered from many users over a long period of time,

15 were a single ASCII character;
72 were strings of two ASCII characters;
464 were strings of three ASCII characters;
477 were strings of four alphamerics;
706 were five letters, all upper-case or all lower-case;
605 were six letters, all lower-case.

An additional 492 passwords appeared in various available dictionaries, name lists, and the like. A total of 2,831 or 86 percent of this sample of passwords fell into one of these classes.

There was, of course, considerable overlap between the dictionary results and the character string searches. The dictionary search alone, which required only five minutes to run, produced about one third of the passwords.

Users could be urged (or forced) to use either longer passwords or passwords chosen from a larger character set, or the system could itself choose passwords for the users.

An Anecdote

An entertaining and instructive example is the attempt made at one installation to force users to use less predictable passwords. The users did not choose their own passwords; the system supplied them. The supplied passwords were eight characters long and were taken from the character set consisting of lower-case letters and digits. They were generated by a pseudorandom number generator with only 2^{15} starting values. The time required to search (again on a PDP-11/70) through all character strings of length 8 from a 36-character alphabet is 112 years.

Unfortunately, only 2^{15} of them need be looked at, because that is the number of possible outputs of the random number generator. The bad guy did, in fact, generate and test each of these strings and found every one of the system-generated passwords using a total of only about one minute of machine time. In this case, no harm was done, as the bad guy happened to be a friendly user.

Improvements to the First Approach

1. Slower Encryption

Obviously, the first algorithm used was far too fast. The announcement of the DES encryption algorithm [2] by the National Bureau of Standards was timely and fortunate. The DES is, by design, hard to invert, but equally valuable is the fact that it is extremely slow when implemented in software. The DES was implemented and used in the following way: The first eight characters of the user's password are used as a key for the DES; then the algorithm is used to encrypt a constant. Although this constant is zero at the moment, it is easily accessible and can be made installation-dependent. Then the DES algorithm is iterated 25 times and the resulting 64 bits are repacked to become a string of 11 printable characters.

2. Less Predictable Passwords

The password entry program was modified so as to urge the user to use more obscure passwords. If the user enters an alphabetic password (all upper-case or all lower-case) shorter than six characters, or a password from a larger character set shorter than five characters, then the program asks him to enter a longer password. This further reduces the efficacy of key search.

These improvements make it exceedingly difficult to find any individual password. The user is warned of the risks and if he cooperates, he is very safe indeed. On the other hand, he is not prevented from using his spouse's name if he wants to.

Plaid Ex1003
Page 271 of 372

3. Salted Passwords

The key search technique is still likely to turn up a few passwords when it is used on a large collection of passwords, and it seemed wise to make this task as difficult as possible. To this end, when a password is first entered, the password program obtains a 12-bit random number (by reading the real-time clock) and appends this to the password typed in by the user. The concatenated string is encrypted and both the 12-bit random quantity (called the *salt*) and the 64-bit result of the encryption are entered into the password file.

When the user later logs in to the system, the 12-bit quantity is extracted from the password file and appended to the typed password. The encrypted result is required, as before, to be the same as the remaining 64 bits in the password file. This modification does not increase the task of finding any individual password, starting from scratch, but now the work of testing a given character string against a large collection of encrypted passwords has been multiplied by 4,096 (2^{12}). The reason for this is that there are 4,096 encrypted versions of each password and one of them has been picked more or less at random by the system.

With this modification, it is likely that the bad guy can spend days of computer time trying to find a password on a system with hundreds of passwords, and find none at all. More important is the fact that it becomes impractical to prepare an encrypted dictionary in advance. Such an encrypted dictionary could be used to crack new passwords in milliseconds when they appear.

There is a (not inadvertent) side effect of this modification. It becomes nearly impossible to find out whether a person with passwords on two or more systems has used the same password on all of them, unless you already know that.

4. The Threat of the DES Chip

Chips to perform the DES encryption are already commercially available and they are very fast. The use of such a chip speeds up the process of password hunting by three orders of magnitude. To avert this possibility, one of the internal tables of the DES algorithm (in particular, the so-called E-table) is changed in a way that depends on the 12-bit random number. The E-table is inseparably wired into the DES chip, so that the commercial chip cannot be used. Obviously, the bad guy could have his own chip designed and built, but the cost would be very high.

5. A Subtle Point

To log in successfully on the UNIX system, it is necessary after dialing in to type a valid user name, and then the correct password for that user name. It is poor design to write the login command in such a way that it tells an interloper when he has typed in an invalid user name. The response to an invalid name should be identical to that for a valid name.

When the slow encryption algorithm was first implemented, the encryption was done only if the user name was valid, because otherwise there was no encrypted password to compare with the supplied password. The result was that the response was delayed by about one-half second if the name was valid, but was immediate if invalid. The bad guy could find out whether a particular user name was valid. The routine was modified to do the encryption in either case.

Conclusions

On the issue of password security, UNIX is probably better than most systems. The use of encrypted passwords appears reasonably secure in the absence of serious attention of experts in the field.

It is also worth some effort to conceal even the encrypted passwords. Some UNIX systems have instituted what is called an "external security code" that must be typed when dialing into the system, but before logging in. If this code is changed periodically, then someone with an old password will likely be prevented from using it.

Whenever any security procedure is set up to deny access to unauthorized persons, it is wise to keep a record of both successful and unsuccessful attempts to get at the secured resource. For example, an out-of-hours visitor to a computer center normally must not only identify himself, but a record is usually also kept of his entry. Just so, it is a wise precaution to make and keep a record of all attempts to log into a remote-access time-sharing system, and certainly all unsuccessful attempts.

Bad guys fall on a spectrum whose one end is someone with ordinary access to a system and whose goal is to find out a particular password (usually that of the super-user) and, at the other end, someone who wishes to collect as much password information as possible from as many systems as possible. Most of the work reported here serves to frustrate the latter type; our experience indicates that the former type of bad guy never was very successful.

We recognize that a time-sharing system must operate in a hostile environment. We did not attempt to hide the security aspects of the operating system, thereby playing the customary make-believe game in which weaknesses of the system are not discussed no matter how apparent. Rather we advertised the password algorithm and invited attack in the belief that this approach would minimize future trouble. The approach has been successful.

Received August 1978; revised August 1979

References

1. Hagelin, B. Ciphering Machine (M–209), U.S. Patent No. 2,089,603, Aug. 10, 1937.
2. Proposed federal information processing data encryption standard. Federal Register (40FR12134), March 17, 1975.
3. Ritchie, D.M., and Thompson, K. The UNIX time-sharing system. *Comm. ACM 17*, 7 (July 1974), 365–375.
4. Wilkes, M.V. *Time-Sharing Computer Systems.* American Elsevier, New York, 1968.

Plaid Ex1003
Page 272 of 372

EXHIBIT E

To access the contents, click the chapter and section titles.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)

(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

Bookmark it

Search this book:

Go!

Foreword by Whitfield Diffie

Preface

About the Author

Chapter 1—Foundations

1.1 Terminology
1.2 Steganography
1.3 Substitution Ciphers and Transposition Ciphers
1.4 Simple XOR
1.5 One-Time Pads
1.6 Computer Algorithms
1.7 Large Numbers

Part I—Cryptographic Protocols

Chapter 2—Protocol Building Blocks

2.1 Introduction to Protocols
2.2 Communications Using Symmetric Cryptography
2.3 One-Way Functions
2.4 One-Way Hash Functions
2.5 Communications Using Public-Key Cryptography
2.6 Digital Signatures

2.7 Digital Signatures with Encryption
2.8 Random and Pseudo-Random-Sequence Generation

Chapter 3—Basic Protocols

3.1 Key Exchange
3.2 Authentication
3.3 Authentication and Key Exchange
3.4 Formal Analysis of Authentication and Key-Exchange Protocols
3.5 Multiple-Key Public-Key Cryptography
3.6 Secret Splitting
3.7 Secret Sharing
3.8 Cryptographic Protection of Databases

Chapter 4—Intermediate Protocols

4.1 Timestamping Services
4.2 Subliminal Channel
4.3 Undeniable Digital Signatures
4.4 Designated Confirmer Signatures
4.5 Proxy Signatures
4.6 Group Signatures
4.7 Fail-Stop Digital Signatures
4.8 Computing with Encrypted Data
4.9 Bit Commitment
4.10 Fair Coin Flips
4.11 Mental Poker
4.12 One-Way Accumulators
4.13 All-or-Nothing Disclosure of Secrets
4.14 Key Escrow

Chapter 5—Advanced Protocols

5.1 Zero-Knowledge Proofs
5.2 Zero-Knowledge Proofs of Identity
5.3 Blind Signatures
5.4 Identity-Based Public-Key Cryptography
5.5 Oblivious Transfer
5.6 Oblivious Signatures
5.7 Simultaneous Contract Signing

5.8 Digital Certified Mail
5.9 Simultaneous Exchange of Secrets

Chapter 6—Esoteric Protocols
6.1 Secure Elections
6.2 Secure Multiparty Computation
6.3 Anonymous Message Broadcast
6.4 Digital Cash

Part II—Cryptographic Techniques

Chapter 7—Key Length
7.1 Symmetric Key Length
7.2 Public-Key Key Length
7.3 Comparing Symmetric and Public-Key Key Length
7.4 Birthday Attacks against One-Way Hash Functions
7.5 How Long Should a Key Be?
7.6 Caveat Emptor

Chapter 8—Key Management
8.1 Generating Keys
8.2 Nonlinear Keyspaces
8.3 Transferring Keys
8.4 Verifying Keys
8.5 Using Keys
8.6 Updating Keys
8.7 Storing Keys
8.8 Backup Keys
8.9 Compromised Keys
8.10 Lifetime of Keys
8.11 Destroying Keys
8.12 Public-Key Key Management

Chapter 9—Algorithm Types and Modes
9.1 Electronic Codebook Mode
9.2 Block Replay
9.3 Cipher Block Chaining Mode
9.4 Stream Ciphers

9.5 Self-Synchronizing Stream Ciphers
9.6 Cipher-Feedback Mode
9.7 Synchronous Stream Ciphers
9.8 Output-Feedback Mode
9.9 Counter Mode
9.10 Other Block-Cipher Modes
9.11 Choosing a Cipher Mode
9.12 Interleaving
9.13 Block Ciphers versus Stream Ciphers

Chapter 10—Using Algorithms

10.1 Choosing an Algorithm
10.2 Public-Key Cryptography versus Symmetric Cryptography
10.3 Encrypting Communications Channels
10.4 Encrypting Data for Storage
10.5 Hardware Encryption versus Software Encryption
10.6 Compression, Encoding, and Encryption
10.7 Detecting Encryption
10.8 Hiding Ciphertext in Ciphertext
10.9 Destroying Information

Part III—Cryptographic Algorithms

Chapter 11—Mathematical Background

11.1 Information Theory
11.2 Complexity Theory
11.3 Number Theory
11.4 Factoring
11.5 Prime Number Generation
11.6 Discrete Logarithms in a Finite Field

Chapter 12—Data Encryption Standard (DES)

12.1 Background
12.2 Description of DES
12.3 Security of DES
12.4 Differential and Linear Cryptanalysis
12.5 The Real Design Criteria

12.6 DES Variants
12.7 How Secure Is DES Today?

Chapter 13—Other Block Ciphers
13.1 Lucifer
13.2 Madryga
13.3 NewDES
13.4 FEAL
13.5 REDOC
13.6 LOKI
13.7 Khufu and Khafre
13.8 RC2
13.9 IDEA
13.10 MMB
13.11 CA-1.1
13.12 Skipjack

Chapter 14—Still Other Block Ciphers
14.1 GOST
14.2 CAST
14.3 Blowfish
14.4 SAFER
14.5 3-Way
14.6 Crab
14.7 SXAL8/MBAL
14.8 RC5
14.9 Other Block Algorithms
14.10 Theory of Block Cipher Design
14.11 Using one-Way Hash Functions
14.12 Choosing a Block Algorithm

Chapter 15—Combining Block Ciphers
15.1 Double Encryption
15.2 Triple Encryption
15.3 Doubling the Block Length
15.4 Other Multiple Encryption Schemes
15.5 CDMF Key Shortening
15.6 Whitening

15.7 Cascading Multiple Block Algorithms
15.8 Combining Multiple Block Algorithms

Chapter 16—Pseudo-Random-Sequence Generators and Stream Ciphers
16.1 Linear Congruential Generators
16.2 Linear Feedback Shift Registers
16.3 Design and Analysis of Stream Ciphers
16.4 Stream Ciphers Using LFSRs
16.5 A5
16.6 Hughes XPD/KPD
16.7 Nanoteq
16.8 Rambutan
16.9 Additive Generators
16.10 Gifford
16.11 Algorithm M
16.12 PKZIP

Chapter 17—Other Stream Ciphers and Real Random-Sequence Generators
17.1 RC4
17.2 SEAL
17.3 WAKE
17.4 Feedback with Carry Shift Registers
17.5 Stream Ciphers Using FCSRs
17.6 Nonlinear-Feedback Shift Registers
17.7 Other Stream Ciphers
17.8 System-Theoretic Approach to Stream-Cipher Design
17.9 Complexity-Theoretic Approach to Stream-Cipher Design
17.10 Other Approaches to Stream-Cipher Design
17.11 Cascading Multiple Stream Ciphers
17.12 Choosing a Stream Cipher
17.13 Generating Multiple Streams from a Single Pseudo-Random-Sequence Generator
17.14 Real Random-Sequence Generators

Chapter 18—One-Way Hash Functions

18.1 Background

18.2 Snefru

18.3 N- Hash

18.4 MD4

18.5 MD5

18.6 MD2

18.7 Secure Hash Algorithm (SHA)

18.8 RIPE-MD

18.9 HAVAL

18.10 Other One-Way Hash Functions

18.11 One-Way Hash Functions Using Symmetric Block Algorithms

18.12 Using Public-Key Algorithms

18.13 Choosing a One-Way Hash Function

18.14 Message Authentication Codes

Chapter 19—Public-Key Algorithms

19.1 Background

19.2 Knapsack Algorithms

19.3 RSA

19.4 Pohlig-Hellman

19.5 Rabin

19.6 ElGamal

19.7 McEliece

19.8 Elliptic Curve Cryptosystems

19.9 LUC

19.10 Finite Automaton Public-Key Cryptosystems

Chapter 20—Public-Key Digital Signature Algorithms

20.1 Digital Signature Algorithm (DSA)

20.2 DSA Variants

20.3 Gost Digital Signature Algorithm

20.4 Discrete Logarithm Signature Schemes

20.5 Ong-Schnorr-Shamir

20.6 ESIGN

20.7 Cellular Automata
20.8 Other Public-Key Algorithms

Chapter 21—Identification Schemes

21.1 Feige-Fiat-Shamir
21.2 Guillou-Quisquater
21.3 Schnorr
21.4 Converting Identification Schemes to Signature Schemes

Chapter 22—Key-Exchange Algorithms

22.1 Diffie-Hellman
22.2 Station-to-Station Protocol
22.3 Shamir's Three-Pass Protocol
22.4 COMSET
22.5 Encrypted Key Exchange
22.6 Fortified Key Negotiation
22.7 Conference Key Distribution and Secret Broadcasting

Chapter 23—Special Algorithms for Protocols

23.1 Multiple-Key Public-Key Cryptography
23.2 Secret-Sharing Algorithms
23.3 Subliminal Channel
23.4 Undeniable Digital Signatures
23.5 Designated Confirmer Signatures
23.6 Computing with Encrypted Data
23.7 Fair Coin Flips
23.8 One-Way Accumulators
23.9 All-or-Nothing Disclosure of Secrets
23.10 Fair and Failsafe Cryptosystems
23.11 Zero-Knowledge Proofs of Knowledge
23.12 Blind Signatures
23.13 Oblivious Transfer
23.14 Secure Multiparty Computation
23.15 Probabilistic Encryption
23.16 Quantum Cryptography

Part IV—The Real World

Chapter 24—Example Implementations

24.1 IBM Secret-Key Management Protocol

24.2 MITRENET

24.3 ISDN

24.4 STU-III

24.5 Kerberos

24.6 KryptoKnight

24.7 SESAME

24.8 IBM Common Cryptographic Architecture

24.9 ISO Authentication Framework

24.10 Privacy-Enhanced Mail (PEM)

24.11 Message Security Protocol (MSP)

24.12 Pretty Good Privacy (PGP)

24.13 Smart Cards

24.14 Public-Key Cryptography Standards (PKCS)

24.15 Universal Electronic Payment System (UEPS)

24.16 Clipper

24.17 Capstone

24.18 AT&ampT Model 3600 Telephone Security Device (TSD)

Chapter 25—Politics

25.1 National Security Agency (NSA)

25.2 National Computer Security Center (NCSC)

25.3 National Institute of Standards and Technology (NIST)

25.4 RSA Data Security, Inc.

25.5 Public Key Partners

25.6 International Association for Cryptologic Research (IACR)

25.7 RACE Integrity Primitives Evaluation (RIPE)

25.8 Conditional Access for Europe (CAFE)

25.9 ISO/IEC 9979

25.10 Professional, Civil Liberties, and Industry Groups

25.11 Sci.crypt

25.12 Cypherpunks

25.13 Patents
25.14 U.S. Export Rules
25.15 Foreign Import and Export of Cryptography
25.16 Legal Issues

Afterword by Matt Blaze

Part V—Source Code

References

Index

Products | Contact Us | About Us | Privacy | Ad Info | Home

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

Foreword By Whitfield Diffie

The literature of cryptography has a curious history. Secrecy, of course, has always played a central role, but until the First World War, important developments appeared in print in a more or less timely fashion and the field moved forward in much the same way as other specialized disciplines. As late as 1918, one of the most influential cryptanalytic papers of the twentieth century, William F. Friedman's monograph *The Index of Coincidence and Its Applications in Cryptography*, appeared as a research report of the private Riverbank Laboratories [577]. And this, despite the fact that the work had been done as part of the war effort. In the same year Edward H. Hebern of Oakland, California filed the first patent for a rotor machine [710], the device destined to be a mainstay of military cryptography for nearly 50 years.

After the First World War, however, things began to change. U.S. Army and Navy organizations, working entirely in secret, began to make fundamental advances in cryptography. During the thirties and forties a few basic papers did appear in the open literature and several treatises on the subject were published, but the latter were farther and farther behind the state of the art. By the end of the war the transition was complete. With one notable exception, the public literature had died. That exception was Claude Shannon's paper "The Communication Theory of Secrecy Systems," which appeared in the *Bell System Technical Journal* in 1949 [1432]. It was similar to Friedman's 1918 paper, in that it grew out of wartime work of Shannon's. After the Second World War ended it was declassified, possibly by mistake.

From 1949 until 1967 the cryptographic literature was barren. In that year a different sort of contribution appeared: David Kahn's history, *The Codebreakers* [794]. It didn't contain any new technical ideas, but it did contain a remarkably complete history of what had gone before, including

mention of some things that the government still considered secret. The significance of *The Codebreakers* lay not just in its remarkable scope, but also in the fact that it enjoyed good sales and made tens of thousands of people, who had never given the matter a moment's thought, aware of cryptography. A trickle of new cryptographic papers began to be written.

At about the same time, Horst Feistel, who had earlier worked on identification friend or foe devices for the Air Force, took his lifelong passion for cryptography to the IBM Watson Laboratory in Yorktown Heights, New York. There, he began development of what was to become the U.S. Data Encryption Standard; by the early 1970s several technical reports on this subject by Feistel and his colleagues had been made public by IBM [1482,1484,552].

This was the situation when I entered the field in late 1972. The cryptographic literature wasn't abundant, but what there was included some very shiny nuggets.

Cryptology presents a difficulty not found in normal academic disciplines: the need for the proper interaction of cryptography and cryptanalysis. This arises out of the fact that in the absence of real communications requirements, it is easy to propose a system that appears unbreakable. Many academic designs are so complex that the would–be cryptanalyst doesn't know where to start; exposing flaws in these designs is far harder than designing them in the first place. The result is that the competitive process, which is one strong motivation in academic research, cannot take hold.

When Martin Hellman and I proposed public–key cryptography in 1975 [496], one of the indirect aspects of our contribution was to introduce a problem that does not even appear easy to solve. Now an aspiring cryptosystem designer could produce something that would be recognized as clever—something that did more than just turn meaningful text into nonsense. The result has been a spectacular increase in the number of people working in cryptography, the number of meetings held, and the number of books and papers published.

In my acceptance speech for the Donald E. Fink award—given for the best expository paper to appear in an IEEE journal—which I received jointly with Hellman in 1980, I told the audience that in writing "Privacy and Authentication," I had an experience that I suspected was rare even among the prominent scholars who populate the IEEE awards ceremony: I had written the paper I had wanted to study, but could not find, when I first became seriously interested in cryptography. Had I been able to go to the Stanford bookstore and pick up a modern cryptography text, I would probably have learned about the field years earlier. But the only things available in the fall of 1972 were a few classic papers and some obscure technical reports.

The contemporary researcher has no such problem. The problem now is choosing where to start among the thousands of papers and dozens of books. The contemporary researcher, yes, but what about the contemporary programmer or engineer who merely wants to use cryptography? Where does that person turn? Until now, it has been necessary to spend long hours hunting out and then studying the research literature before being able to design the sort of cryptographic utilities glibly described in popular articles.

This is the gap that Bruce Schneier's *Applied Cryptography* has come to fill. Beginning with the objectives of communication security and elementary examples of programs used to achieve these objectives, Schneier gives us a panoramic view of the fruits of 20 years of public research. The title says it all; from the mundane objective of having a secure conversation the very first time you call someone to the possibilities of digital money and cryptographically secure elections, this is where you'll find it.

Not satisfied that the book was about the real world merely because it went all the way down to the code, Schneier has included an account of the world in which cryptography is developed and applied, and discusses entities ranging from the International Association for Cryptologic Research to the NSA.

When public interest in cryptography was just emerging in the late seventies and early eighties, the National Security Agency (NSA), America's official cryptographic organ, made several attempts to quash it. The first was a letter from a long–time NSA employee allegedly, avowedly, and apparently acting on his own. The letter was sent to the IEEE and warned that the publication of cryptographic material was a violation of the International Traffic in Arms Regulations (ITAR). This viewpoint turned out not even to be supported by the regulations themselves—which contained an explicit exemption for published material—but gave both the public practice of cryptography and the 1977 Information Theory Workshop lots of unexpected publicity.

A more serious attempt occurred in 1980, when the NSA funded the American Council on Education to examine the issue with a view to persuading Congress to give it legal control of publications in the field of cryptography. The results fell far short of NSA's ambitions and resulted in a program of voluntary review of cryptographic papers; researchers were requested to ask the NSA's opinion on whether disclosure of results would adversely affect the national interest before publication.

As the eighties progressed, pressure focused more on the practice than the study of cryptography. Existing laws gave the NSA the power, through the Department of State, to regulate the export of cryptographic equipment. As business became more and more international and the American fraction of the world market declined, the pressure to have a single product in both domestic and offshore markets increased. Such single products were subject to export control and thus the NSA acquired substantial influence not only over what was exported, but also over what was sold in the United States.

As this is written, a new challenge confronts the public practice of cryptography. The government has augmented the widely published and available Data Encryption Standard, with a secret algorithm implemented in tamper–resistant chips. These chips will incorporate a codified mechanism of government monitoring. The negative aspects of this "key–escrow" program range from a potentially disastrous impact on personal privacy to the high cost of having to add hardware to products that had previously encrypted in software. So far key escrow products are enjoying less than stellar sales and the scheme has attracted widespread negative comment, especially from the independent cryptographers. Some people, however, see more future in programming than politicking and have redoubled their efforts to provide the

world with strong cryptography that is accessible to public scrutiny.

A sharp step back from the notion that export control law could supersede the First Amendment seemed to have been taken in 1980 when the *Federal Register* announcement of a revision to ITAR included the statement: "...provision has been added to make it clear that the regulation of the export of technical data does not purport to interfere with the First Amendment rights of individuals." But the fact that tension between the First Amendment and the export control laws has not gone away should be evident from statements at a conference held by RSA Data Security. NSA's representative from the export control office expressed the opinion that people who published cryptographic programs were "in a grey area" with respect to the law. If that is so, it is a grey area on which the first edition of this book has shed some light. Export applications for the book itself have been granted, with acknowledgement that published material lay beyond the authority of the Munitions Control Board. Applications to export the enclosed programs on disk, however, have been denied.

The shift in the NSA's strategy, from attempting to control cryptographic research to tightening its grip on the development and deployment of cryptographic products, is presumably due to its realization that all the great cryptographic papers in the world do not protect a single bit of traffic. Sitting on the shelf, this volume may be able to do no better than the books and papers that preceded it, but sitting next to a workstation, where a programmer is writing cryptographic code, it just may.

Whitfield Diffie
Mountain View, CA

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

Preface

There are two kinds of cryptography in this world: cryptography that will stop your kid sister from reading your files, and cryptography that will stop major governments from reading your files. This book is about the latter.

If I take a letter, lock it in a safe, hide the safe somewhere in New York, then tell you to read the letter, that's not security. That's obscurity. On the other hand, if I take a letter and lock it in a safe, and then give you the safe along with the design specifications of the safe and a hundred identical safes with their combinations so that you and the world's best safecrackers can study the locking mechanism—and you still can't open the safe and read the letter—that's security.

For many years, this sort of cryptography was the exclusive domain of the military. The United States' National Security Agency (NSA), and its counterparts in the former Soviet Union, England, France, Israel, and elsewhere, have spent billions of dollars in the very serious game of securing their own communications while trying to break everyone else's. Private individuals, with far less expertise and budget, have been powerless to protect their own privacy against these governments.

During the last 20 years, public academic research in cryptography has exploded. While classical cryptography has been long used by ordinary citizens, computer cryptography was the exclusive domain of the world's militaries since World War II. Today, state–of–the–art computer cryptography is practiced outside the secured walls of the military agencies. The layperson can now employ security practices that can protect against the most powerful of adversaries—security that may protect against military agencies for years to come.

Do average people really need this kind of security? Yes. They may be planning a political campaign, discussing taxes, or having an illicit affair. They may be designing a new product, discussing a marketing strategy, or planning a hostile business takeover. Or they may be living in a country that does not respect the rights of privacy of its citizens. They may be doing something that they feel shouldn't be illegal, but is. For whatever reason, the data and communications are personal, private, and no one else's business.

This book is being published in a tumultuous time. In 1994, the Clinton administration approved the Escrowed Encryption Standard (including the Clipper chip and Fortezza card) and signed the Digital Telephony bill into law. Both of these initiatives try to ensure the government's ability to conduct electronic surveillance.

Some dangerously Orwellian assumptions are at work here: that the government has the right to listen to private communications, and that there is something wrong with a private citizen trying to keep a secret from the government. Law enforcement has always been able to conduct court–authorized surveillance if possible, but this is the first time that the people have been forced to take active measures to *make themselves available* for surveillance. These initiatives are not simply government proposals in some obscure area; they are preemptive and unilateral attempts to usurp powers that previously belonged to the people.

Clipper and Digital Telephony do not protect privacy; they force individuals to unconditionally trust that the government will respect their privacy. The same law enforcement authorities who illegally tapped Martin Luther King Jr.'s phones can easily tap a phone protected with Clipper. In the recent past, local police authorities have either been charged criminally or sued civilly in numerous jurisdictions—Maryland, Connecticut, Vermont, Georgia, Missouri, and Nevada—for conducting illegal wiretaps. It's a poor idea to deploy a technology that could some day facilitate a police state.

The lesson here is that it is insufficient to protect ourselves with laws; we need to protect ourselves with mathematics. Encryption is too important to be left solely to governments.

This book gives you the tools you need to protect your own privacy; cryptography products may be declared illegal, but the information will never be.

How to Read This Book

I wrote *Applied Cryptography* to be both a lively introduction to the field of cryptography and a comprehensive reference. I have tried to keep the text readable without sacrificing accuracy. This book is not intended to be a mathematical text. Although I have not deliberately given any false information, I do play fast and loose with theory. For those interested in formalism, there are copious references to the academic literature.

Chapter 1 introduces cryptography, defines many terms, and briefly discusses precomputer cryptography.

Chapters 2 through 6 (Part I) describe cryptographic protocols: what people can do with cryptography. The protocols range from the simple (sending encrypted messages from one person to another) to the complex (flipping a coin over the telephone) to the esoteric (secure and anonymous digital money exchange). Some of these protocols are obvious; others are almost amazing. Cryptography can solve a lot of problems that most people never realized it could.

Chapters 7 through 10 (Part II) discuss cryptographic techniques. All four chapters in this section are important for even the most basic uses of cryptography. Chapters 7 and 8 are about keys: how long a key should be in order to be secure, how to generate keys, how to store keys, how to dispose of keys, and so on. Key management is the hardest part of cryptography and often the Achilles' heel of an otherwise secure system. Chapter 9 discusses different ways of using cryptographic algorithms, and Chapter 10 gives the odds and ends of algorithms: how to choose, implement, and use algorithms.

Chapters 11 through 23 (Part III) list algorithms. Chapter 11 provides the mathematical background. This chapter is only required if you are interested in public–key algorithms. If you just want to implement DES (or something similar), you can skip ahead. Chapter 12 discusses DES: the algorithm, its history, its security, and some variants. Chapters 13, 14, and 15 discuss other block algorithms; if you want something more secure than DES, skip to the section on IDEA and triple–DES. If you want to read about a bunch of algorithms, some of which may be more secure than DES, read the whole chapter. Chapters 16 and 17 discuss stream algorithms. Chapter 18 focuses on one–way hash functions; MD5 and SHA are the most common, although I discuss many more. Chapter 19 discusses public–key encryption algorithms, Chapter 20 discusses public–key digital signature algorithms, Chapter 21 discusses public–key identification algorithms, and Chapter 22 discusses public–key key exchange algorithms. The important algorithms are RSA, DSA, Fiat–Shamir, and Diffie–Hellman, respectively. Chapter 23 has more esoteric public–key algorithms and protocols; the math in this chapter is quite complicated, so wear your seat belt.

Chapters 24 and 25 (Part IV) turn to the real world of cryptography. Chapter 24 discusses some of the current implementations of these algorithms and protocols, while Chapter 25 touches on some of the political issues surrounding cryptography. These chapters are by no means intended to be comprehensive.

Also included are source code listings for 10 algorithms discussed in Part III. I was unable to include all the code I wanted to due to space limitations, and cryptographic source code cannot otherwise be exported. (Amazingly enough, the State Department allowed export of the first edition of this book with source code, but denied export for a computer disk with the exact same source code on it. Go figure.) An associated source code disk set includes much more source code than I could fit in this book; it is probably the largest collection of cryptographic source code outside a military institution. I can only send source code disks to U.S. and Canadian citizens living in the U.S. and Canada, but hopefully that will change someday. If you are interested in implementing or playing with the cryptographic algorithms in this book, get the disk. See the

last page of the book for details.

One criticism of this book is that its encyclopedic nature takes away from its readability. This is true, but I wanted to provide a single reference for those who might come across an algorithm in the academic literature or in a product. For those who are more interested in a tutorial, I apologize. A lot is being done in the field; this is the first time so much of it has been gathered between two covers. Even so, space considerations forced me to leave many things out. I covered topics that I felt were important, practical, or interesting. If I couldn't cover a topic in depth, I gave references to articles and papers that did.

I have done my best to hunt down and eradicate all errors in this book, but many have assured me that it is an impossible task. Certainly, the second edition has far fewer errors than the first. An errata listing is available from me and will be periodically posted to the Usenet newsgroup sci.crypt. If any reader finds an error, please let me know. I'll send the first person to find each error in the book a free copy of the source code disk.

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

About the Author

BRUCE SCHNEIER is president of Counterpane Systems, an Oak Park, Illinois consulting firm specializing in cryptography and computer security. Bruce is also the author of *E–Mail Security* (John Wiley & Sons, 1995) and *Protect Your Macintosh* (Peachpit Press, 1994); and has written dozens of articles on cryptography for major magazines. He is a contributing editor to *Dr. Dobb's Journal*, where he edits the "Algorithms Alley" column, and a contributing editor to *Computer and Communications Security Reviews*. Bruce serves on the board of directors of the International Association for Cryptologic Research, is a member of the Advisory Board for the Electronic Privacy Information Center, and is on the program committee for the New Security Paradigms Workshop. In addition, he finds time to give frequent lectures on cryptography, computer security, and privacy.

Acknowledgments

The list of people who had a hand in this book may seem unending, but all are worthy of mention. I would like to thank Don Alvarez, Ross Anderson, Dave Balenson, Karl Barrus, Steve Bellovin, Dan Bernstein, Eli Biham, Joan Boyar, Karen Cooper, Whit Diffie, Joan Feigenbaum, Phil Karn, Neal Koblitz, Xuejia Lai, Tom Leranth, Mike Markowitz, Ralph Merkle, Bill Patton, Peter Pearson, Charles Pfleeger, Ken Pizzini, Bart Preneel, Mark Riordan, Joachim Schurman, and Marc Schwartz for reading and editing all or parts of the first edition; Marc Vauclair for translating the first edition into French; Abe Abraham, Ross Anderson, Dave Banisar, Steve Bellovin, Eli Biham, Matt Bishop, Matt Blaze, Gary Carter, Jan Camenisch, Claude CrŽpeau, Joan Daemen, Jorge Davila, Ed Dawson, Whit Diffie, Carl Ellison, Joan Feigenbaum, Niels Ferguson, Matt Franklin, Rosario Gennaro, Dieter

Gollmann, Mark Goresky, Richard Graveman, Stuart Haber, Jingman He, Bob Hogue, Kenneth Iversen, Markus Jakobsson, Burt Kaliski, Phil Karn, John Kelsey, John Kennedy, Lars Knudsen, Paul Kocher, John Ladwig, Xuejia Lai, Arjen Lenstra, Paul Leyland, Mike Markowitz, Jim Massey, Bruce McNair, William Hugh Murray, Roger Needham, Clif Neuman, Kaisa Nyberg, Luke O'Connor, Peter Pearson, RenŽ Peralta, Bart Preneel, Yisrael Radai, Matt Robshaw, Michael Roe, Phil Rogaway, Avi Rubin, Paul Rubin, Selwyn Russell, Kazue Sako, Mahmoud Salmasizadeh, Markus Stadler, Dmitry Titov, Jimmy Upton, Marc Vauclair, Serge Vaudenay, Gideon Yuval, Glen Zorn, and several anonymous government employees for reading and editing all or parts of the second edition; Lawrie Brown, Leisa Condie, Joan Daemen, Peter Gutmann, Alan Insley, Chris Johnston, John Kelsey, Xuejia Lai, Bill Leininger, Mike Markowitz, Richard Outerbridge, Peter Pearson, Ken Pizzini, Colin Plumb, RSA Data Security, Inc., Michael Roe, Michael Wood, and Phil Zimmermann for providing source code; Paul MacNerland for creating the figures for the first edition; Karen Cooper for copyediting the second edition; Beth Friedman for proofreading the second edition; Carol Kennedy for indexing the second edition; the readers of sci.crypt and the Cypherpunks mailing list for commenting on ideas, answering questions, and finding errors in the first edition; Randy Seuss for providing Internet access; Jeff Duntemann and Jon Erickson for helping me get started; assorted random Insleys for the impetus, encouragement, support, conversations, friendship, and dinners; and AT&ampT Bell Labs for firing me and making this all possible. All these people helped to create a far better book than I could have created alone.

Bruce Schneier
Oak Park, Ill.
schneier@counterpane.com

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

To access the contents, click the chapter and section titles.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

Bookmark it

Search this book:

Go!

Chapter 12
Data Encryption Standard (DES)

12.1 Background

The Data Encryption Standard (DES), known as the Data Encryption Algorithm (DEA) by ANSI and the DEA-1 by the ISO, has been a worldwide standard for 20 years. Although it is showing signs of old age, it has held up remarkably well against years of cryptanalysis and is still secure against all but possibly the most powerful of adversaries.

Development of the Standard

In the early 1970s, nonmilitary cryptographic research was haphazard. Almost no research papers were published in the field. Most people knew that the military used special coding equipment to communicate, but few understood the science of cryptography. The National Security Agency (NSA) had considerable knowledge, but they did not even publicly admit their own existence.

Buyers didn't know what they were buying. Several small companies made and sold cryptographic equipment, primarily to overseas governments. The equipment was all different and couldn't interoperate. No one really knew if any of it was secure; there was no independent body to certify the security. As one government report said [441]:

> The intricacies of relating key variations and working principles to the real strength of the encryption/decryption equipment were, and are, virtually unknown to almost all buyers, and informed

decisions as to the right type of on-line, off-line, key generation, etc., which will meet buyers' security needs, have been most difficult to make.

In 1972, the National Bureau of Standards (NBS), now the National Institute of Standards and Technology (NIST), initiated a program to protect computer and communications data. As part of that program, they wanted to develop a single, standard cryptographic algorithm. A single algorithm could be tested and certified, and different cryptographic equipment using it could interoperate. It would also be cheaper to implement and readily available.

In the May 15, 1973 *Federal Register,* the NBS issued a public request for proposals for a standard cryptographic algorithm. They specified a series of design criteria:

— The algorithm must provide a high level of security.

— The algorithm must be completely specified and easy to understand.

— The security of the algorithm must reside in the key; the security should not depend on the secrecy of the algorithm.

— The algorithm must be available to all users.

— The algorithm must be adaptable for use in diverse applications.

— The algorithm must be economically implementable in electronic devices.

— The algorithm must be efficient to use.

— The algorithm must be able to be validated.

— The algorithm must be exportable.

Public response indicated that there was considerable interest in a cryptographic standard, but little public expertise in the field. None of the submissions came close to meeting the requirements.

The NBS issued a second request in the August 27, 1974 *Federal Register.* Eventually they received a promising candidate: an algorithm based on one developed by IBM during the early 1970s, called Lucifer (see Section 13.1). IBM had a team working on cryptography at both Kingston and Yorktown Heights, including Roy Adler, Don Coppersmith, Horst Feistel, Edna Grossman, Alan Konheim, Carl Meyer, Bill Notz, Lynn Smith, Walt Tuchman, and Bryant Tuckerman.

The algorithm, although complicated, was straightforward. It used only simple logical operations on small groups of bits and could be implemented fairly efficiently in hardware.

The NBS requested the NSA's help in evaluating the algorithm's security and determining its suitability as a federal standard. IBM had already filed for a patent [514], but was willing to make its intellectual property available to others for manufacture, implementation, and use. Eventually, the NBS worked out the terms of agreement with IBM and received a nonexclusive, royalty-free license to make, use, and sell equipment that implemented the algorithm.

Finally, in the March 17, 1975 *Federal Register,* the NBS published both the details of the algorithm and IBM's statement granting a nonexclusive,

royalty-free license for the algorithm, and requested comment [536]. Another notice, in the August 1, 1975 *Federal Register,* again requested comments from agencies and the general public.

And there were comments [721,497,1120]. Many were wary of the NSA's "invisible hand" in the development of the algorithm. They were afraid that the NSA had modified the algorithm to install a trapdoor. They complained that the NSA reduced the key size from the original 128-bits to 56-bits (see Section 13.1). They complained about the inner workings of the algorithm. Much of NSA's reasoning became clear in the early 1990s, but in the 1970s this seemed mysterious and worrisome.

In 1976, the NBS held two workshops to evaluate the proposed standard. The first workshop discussed the mathematics of the algorithm and the possibility of a trapdoor [1139]. The second workshop discussed the possibility of increasing the algorithm's key length [229]. The algorithm's designers, evaluators, implementors, vendors, users, and critics were invited. From all reports, the workshops were lively [1118].

Despite criticism, the Data Encryption Standard was adopted as a federal standard on November 23, 1976 [229] and authorized for use on all unclassified government communications. The official description of the standard, FIPS PUB 46, "Data Encryption Standard," was published on January 15, 1977 and became effective six months later [1140]. FIPS PUB 81, "DES Modes of Operation," was published in 1980 [1143]. FIPS PUB 74, "Guidelines for Implementing and Using the NBS Data Encryption Standard," was published in 1981 [1142]. NBS also published FIPS PUB 112, specifying DES for password encryption [1144], and FIPS PUB 113, specifying DES for computer data authentication [1145]. (FIPS stands for Federal Information Processing Standard.)

These standards were unprecedented. Never before had an NSA-evaluated algorithm been made public. This was probably the result of a misunderstanding between NSA and NBS. The NSA thought DES was hardware-only. The standard mandated a hardware implementation, but NBS published enough details so that people could write DES software. Off the record, NSA has characterized DES as one of their biggest mistakes. If they knew the details would be released so that people could write software, they would never have agreed to it. DES did more to galvanize the field of cryptanalysis than anything else. Now there was an algorithm to study: one that the NSA said was secure. It is no accident that the next government standard algorithm, Skipjack (see Section 13.12), was classified.

Previous | Table of Contents | Next

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

Adoption of the Standard

The American National Standards Institute (ANSI) approved DES as a private-sector standard in 1981 (ANSI X3.92) [50]. They called it the Data Encryption Algorithm (DEA). ANSI published a standard for DEA modes of operation (ANSI X3.106) [52], similar to the NBS document, and a standard for network encryption that uses DES (ANSI X3.105) [51].

Two other groups within ANSI, representing retail and wholesale banking, developed DES-based standards. Retail banking involves transactions between financial institutions and private individuals, and wholesale banking involves transactions between financial institutions.

ANSI's Financial Institution Retail Security Working Group developed a standard for the management and security of PINs (ANSI X9.8) [53] and another DES-based standard for the authentication of retail financial messages (ANSI X9.19) [56]. The group has a draft standard for secure key distribution (ANSI X9.24) [58].

ANSI's Financial Institution Wholesale Security Working Group developed its own set of standards for message authentication (ANSI X9.9) [54], key management (ANSI X9.17) [55,1151], encryption (ANSI X9.23) [57], and secure personal and node authentication (ANSI X9.26) [59].

The American Bankers Association develops voluntary standards for the financial industry. They published a standard recommending DES for encryption [1], and another standard for managing cryptographic keys [2].

Before the Computer Security Act of 1987, the General Services Administration (GSA) was responsible for developing federal telecommunications standards. Since then, that responsibility transferred to NIST. The GSA published three standards that used DES: two for general

security and interoperability requirements (Federal Standard 1026 [662] and Federal Standard 1027 [663]), and one for Group 3 facsimile equipment (Federal Standard 1028) [664].

The Department of the Treasury wrote policy directives requiring that all electronic-funds transfer messages be authenticated with DES [468,470]. They also wrote DES-based criteria that all authentication devices must meet [469].

The ISO first voted to approve DES—they called it the DEA-1—as an international standard, then decided not to play a role in the standardization of cryptography. However, in 1987 the International Wholesale Financial Standards group of ISO used DES in an international authentication standard [758] and for key management [761]. DES is also specified in an Australian banking standard [1497].

Validation and Certification of DES Equipment

As part of the DES standard, NIST validates implementations of DES. This validation confirms that the implementation follows the standard. Until 1994, NIST only validated hardware and firmware implementations—until then the standard prohibited software implementations. As of March 1995, 73 different implementations had been validated.

NIST also developed a program to certify that authentication equipment conformed to ANSI X9.9 and FIPS 113. As of March, 1995, 33 products had been validated. The Department of the Treasury has an additional certification procedure. NIST also has a program to confirm that equipment conforms to ANSI X9.17 for wholesale key management [1151]; four products have been validated as of March, 1995.

1987

The terms of the DES standard stipulate that it be reviewed every five years. In 1983 DES was recertified without a hitch. In the March 6, 1987 *Federal Register,* NBS published a request for comments on the second five-year review. NBS offered three alternatives for consideration [1480,1481]: reaffirm the standard for another five years, withdraw the standard, or revise the applicability of the standard.

NBS and NSA reviewed the standard. NSA was more involved this time. Because of an executive directive called NSDD-145, signed by Reagan, NSA had veto power over the NBS in matters of cryptography. Initially, the NSA announced that it would not recertify the standard. The problem was not that DES had been broken, or even that it was suspected of having been broken. It was simply increasingly likely that it would soon be broken.

In its place, the NSA proposed the Commercial COMSEC Endorsement Program (CCEP), which would eventually provide a series of algorithms to replace DES [85]. These NSA-designed algorithms would not be made public, and would only be available in tamper-proof VLSI chips (see Section 25.1).

This announcement wasn't well received. People pointed out that business (especially the financial industry) uses DES extensively, and that no adequate

alternative is available. Withdrawal of the standard would leave many organizations with no data protection. After much debate, DES was reaffirmed as a U.S. government standard until 1992 [1141]. According to the NBS, DES would not be certified again [1480].

1993

Never say "not." In 1992, there was still no alternative for DES. The NBS, now called NIST, again solicited comments on DES in the *Federal Register* [540]:

> The purpose of this notice is to announce the review to assess the continued adequacy of the standard to protect computer data. Comments from industry and the public are invited on the following alternatives for FIPS 46-1. The costs (impacts) and benefits of these alternatives should be included in the comments:
>
> — Reaffirm the standard for another five (5) years. The National Institute of Standards and Technology would continue to validate equipment that implements the standard. FIPS 46-1 would continue to be the only approved method for protecting unclassified computer data.
>
> — Withdraw the standard. The National Institute of Standards and Technology would no longer continue to support the standard. Organizations could continue to utilize existing equipment that implements the standard. Other standards could be issued by NIST as a replacement for the DES.
>
> — Revise the applicability and/or implementation statements for the standard. Such revisions could include changing the standard to allow the use of implementations of the DES in software as well as hardware; to allow the iterative use of the DES in specific applications; to allow the use of alternative algorithms that are approved and registered by NIST.

The comment period closed on December 10, 1992. According to Raymond Kammer, then the acting director of NIST [813]:

> Last year, NIST formally solicited comments on the recertification of DES. After reviewing those comments, and the other technical inputs that I have received, I plan to recommend to the Secretary of Commerce that he recertify DES for another five years. I also plan to suggest to the Secretary that when we announce the recertification we state our intention to consider alternatives to it over the next five years. By putting that announcement on the table, we hope to give people an opportunity to comment on orderly technological transitions. In the meantime, we need to consider the large installed base of systems that rely upon this proven standard.

Even though the Office of Technology Assessment quoted NIST's Dennis

Branstead as saying that the useful lifetime of DES would end in the late 1990s [1191], the algorithm was recertified for another five years [1150]. Software implementations of DES were finally allowed to be certified.

Anyone want to guess what will happen in 1998?

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

To access the contents, click the chapter and section titles.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

Bookmark it

Search this book:

Go!

Previous | Table of Contents | Next

12.2 Description of DES

DES is a block cipher; it encrypts data in 64-bit blocks. A 64-bit block of plaintext goes in one end of the algorithm and a 64-bit block of ciphertext comes out the other end. DES is a symmetric algorithm: The same algorithm and key are used for both encryption and decryption (except for minor differences in the key schedule).

The key length is 56 bits. (The key is usually expressed as a 64-bit number, but every eighth bit is used for parity checking and is ignored. These parity bits are the least-significant bits of the key bytes.) The key can be any 56-bit number and can be changed at any time. A handful of numbers are considered weak keys, but they can easily be avoided. All security rests within the key.

At its simplest level, the algorithm is nothing more than a combination of the two basic techniques of encryption: confusion and diffusion. The fundamental building block of DES is a single combination of these techniques (a substitution followed by a permutation) on the text, based on the key. This is known as a **round**. DES has 16 rounds; it applies the same combination of techniques on the plaintext block 16 times (see Figure 12.1).

The algorithm uses only standard arithmetic and logical operations on numbers of 64 bits at most, so it was easily implemented in late 1970s hardware technology. The repetitive nature of the algorithm makes it ideal for use on a special-purpose chip. Initial software implementations were clumsy, but current implementations are better.

Outline of the Algorithm

DES operates on a 64-bit block of plaintext. After an initial permutation, the block is broken into a right half and a left half, each 32 bits long. Then there

are 16 rounds of identical operations, called Function f, in which the data are combined with the key. After the sixteenth round, the right and left halves are joined, and a final permutation (the inverse of the initial permutation) finishes off the algorithm.

In each round (see Figure 12.2), the key bits are shifted, and then 48 bits are selected from the 56 bits of the key. The right half of the data is expanded to 48 bits via an expansion permutation, combined with 48 bits of a shifted and permuted key via an XOR, sent through 8 S-boxes producing 32 new bits, and permuted again. These four operations make up Function f. The output of Function f is then combined with the left half via another XOR. The result of these operations becomes the new right half; the old right half becomes the new left half. These operations are repeated 16 times, making 16 rounds of DES.

If B_i is the result of the *i*th iteration, L_i and R_i are the left and right halves of B_i, K_i is the 48-bit key for round *i,* and f is the function that does all the substituting and permuting and XORing with the key, then a round looks like:

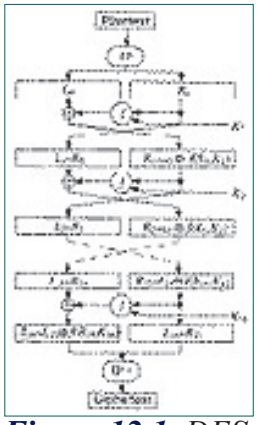


Figure 12.1 *DES.*

$$L_i = R_{i-1}$$

$$R_i = L_{i-1} \bullet f(R_{i-1}, K_i)$$

The Initial Permutation

The initial permutation occurs before round 1; it transposes the input block as described in Table 12.1. This table, like all the other tables in this chapter, should be read left to right, top to bottom. For example, the initial permutation moves bit 58 of the plaintext to bit position 1, bit 50 to bit position 2, bit 42 to bit position 3, and so forth.

The initial permutation and the corresponding final permutation do not affect DES's security. (As near as anyone can tell, its primary purpose is to make it easier to load plaintext and ciphertext data into a DES chip in byte-sized pieces. Remember that DES predates 16-bit or 32-bit microprocessor busses.) Since this bit-wise permutation is difficult in software (although it is trivial in hardware), many software implementations of DES leave out both the initial and final permutations. While this new algorithm is no less secure than DES, it does not follow the DES standard and should not be called DES.



Figure 12.2 *One round of DES.*

The Key Transformation

Initially, the 64-bit DES key is reduced to a 56-bit key by ignoring every eighth bit. This is described in Table 12.2. These bits can be used as parity check to ensure the key is error-free. After the 56-bit key is extracted, a different 48-bit **subkey** is generated for each of the 16 rounds of DES. These subkeys, K_i are determined in the following manner.

First, the 56-bit key is divided into two 28-bit halves. Then, the halves are circularly shifted left by either one or two bits, depending on the round. This shift is given in Table 12.3.

Table 12.1
Initial Permutation

58,	50,	42,	34,	26,	18,	10,	2,	60,	52,	44,	36,	28,	20,	12,	4,
62,	54,	46,	38,	30,	22,	14,	6,	64,	56,	48,	40,	32,	24,	16,	8,
57,	49,	41,	33,	25,	17,	9,	1,	59,	51,	43,	35,	27,	19,	11,	3,
61,	53,	45,	37,	29,	21,	13,	5,	63,	55,	47,	39,	31,	23,	15,	7

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

Table 12.2
Key Permutation

57,	49,	41,	33,	25,	17,	9,	1,	58,	50,	42,	34,	26,	18,
10,	2,	59,	51,	43,	35,	27,	19,	11,	3,	60,	52,	44,	36,
63,	55,	47,	39,	31,	23,	15,	7,	62,	54,	46,	38,	30,	22,
14,	6,	61,	53,	45,	37,	29,	21,	13,	5,	28,	20,	12,	4

After being shifted, 48 out of the 56 bits are selected. Because this operation permutes the order of the bits as well as selects a subset of bits, it is called a **compression permutation**. This operation provides a subset of 48 bits. Table 12.4 defines the compression permutation (also called the permuted choice). For example, the bit in position 33 of the shifted key moves to position 35 of the output, and the bit in position 18 of the shifted key is ignored.

Because of the shifting, a different subset of key bits is used in each subkey. Each bit is used in approximately 14 of the 16 subkeys, although not all bits are used exactly the same number of times.

The Expansion Permutation

This operation expands the right half of the data, R_i, from 32 bits to 48 bits. Because this operation changes the order of the bits as well as repeating certain bits, it is known as an **expansion permutation**. This operation has two purposes: It makes the right half the same size as the key for the XOR operation and it provides a longer result that can be compressed during the substitution operation. However, neither of those is its main cryptographic purpose. By allowing one bit to affect two substitutions, the dependency of the output bits on the input bits spreads faster. This is called an **avalanche effect**. DES is designed to reach the condition of having every bit of the ciphertext

depend on every bit of the plaintext and every bit of the key as quickly as possible.

Figure 12.3 defines the expansion permutation. This is sometimes called the **E-box**. For each 4-bit input block, the first and fourth bits each represent two bits of the output block, while the second and third bits each represent one bit of the output block. Table 12.5 shows which output positions correspond to which input positions. For example, the bit in position 3 of the input block moves to position 4 of the output block, and the bit in position 21 of the input block moves to positions 30 and 32 of the output block.

Although the output block is larger than the input block, each input block generates a unique output block.

Table 12.3
Number of Key Bits Shifted per Round

Round	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Number	1	1	2	2	2	2	2	2	1	2	2	2	2	2	2	1

Table 12.4
Compression Permutation

14,	17,	11,	24,	1,	5,	3,	28,	15,	6,	21,	10,
23,	19,	12,	4,	26,	8,	16,	7,	27,	20,	13,	2,
41,	52,	31,	37,	47,	55,	30,	40,	51,	45,	33,	48,
44,	49,	39,	56,	34,	53,	46,	42,	50,	36,	29,	32

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

The S-Box Substitution

After the compressed key is XORed with the expanded block, the 48-bit result moves to a substitution operation. The substitutions are performed by eight **substitution boxes,** or **S-boxes**. Each S-box has a 6-bit input and a 4-bit output, and there are eight different S-boxes. (The total memory requirement for the eight DES S-boxes is 256 bytes.) The 48 bits are divided into eight 6-bit sub-blocks. Each separate block is operated on by a separate S-box: The first block is operated on by S-box 1, the second block is operated on by S-box 2, and so on. See Figure 12.4.

Each S-box is a table of 4 rows and 16 columns. Each entry in the box is a 4-bit number. The 6 input bits of the S-box specify under which row and column number to look for the output. Table 12.6 shows all eight S-boxes.

The input bits specify an entry in the S-box in a very particular manner. Consider an S-box input of 6 bits, labeled b_1 b_2 b_3 b_4 b_5 and b_6. Bits b_1 and b_6 are combined to form a 2-bit number, from 0 to 3, which corresponds to a row in the table. The middle 4 bits, b_2 through b_5 are combined to form a 4-bit number, from 0 to 15, which corresponds to a column in the table.

For example, assume that the input to the sixth S-box (i.e., bits 31 through 36 of the XOR function) is 110011. The first and last bits combine to form 11, which corresponds to row 3 of the sixth S-box. The middle 4 bits combine to form 1001, which corresponds to the column 9 of the same S-box. The entry under row 3, column 9 of S-box 6 is 14. (Remember to count rows and columns from 0 and not from 1.) The value 1110 is substituted for 110011.

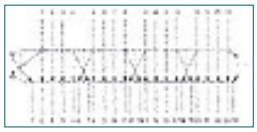


Figure 12.3 *Expansion permutation.*

Table 12.5
Expansion Permutation

32,	1,	2,	3,	4,	5,	4,	5,	6,	7,	8,	9,
8,	9,	10,	11,	12,	13,	12,	13,	14,	15,	16,	17,
16,	17,	18,	19,	20,	21,	20,	21,	22,	23,	24,	25,
24,	25,	26,	27,	28,	29,	28,	29,	30,	31,	32,	1

It is, of course, far easier to implement the S-boxes in software as 64-entry arrays. It takes some rearranging of the entries to do this, but that's not hard. (Don't just change the indexing without rearranging the entries. The S-boxes are designed very carefully.) However, this way of describing the S-boxes helps visualize how they work. Each S-box can be viewed as a substitution function on a 4-bit entry: b_2 through b_5 go in, and a 4-bit result comes out. Bits b_1 and b_6 come from neighboring blocks; they select one out of four substitution functions available in the particular S-box.

The S-box substitution is the critical step in DES. The algorithm's other operations are linear and easy to analyze. The S-boxes are nonlinear and, more than anything else, give DES its security.

The result of this substitution phase is eight 4-bit blocks which are recombined into a single 32-bit block. This block moves to the next step: the P-box permutation.

The P-Box Permutation

The 32-bit output of the S-box substitution is permuted according to a **P-box**. This permutation maps each input bit to an output position; no bits are used twice and no bits are ignored. This is called a **straight permutation** or just a permutation. Table 12.7 shows the position to which each bit moves. For example, bit 21 moves to bit 4, while bit 4 moves to bit 31.



Figure 12.4 *S-box substitution.*

Table 12.6
S-Boxes

S-box 1:															
14,	4,	13,	1,	2,	15,	11,	8,	3,	10,	6,	12,	5,	9,	0,	7,
0,	15,	7,	4,	14,	2,	13,	1,	10,	6,	12,	11,	9,	5,	3,	8,
4,	1,	14,	8,	13,	6,	2,	11,	15,	12,	9,	7,	3,	10,	5,	0,
15,	12,	8,	2,	4,	9,	1,	7,	5,	11,	3,	14,	10,	0,	6,	13,
S-box 2:															
15,	1,	8,	14,	6,	11,	3,	4,	9,	7,	2,	13,	12,	0,	5,	10,
3,	13,	4,	7,	15,	2,	8,	14,	12,	0,	1,	10,	6,	9,	11,	5,

```
    0,  14,   7,  11,  10,   4,  13,   1,   5,   8,  12,   6,   9,   3,   2,  15,
   13,   8,  10,   1,   3,  15,   4,   2,  11,   6,   7,  12,   0,   5,  14,   9,
```

S-box 3:

```
   10,   0,   9,  14,   6,   3,  15,   5,   1,  13,  12,   7,  11,   4,   2,   8,
   13,   7,   0,   9,   3,   4,   6,  10,   2,   8,   5,  14,  12,  11,  15,   1,
   13,   6,   4,   9,   8,  15,   3,   0,  11,   1,   2,  12,   5,  10,  14,   7,
    1,  10,  13,   0,   6,   9,   8,   7,   4,  15,  14,   3,  11,   5,   2,  12,
```

S-box 4:

```
    7,  13,  14,   3,   0,   6,   9,  10,   1,   2,   8,   5,  11,  12,   4,  15,
   13,   8,  11,   5,   6,  15,   0,   3,   4,   7,   2,  12,   1,  10,  14,   9,
   10,   6,   9,   0,  12,  11,   7,  13,  15,   1,   3,  14,   5,   2,   8,   4,
    3,  15,   0,   6,  10,   1,  13,   8,   9,   4,   5,  11,  12,   7,   2,  14,
```

S-box 5:

```
    2,  12,   4,   1,   7,  10,  11,   6,   8,   5,   3,  15,  13,   0,  14,   9,
   14,  11,   2,  12,   4,   7,  13,   1,   5,   0,  15,  10,   3,   9,   8,   6,
    4,   2,   1,  11,  10,  13,   7,   8,  15,   9,  12,   5,   6,   3,   0,  14,
   11,   8,  12,   7,   1,  14,   2,  13,   6,  15,   0,   9,  10,   4,   5,   3,
```

S-box 6:

```
   12,   1,  10,  15,   9,   2,   6,   8,   0,  13,   3,   4,  14,   7,   5,  11,
   10,  15,   4,   2,   7,  12,   9,   5,   6,   1,  13,  14,   0,  11,   3,   8,
    9,  14,  15,   5,   2,   8,  12,   3,   7,   0,   4,  10,   1,  13,  11,   6,
    4,   3,   2,  12,   9,   5,  15,  10,  11,  14,   1,   7,   6,   0,   8,  13,
```

S-box 7:

```
    4,  11,   2,  14,  15,   0,   8,  13,   3,  12,   9,   7,   5,  10,   6,   1,
   13,   0,  11,   7,   4,   9,   1,  10,  14,   3,   5,  12,   2,  15,   8,   6,
    1,   4,  11,  13,  12,   3,   7,  14,  10,  15,   6,   8,   0,   5,   9,   2,
    6,  11,  13,   8,   1,   4,  10,   7,   9,   5,   0,  15,  14,   2,   3,  12,
```

S-box 8:

```
   13,   2,   8,   4,   6,  15,  11,   1,  10,   9,   3,  14,   5,   0,  12,   7,
    1,  15,  13,   8,  10,   3,   7,   4,  12,   5,   6,  11,   0,  14,   9,   2,
    7,  11,   4,   1,   9,  12,  14,   2,   0,   6,  10,  13,  15,   3,   5,   8,
    2,   1,  14,   7,   4,  10,   8,  13,  15,  12,   9,   0,   3,   5,   6,  11
```

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

Table 12.7
P-Box Permutation

16,	7,	20,	21,	29,	12,	28,	17,	1,	15,	23,	26,	5,	18,	31,	10,
2,	8,	24,	14,	32,	27,	3,	9,	19,	13,	30,	6,	22,	11,	4,	25

Finally, the result of the P-box permutation is XORed with the left half of the initial 64-bit block. Then the left and right halves are switched and another round begins.

The Final Permutation

The final permutation is the inverse of the initial permutation and is described in Table 12.8. Note that the left and right halves are not exchanged after the last round of DES; instead the concatenated block $R_{16}L_{16}$ is used as the input to the final permutation. There's nothing going on here; exchanging the halves and shifting around the permutation would yield exactly the same result. This is so that the algorithm can be used to both encrypt and decrypt.

Decrypting DES

After all the substitutions, permutations, XORs, and shifting around, you might think that the decryption algorithm is completely different and just as confusing as the encryption algorithm. On the contrary, the various operations were chosen to produce a very useful property: The same algorithm works for both encryption and decryption.

With DES it is possible to use the same function to encrypt or decrypt a block. The only difference is that the keys must be used in the reverse order. That is, if the encryption keys for each round are K_1 K_2 K_3,..., K_{16} then the decryption

keys are K_{16} K_{15} $K_{14,}$ ..., K_1. The algorithm that generates the key used for each round is circular as well. The key shift is a right shift and the number of positions shifted is 0,1,2,2,2,2,2,2,1,2,2,2,2,2,2,1.

Modes of DES

FIPS PUB 81 specifies four modes of operation: ECB, CBC, OFB, and CFB (see Chapter 9) [1143]. The ANSI banking standards specify ECB and CBC for encryption, and CBC and *n*-bit CFB for authentication [52].

In the software world, certification is usually not an issue. Because of its simplicity, ECB is most often used in off-the-shelf commercial software products, although it is the most vulnerable to attack. CBC is used occasionally, even though it is just slightly more complicated than ECB and provides much more security.

Table 12.8
Final Permutation

40,	8,	48,	16,	56,	24,	64,	32,	39,	7,	47,	15,	55,	23,	63,	31,
38,	6,	46,	14,	54,	22,	62,	30,	37,	5,	45,	13,	53,	21,	61,	29,
36,	4,	44,	12,	52,	20,	60,	28,	35,	3,	43,	11,	51,	19,	59,	27,
34,	2,	42,	10,	50,	18,	58,	26,	33,	1,	41,	9,	49,	17,	57,	25

Hardware and Software Implementations of DES

Much has been written on efficient hardware and software implementations of the algorithm [997,81,533,534,437,738,1573,176,271,1572]. At this writing, the recordholder for the fastest DES chip is a prototype developed at Digital Equipment Corporation [512]. It supports ECB and CBC modes and is based on a GaAs gate array of 50,000 transistors. Data can be encrypted and decrypted at a rate of 1 gigabit per second, which translates to 16.8 million blocks per second. This is impressive. Table 12.9 gives the specifications for some commercial DES chips. Seeming discrepancies between clock speed and data rate are due to pipelining within the chip; a chip might have multiple DES engines working in parallel.

The most impressive DES chip is VLSI's 6868 (formerly called "Gatekeeper"). Not only can it perform DES encryption in only 8 clock cycles (prototypes in the lab can do it in 4 clock cycles), but it can also do ECB triple-DES in 25 clock cycles, and OFB or CBC triple-DES in 35 clock cycles. This sounds impossible to me, too, but I assure you it works.

A software implementation of DES on an IBM 3090 mainframe can perform 32,000 DES encryptions per second. Most microcomputers are slower, but impressive nonetheless. Table 12.10 [603,793] gives actual results and estimates for various Intel and Motorola microprocessors.

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

12.3 Security of DES

People have long questioned the security of DES [458]. There has been much speculation on the key length, number of iterations, and design of the S-boxes. The S-boxes were particularly mysterious—all those constants, without any apparent reason as to why or what they're for. Although IBM claimed that the inner workings were the result of 17 man-years of intensive cryptanalysis some people feared that the NSA embedded a trapdoor into the algorithm so they would have an easy means of decrypting messages.

The U.S. Senate Select Committee on Intelligence, with full top-secret clearances, investigated the matter in 1978. The findings of the committee are classified, but an unclassified summary of those findings exonerated the NSA from any improper involvement in the algorithm's design [1552]. "It was said to have convinced IBM that a shorter key was adequate, to have indirectly assisted in the development of the S-box structures and to have certified that the final DES algorithm was, to the best of their knowledge, free of any statistical or mathematical weaknesses" [435]. However, since the government never made the details of the investigation public, many people remained unconvinced.

Tuchman and Meyer, two of the IBM cryptographers who designed DES, said the NSA did not alter the design [841]:

> Their basic approach was to look for strong substitution, permutation, and key scheduling functions.... IBM has classified the notes containing the selection criteria at the request of the NSA.... "The NSA told us we had inadvertently reinvented some of the deep secrets it uses to make its own algorithms," explains Tuchman.

Table 12.9
Commercial DES Chips

Manufacturer	Chip	Year	Clock	Data Rate	Availability
AMD	Am9518	1981	3 MHz	1.3 MByte/s	N
AMD	Am9568	?	4 MHz	1.5 MByte/s	N
AMD	AmZ8068	1982	4 MHz	1.7 MByte/s	N
AT&ampT	T7000A	1985	?	1.9 MByte/s	N
CE-Infosys	SuperCrypt CE99C003	1992	20 MHz	12.5 MByte/s	Y
CE-Infosys	SuperCrypt CE99C003A	1994	30 MHz	20.0 MByte/s	Y
Cryptech	Cry12C102	1989	20 MHz	2.8 MByte/s	Y
Newbridge	CA20C03A	1991	25 MHz	3.85 MByte/s	Y
Newbridge	CA20C03W	1992	8 MHz	0.64 MByte/s	Y
Newbridge	CA95C68/18/09	1993	33 MHz	14.67 MByte/s	Y
Pijnenburg	PCC100	?	?	2.5 MByte/s	Y
Semaphore Communications	Roadrunner284	?	40 MHz	35.5 MByte/s	Y
VLSI Technology	VM007	1993	32 MHz	200.0 MByte/s	Y
VLSI Technology	VM009	1993	33 MHz	14.0 MByte/s	Y
VLSI Technology	6868	1995	32 MHz	64.0 MByte/s	Y
Western Digital	WD2001/2002	1984	3 MHz	0.23 MByte/s	N

Table 12.10
DES Speeds on Different Microprocessors and Computers

Processor	Speed (in MHz)	DES Blocks (per second)
8088	4.7	370
68000	7.6	900
80286	6	1,100
68020	16	3,500
68030	16	3,900
80386	25	5,000
68030	50	10,000

68040	25	16,000
68040	40	23,000
80486	66	43,000
Sun ELC		26,000
HyperSparc		32,000
RS6000-350		53,000
Sparc 10/52		84,000
DEC Alpha 4000/610		154,000
HP 9000/887	125	196,000

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

Later in the article, Tuchman is quoted: “We developed the DES algorithm entirely within IBM using IBMers. The NSA did not dictate a single wire!” Tuchman reaffirmed this when he spoke on the history of DES at the 1992 National Computer Security Conference.

On the other hand, Coppersmith wrote [373,374]: “The National Security Agency (NSA) also provided technical advice to IBM.” And Konheim has been quoted as saying: “We sent the S-boxes off to Washington. They came back and were all different. We ran our tests and they passed.” People have pointed to this as evidence that the NSA put a trapdoor in DES.

NSA, when questioned regarding any imposed weakness in DES, said [363]:

> Regarding the Data Encryption Standard (DES), we believe that the public record from the Senate Committee for Intelligence’s investigation in 1978 into NSA’s role in the development of the DES is responsive to your question. That committee report indicated that NSA did not tamper with the design of the algorithm in any way and that the security afforded by the DES was more than adequate for at least a 5–10 year time span for the unclassified data for which it was intended. In short, NSA did not impose or attempt to impose any weakness on the DES.

Then why did they modify the S-boxes? Perhaps it was to ensure that IBM did not put a trapdoor in DES. The NSA had no reason to trust IBM’s researchers, and would be lax in their duty if they did not make absolutely sure that DES was free of trapdoors. Dictating the S-boxes is one way they could make sure.

Very recently some new cryptanalysis results have shed some light on this issue, but for many years this has been the subject of much speculation.

Weak Keys

Because of the way the initial key is modified to get a subkey for each round of the algorithm, certain initial keys are **weak keys** [721,427]. Remember that the initial value is split into two halves, and each half is shifted independently. If all the bits in each half are either 0 or 1, then the key used for any cycle of the algorithm is the same for all the cycles of the algorithm. This can occur if the key is entirely 1s, entirely 0s, or if one half of the key is entirely 1s and the other half is entirely 0s. Also, two of the weak keys have other properties that make them less secure [427].

The four weak keys are shown in hexadecimal notation in Table 12.11. (Remember that every eighth bit is a parity bit.)

Additionally, some pairs of keys encrypt plaintext to the identical ciphertext. In other words, one key in the pair can decrypt messages encrypted with the other key in the pair. This is due to the way in which DES generates subkeys; instead of generating 16 different subkeys, these keys generate only two different subkeys. Each of these subkeys is used eight times in the algorithm. These keys are called **semiweak keys,** and are shown in hexadecimal notation in Table 12.12.

Table 12.11
DES Weak Keys

Weak Key Value		(with parity bits)		Actual Key
0101	0101	0101	0101	0000000 0000000
1F1F	1F1F	0E0E	0E0E	0000000 FFFFFFF
E0E0	E0E0	F1F1	F1F1	FFFFFFF 0000000
FEFE	FEFE	FEFE	FEFE	FFFFFFF FFFFFFF

Some keys produce only four subkeys, each used four times in the algorithm. These **possibly weak** keys are listed in Table 12.13.

Before condemning DES for having weak keys, consider that this list of 64 keys is minuscule compared to the total set of 72,057,594,037,927,936 possible keys. If you select a random key, the odds of picking one of these keys is negligible. If you are truly paranoid, you could always check for weak keys during key generation. Some people don't think it's worth the bother. Others say that it's so easy to check, there's no reason not to.

There is further analysis on weak and semiweak keys in [1116], and additional key patterns have been investigated for weaknesses. None have been found.

Complement Keys

Take the bit-wise complement of a key; that is, replace all the 0s with 1s and the 1s with 0s. Now, if the original key encrypts a block of plaintext, then the complement of the key will encrypt the complement of the plaintext block into the complement of the ciphertext block.

If *x'* is the complement of *x,* then the identity is as follows:

$$E_K(P) = C$$

$$E_K'(P') = C'$$

This isn't anything mysterious. The subkeys are XORed with the right half after the expansion permutation in every round. This **complementation property** is a direct result of that fact.

Table 12.12
DES Semiweak Key Pairs

01FE	01FE	01FE	01FE	and	FE01	FE01	FE01	FE01
1FE0	1FE0	0EF1	0EF1	and	E01F	E01F	F10E	F10E
01E0	01E0	01F1	01F1	and	E001	E001	F101	F101
1FFE	1FFE	0EFE	0EFE	and	FE1F	FE1F	FE0E	FE0E
011F	011F	010E	010E	and	1F01	1F01	0E01	0E01
E0FE	E0FE	F1FE	F1FE	and	FEE0	FEE0	FEF1	FEF1

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

Table 12.13
DES Possibly Weak Keys

1F 1F 01 01 0E 0E 01 01 01 1F 1F 01 01 0E 0E 01 1F 01 01 1F 0E 01 01 0E 01 01 1F 1F 01 01 0E 0E	E0 01 01 E0 F1 01 01 F1 FE 1F 01 E0 FE 0E 01 F1 FE 01 1F E0 FE 01 0E F1 E0 1F 1F E0 F1 0E 0E F1
E0 E0 01 01 F1 F1 01 01 FE FE 01 01 FE FE 01 01 FE E0 1F 01 FE F1 0E 01 E0 FE 1F 01 F1 FE 0E 01	FE 01 01 FE FE 01 01 FE E0 1F 01 FE F1 0E 01 FE E0 01 1F FE F1 01 0E FE FE 1F 1F FE FE 0E 0E FE
FE E0 01 1F FE F1 01 0E E0 FE 01 1F F1 FE 01 0E E0 E0 1F 1F F1 F1 0E 0E FE FE 1F 1F FE FE 0E 0E	1F FE 01 E0 0E FE 01 F1 01 FE 1F E0 01 FE 0E F1 1F E0 01 FE 0E F1 01 FE 01 E0 1F FE 01 F1 0E FE
FE 1F E0 01 FE 0E F1 01 E0 1F FE 01 F1 0E FE 01 FE 01 E0 1F FE 01 F1 0E E0 01 FE 1F F1 01 FE 0E	01 01 E0 E0 01 01 F1 F1 1F 1F E0 E0 0E 0E F1 F1 1F 01 FE E0 0E 01 FE F1 01 1F FE E0 01 0E FE F1
01 E0 E0 01 01 F1 F1 01 1F FE E0 01 0E FE F0 01 1F E0 FE 01 0E F1 FE 01 01 FE FE 01 01 FE FE 01	1F 01 E0 FE 0E 01 F1 FE 01 1F E0 FE 01 0E F1 FE 01 01 FE FE 01 01 FE FE 1F 1F FE FE 0E 0E FE FE

1F E0 E0 1F 0E F1 F1 0E	FE FE E0 E0 FE FE F1 F1
01 FE E0 1F 01 FE F1 0E	E0 FE FE E0 F1 FE FE F1
01 E0 FE 1F 01 F1 FE 0E	FE E0 E0 FE FE F1 F1 FE
1F FE FE 1F 0E FE FE 0E	E0 E0 FE FE F1 F1 FE FE

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

What this means is that a chosen-plaintext attack against DES only has to test half the possible keys: 2^{55} keys instead of 2^{56} [1080]. Eli Biham and Adi Shamir showed [172] that there is a known-plaintext attack of the same complexity, requiring at least 2^{33} known plaintexts.

It is questionable whether this is a weakness, since most messages don't have complement blocks of plaintext (in random plaintext, the odds against it are extremely high) and users can be warned not to use complement keys.

Algebraic Structure

All possible 64-bit plaintext blocks can be mapped onto all possible 64-bit ciphertext blocks in $2^{64}!$ different ways. The DES algorithm, with its 56-bit key, gives us 2^{56} (approximately 10^{17}) of these mappings. Using multiple encryption, it seems possible to reach a larger portion of those possible mappings. However, this is only true if the DES operation does not have certain algebraic structures.

If DES were **closed,** then for any K_1 and K_2 there would always be a K_3 such that

$$E_K2(E_K1(P)) = E_K3(P)$$

In other words, the DES encryption operation would form a group, and encrypting a set of plaintext blocks with K_1 followed by K_2 would be identical to encrypting the blocks with K_3. Even worse, DES would be vulnerable to a meet-in-the-middle known-plaintext attack that runs in only 228 steps [807].

If DES were **pure,** then for any K_1 K_2 and K_3 there would always be a K_4 such that

$$E_K3(E_K2(E_K1(P))) = E_K4(P)$$

Triple encryption would be useless. (Note that a closed cipher is necessarily pure but a pure cipher is not necessarily closed.)

An early theoretical paper by Don Coppersmith gave some hints, but it wasn't enough [377]. Various cryptographers wrestled with this question [588,427,431,527,723,789]. Cycling experiments gathered "overwhelming evidence" that DES is not a group [807,371,808,1116,809], but it wasn't until 1992 that cryptographers proved that DES is not a group [293]. Coppersmith said that the IBM team knew it all along.

Key Length

IBM's original submission to NBS had a 112-bit key. By the time the DES became a standard, that was reduced to a 56-bit key. Many cryptographers argued for the longer key. Their arguments centered on the possibility of a brute-force attack (see Section 7.1).

In 1976 and 1977, Diffie and Hellman argued that a special-purpose DES-cracking parallel computer could recover the key in a day and cost $20 million. In 1981, Diffie upped this to a two-day search time and a cost of $50 million [491]. Diffie and Hellman argued then that this was out of reach for everybody except organizations like the NSA, but that by 1990 DES would be totally insecure [714].

Hellman [716] presented another argument against the small key size: By trading memory space for time, it would be possible to speed up the searching process. He suggested the possibility of computing and storing 256 possible results of encrypting a single plaintext block under every possible key. Then, to break an unknown key, all that would be required would be for the cryptanalyst to insert the plaintext block into the encryption stream, recover the resulting ciphertext, and look the key up. Hellman pegged the cost of this cracking machine at $5 million.

Arguments for and against the existence of a DES-cracker lurking in some government basement somewhere have continued. Several people pointed out that the mean time between failures for the DES chips would never be high enough to ensure that the machine would work. This objection was shown to be superfluous in [1278]. Others suggested ways to speed the process even further and to reduce the effects of chip failures.

Meanwhile, hardware implementations of DES slowly approached the million-encryptions-per-second requirement of Diffie and Hellman's special-purpose machine. In 1984 DES chips capable of performing 256,000 encryptions per second had been produced [533,534]. By 1987 chips performing 512,000 encryptions per second were being developed, and a version capable of checking over a million keys per second was feasible [738,1573]. And in 1993 Michael Wiener designed a $1 million machine that could complete a brute-force attack against DES in an average of 3.5 hours (see Section 7.1).

No one has publicly admitted building this machine, although it is a reasonable assumption that someone has. A million dollars is not a lot of money to a

large—or even a medium-sized—country.

It was not until 1990 that two Israeli mathematicians, Biham and Shamir, discovered **differential cryptanalysis,** a technique that put to rest the question of key length. Before we discuss that technique, let's turn to some other design criticisms of DES.

Number of Rounds

Why 16 rounds? Why not 32? After five rounds every ciphertext bit is a function of every plaintext bit and every key bit [1078,1080], and after eight rounds the ciphertext was essentially a random function of every plaintext bit and every key bit [880]. (This is called the avalanche effect.) So why not stop after eight rounds?

Over the years, variants of DES with a reduced number of rounds have been successfully attacked. DES with three or four rounds was easily broken in 1982 [49]. DES with six rounds fell some years later [336]. Biham and Shamir's differential cryptanalysis explained this as well: DES with any number of rounds fewer than 16 could be broken with a known-plaintext attack more efficiently than by a brute-force attack. Certainly brute-force is a much more likely attack, but it is interesting that the algorithm has exactly 16 rounds.

Design of the S-Boxes

In addition to being accused of reducing the key length, NSA was also accused of modifying the contents of the S-boxes. When pressed for design justification for the S-boxes, the NSA indicated that elements of the algorithm's design were "sensitive" and would not be made public. Many cryptographers were concerned that the NSA-designed S-boxes hid a trapdoor, making it possible for them to easily cryptanalyze the algorithm.

Since then, considerable effort has gone into analyzing the design and operation of the S-boxes. In the mid-1970s, Lexar Corporation [961,721] and Bell Laboratories [1120] examined the operation of the S-boxes. Neither analysis revealed any weaknesses, although both found inexplicable features. The S-boxes had more features in common with a linear transformation than one would expect if they were chosen at random. The Bell Laboratories team stated that the S-boxes may have hidden trapdoors, and the Lexar report concluded with:

> Structures have been found in DES that were undoubtedly inserted to strengthen the system against certain types of attack. Structures have also been found that appear to weaken the system.

On the other hand, this report also warned:

> ...the problem [of the search for structure in the S-boxes] is complicated by the ability of the human mind to find apparent structure in random data, which is really not structure at all.

At the second workshop on DES, the National Security Agency revealed

several design criteria behind the S-boxes [229]. This did nothing to quell people's suspicions, and the debate continued [228,422,714,1506,1551].

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

Various oddities about the S-boxes appeared in the literature. The last three output bits of the fourth S-box can be derived in the same way as the first by complementing some of the input bits [436,438]. Two different, but carefully chosen, inputs to S-boxes can produce the same output [436]. It is possible to obtain the same output of a single DES round by changing bits in only three neighboring S-boxes [487]. Shamir noticed that the S-boxes entries appeared to be somewhat imbalanced, but wasn't about to turn that imbalance into an attack [1423]. (He mentioned a feature of the fifth S-box, but it took another eight years before linear cryptanalysis exploited that feature.) Other researchers showed that publicly known design principles could be used to generate S-boxes with the observed characteristics [266].

Additional Results

There were other attempts to cryptanalyze DES. One cryptographer looked at nonrandomness based on spectral tests [559]. Others analyzed sequences of linear factors, but their attack failed after eight rounds [1297,336,531]. A 1987 unpublished attack by Donald Davies exploited the way the expansion permutation repeats bits into adjacent S-boxes; this attack is also impractical after eight rounds [172,429].

12.4 Differential and Linear Cryptanalysis

Differential Cryptanalysis

In 1990, Eli Biham and Adi Shamir introduced **differential cryptanalysis** [167,168,171,172]. This is a new method of cryptanalysis, heretofore unknown to the public. Using this method, Biham and Shamir found a chosen-plaintext attack against DES that was more efficient than brute force.

Differential cryptanalysis looks specifically at **ciphertext pairs**: pairs of

ciphertexts whose plaintexts have particular differences. It analyzes the evolution of these differences as the plaintexts propagate through the rounds of DES when they are encrypted with the same key.

Simply, choose pairs of plaintexts with a fixed difference. The two plaintexts can be chosen at random, as long as they satisfy particular difference conditions; the cryptanalyst does not even have to know their values. (For DES, the term "difference" is defined using XOR. This can be different for different algorithms.) Then, using the differences in the resulting ciphertexts, assign different probabilities to different keys. As you analyze more and more ciphertext pairs, one key will emerge as the most probable. This is the correct key.

The details are more complicated. Figure 12.5 is the DES round function. Imagine a pair of inputs, X and X', that have the difference DX. The outputs, Y and Y' are known, and therefore so is the difference, Y. Both the expansion permutation and the P-box are known, so ΔA and ΔC are known. B and B' are not known, but their difference ΔB is known and equal to ΔA. (When looking at the difference, the XORing of K_i with A and A' cancels out.) So far, so good. Here's the trick: For any given ΔA, not all values of ΔC are equally likely. The combination of ΔA and ΔC suggests values for bits of A XOR K_i and A' XOR K_i. Since A and A' are known, this gives us information about K_i.

Look at the last round of DES. (Differential cryptanalysis ignores the initial and final permutation. They have no effect on the attack, except to make it harder to explain.) If we can identify K_{16} then we have 48 bits of the key. (Remember, the subkey in each round consists of 48 bits of the 56-bit key.) The other 8 bits we can get by brute force. Differential cryptanalysis will get us K_{16}.

Certain differences in plaintext pairs have a high probability of causing certain differences in the resulting ciphertext pairs. These are called **characteristics**. Characteristics extend over a number of rounds and essentially define a path through these rounds. There is an input difference, a difference at each round, and an output difference—with a specific probability.

You can find these characteristics by generating a table where the rows represent the possible input XORs (the XOR of two different sets of input bits), the columns represent the possible output XORs, and the entries represent the number of times a particular output XOR occurs for a given input XOR. You can generate such a table for each of DES's eight S-boxes.

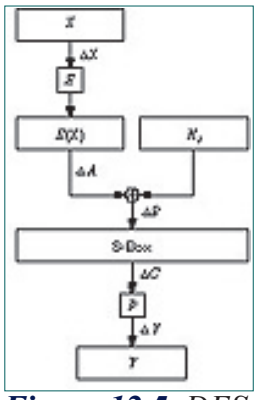


Figure 12.5 *DES round function.*

For example, Figure 12.6a is a one-round characteristic. They input difference of the left side is *L;* it could be anything. The input difference of the right side is 0. (The two inputs have the same right-hand side, so their difference is 0.) Since there is no difference going in to the round function, then there is no difference coming out of the round function. Therefore, the output difference of the left side is $L \bullet 0 = L$, and the output difference of the right side is 0. This is a trivial characteristic, and is true with probability 1.

Figure 12.6b is a less obvious characteristic. Again, the input difference to the left side is arbitrary: *L.* The input difference to the right side is 0x60000000; the two inputs differ in only the second and third bits. With a probability of $^{14}/_{64}$, the output difference of the round function is $L \bullet$ 0x00808200. This means that the output difference of the left side is $L \bullet$ 0x00808200 and the output difference of the right side is 0x60000000—with probability $^{14}/_{64}$.

Different characteristics can be joined. And, assuming the rounds are independent, the probabilities can be multiplied. Figure 12.7 joins the two characteristics previously described. The input difference to the left side is 0x00808200 and the input difference to the right side is 0x60000000. At the end of the first round the input difference and the output of the round function cancel out, leaving an output difference of 0. This feeds into the second round; the final output difference of the left side is 0x60000000 and the final output difference of the right side is 0. This two-round characteristic has a probability of $^{14}/_{64}$.

A plaintext pair that satisfies the characteristic is a **right pair**. A plaintext pair which does not is a **wrong pair**. A right pair will suggest the correct round key (for the last round of the characteristic); a wrong pair will suggest a random round key.

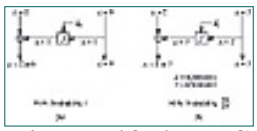


Figure 12.6 *DES characteristics.*

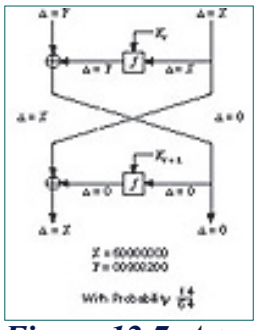


Figure 12.7 *A two-round DES characteristic.*

To find the correct round key, simply collect enough guesses so that one subkey is suggested more often than all the others. In effect, the correct subkey will rise out of all the random alternatives.

So, the basic differential attack on *n*-round DES will recover the 48-bit subkey used in round *n,* and the remaining 8 key bits are obtained by brute-force guessing.

There are still considerable problems. First, there is a negligible chance of success until you reach some threshold. That is, until you accumulate sufficient data you can't tell the correct subkey from all the noise. And the attack isn't practical: You have to use counters to assign different probabilities to 248 possible subkeys, and too much data is required to make this work.

At this point, Biham and Shamir tweaked their attack. Instead of a using a 15-round characteristic on 16-round DES, they used a 13-round characteristic and some tricks to get the last few rounds. A shorter characteristic with a higher probability worked better. And they used some clever mathematics to obtain 56-bit key candidates which could be tested immediately, eliminating the need for counters. This attack succeeds as soon as a right pair is found; this avoids the threshold and gives a linear success probability. If you have 1000 times fewer pairs, then you have 1000 times smaller chance of success. This sounds terrible, but it is a lot better than the threshold. There is always some chance of immediate success.

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

The results are most interesting. Table 12.14 is a summary of the best differential attack against DES with varying numbers of rounds [172]. The first column is the number of rounds. The next two columns are the numbers of chosen plaintexts or known plaintexts that must be examined for the attack, and the fourth column is the number of those plaintexts actually analyzed. The last column is the complexity of analysis, after the required plaintexts are found.

Table 12.14
Differential Cryptanalysis Attacks against DES

No. of Rounds	Chosen Plaintexts	Known Plaintexts	Analyzed Plaintexts	Complexity of Analysis
8	2^{14}	2^{38}	4	2^{9}
9	2^{24}	2^{44}	2	2^{32}
10	2^{24}	2^{43}	2^{14}	2^{15}
11	2^{31}	2^{47}	2	2^{32}
12	2^{31}	2^{47}	2^{21}	2^{21}
13	2^{39}	2^{52}	2	2^{32}
14	2^{39}	2^{51}	2^{29}	2^{29}
15	2^{47}	2^{56}	2^{7}	2^{37}
16	2^{47}	2^{55}	2^{36}	2^{37}

The complexity of the analysis can be greatly reduced for these variants by using about four times as many plaintexts with the clique method.

The best attack against full 16-round DES requires 2^{47} chosen plaintexts. This

can be converted to a known plaintext attack, but that requires 2^{55} known plaintexts. And 2^{37} DES operations are required during analysis.

Differential cryptanalysis works against DES and other similar algorithms with constant S-boxes. The attack is heavily dependent on the structure of the S-boxes; the ones in DES just happen to be optimized against differential cryptanalysis. And the attack works against DES in any of its operating modes—ECB, CBC, CFB, and OFB—with the same complexity [172].

DES's resistance can be improved by increasing the number of rounds. Chosen-plaintext differential cryptanalysis DES with 17 or 18 rounds takes about the same time as a brute-force search [160]. At 19 rounds or more, differential cryptanalysis becomes impossible because it requires more than 2^{64} chosen plaintexts: Remember, DES has a 64-bit block size, so it only *has* 2^{64} possible plaintext blocks. (In general, you can prove that an algorithm is resistant to differential cryptanalysis by showing that the amount of plaintext required to mount such an attack is greater than the amount of plaintext possible.)

Here are a few important points. First, this attack is largely theoretical. The enormous time and data requirements to mount a differential cryptanalytic attack put it beyond the reach of almost everyone. To get the requisite data for this attack against a full DES, you have to encrypt a 1.5 megabits-per-second data stream of *chosen plaintext* for almost three years. Second, this is primarily a chosen-plaintext attack. It can be converted to a known-plaintext attack, but you have to sift through all of the plaintext-ciphertext pairs looking for the useful ones. For full 16-round DES, this makes the attack slightly less efficient than brute force (the differential cryptanalytic attack requires $2^{55.1}$ operations, and brute force requires 2^{55}). The consensus is that DES, when implemented properly, is still secure against differential cryptanalysis.

Why is DES so resistant to differential cryptanalysis? Why are the S-boxes optimized to make this attack as difficult as possible? Why are there as many rounds as required, but no more? Because the designers knew about it. IBM's Don Coppersmith recently wrote [373,374]:

> The design took advantage of certain cryptanalytic techniques, most prominently the technique of "differential cryptanalysis," which were not known in the published literature. After discussions with NSA, it was decided that disclosure of the design consideration would reveal the technique of differential cryptanalysis, a powerful technique that can be used against many ciphers. This in turn would weaken the competitive advantage the United States enjoyed over other countries in the field of cryptography.

Adi Shamir responded to this, challenging Coppersmith to say that he hadn't found any stronger attacks against DES since then. Coppersmith has chosen to remain silent on that question [1426].

Related-Key Cryptanalysis

Table 12.3 showed the number of bits the DES key is rotated after each round: 2 bits after each round, except for 1 bit after rounds 1, 2, 9, and 16. Why?

Related-key cryptanalysis is similar to differential cryptanalysis, but it examines the difference between keys. The attack is different from any previously discussed: The cryptanalyst chooses a relationship between a pair of keys, but does not know the keys themselves. Data is encrypted with both keys. In the known-plaintext version, the cryptanalyst knows the plaintext and ciphertext of data encrypted with the two keys. In the chosen-plaintext version, the cryptanalyst gets to choose the plaintext encrypted with the two keys.

A modified DES, where the key is rotated two bits after every round, is less secure. Related-key cryptanalysis can break that variant using 2^{17} chosen-key chosen plaintexts or 2^{33} chosen-key known plaintexts [158,163].

This attack is not at all practical, but it is interesting for three reasons. One, it is the first cryptanalytic attack against DES's subkey-generation algorithm. Two, this attack is independent of the number of rounds of the cryptographic algorithm; it's just as effective against DES with 16 rounds, 32 rounds, or 1000 rounds. And three, DES is impervious to this attack. The variability in the rotation thwarts related-key cryptanalysis.

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

Linear Cryptanalysis

Linear cryptanalysis is another type of cryptanalytic attack, invented by Mitsuru Matsui [1016,1015,1017]. This attack uses linear approximations to describe the action of a block cipher (in this case, DES.)

This means that if you XOR some of the plaintext bits together, XOR some ciphertext bits together, and then XOR the result, you will get a single bit that is the XOR of some of the key bits. This is a linear approximation and will hold with some probability *p*. If *p* ` ½, then this bias can be exploited. Use collected plaintexts and associated ciphertexts to guess the values of the key bits. The more data you have, the more reliable the guess. The greater the bias, the greater the success rate with the same amount of data.

How do you identify good linear approximations for DES? Find good 1-round linear approximations and join them together. (Again, ignore the initial and final permutations; they don't affect the attack.) Look at the S-boxes. There are 6 input bits and 4 output bits. The input bits can be combined using XOR in 63 useful ways ($2^6 - 1$), and the output bits can be combined in 15 useful ways. Now, for each S-box you can evaluate the probability that for a randomly chosen input, an input XOR combination equals some output XOR combination. If there is a combination with a high enough bias, then linear cryptanalysis may work.

If the linear approximations are unbiased, then they would hold for 32 of the 64 possible inputs. I'll spare you the pages of tables, but the most biased S-box is S-box 5. In fact, the second input bit is equal to the XOR of all 4 output bits for only 12 inputs. This translates to a probability of $^3/_{16}$, or a bias of $^5/_{16}$, and is the most extreme bias in all the S-boxes. (Shamir noted this in [1423], but could not find a way to exploit it.)

Figure 12.8 shows how to turn this into an attack against the DES round function. The input bit into S-box 5 is b_{26}. (I am numbering the bits from left to right and from 1 to 64. Matsui ignores this convention with DES and numbers his bits from right to left and from 0 to 63. It's enough to drive you mad.) The 4 output bits from S-box 5 are c_{17} c_{18} c_{19} and c_{20}. We can trace b_{26} backwards from the input to the S-box. The bit a_{26} is XORed with a bit from the subkey $K_{i,26}$ to obtain b_{26}. And bit X_{17} goes through the expansion permutation to become a_{26}. After the S-box, the 4 output bits go through the P-box to become 4 output bits of the round function: Y_3 Y_8 Y_{14} and Y_{25}. This means that with probability $\frac{1}{2} - \frac{5}{16}$:

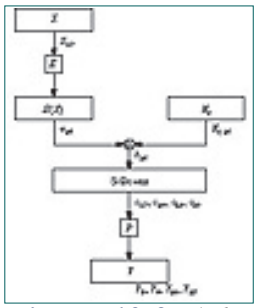


Figure 12.8 *A 1-round linear approximation for DES.*

$$X_{17} \bullet Y_3 \bullet Y_8 \bullet Y_{14} \bullet Y_{25} = K_{i,26}$$

Linear approximations for different rounds can be joined in a manner similar to that discussed under differential cryptanalysis. Figure 12.9 is a 3-round approximation with a probability of $\frac{1}{2} + .0061$. The individual approximations are of varying quality: The last is very good, the first is pretty good, and the middle is bad. But together the three 1-round approximations give a very good three-round approximation.

The basic attack is to use the best linear approximation for 16-round DES. It requires 2^{47} known plaintext blocks, and will result in 1 key bit. That's not very useful. If you interchange the role of plaintext and ciphertext and use decryption as well as encryption, you can get 2 key bits. That's still not very useful.

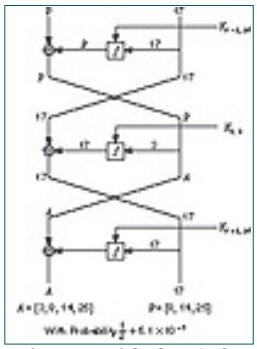


Figure 12.9 *A 3-round linear approximation for DES.*

There are refinements. Use a 14-round linear approximation for rounds 2 through 15. Guess the 6 subkey bits relevant to S-box 5 for the first and last rounds (12 key bits in all). Effectively you are doing 2^{12} linear cryptanalyses in parallel and picking the correct one based on probabilities. This recovers the

12 bits plus the b_{26} and reversing plaintext and ciphertext recovers another 13 bits. To get the remaining 30 bits, use exhaustive search. There are other tricks, but that's basically it.

Against full 16-round DES, this attack can recover the key with an average of 2^{43} known plaintexts. A software implementation of this attack recovered a DES key in 50 days using 12 HP9000/735 workstations [1019]. That is the most effective attack against DES at the time of this writing.

Linear cryptanalysis is heavily dependent on the structure of the S-boxes and the S-boxes in DES are not optimized against this attack. In fact, the ordering of the S-boxes chosen for DES lies among the 9 percent to 16 percent that offer the least protection against linear cryptanalysis [1018]. According to Don Coppersmith [373,374], resistance to linear cryptanalysis "was not part of the design criteria of DES." Either they didn't know about linear cryptanalysis or they knew about something else even more powerful whose resistance criteria took precedence.

Linear cryptanalysis is newer than differential cryptanalysis, and there may be more performance improvements in the coming years. Some ideas are proposed in [1270,811], but it is not clear that they can be used effectively against full DES. They work very well against reduced round variants, however.

Future Directions

Some work has been done to try to extend the concept of differential cryptanalysis to higher-order differentials [702,161,927,858,860]. Lars Knudsen uses something called partial differentials to attack 6-round DES; it requires 32 chosen plaintexts and 20,000 encryptions [860]. It is still too new to know if these extensions will make it easier to attack full 16-round DES.

Another avenue of attack is **differential-linear cryptanalysis**: combining differential and linear cryptanalysis. Susan Langford and Hellman have an attack on 8-round DES that recovers 10 key bits with an 80 percent probability of success with 512 chosen plaintexts and a 95 percent probability of success with 768 chosen plaintexts [938]. After the attack, a brute-force search of the remaining keyspace (2^{46} possible keys) is required. While this attack is comparable in time to previous attacks, it requires far less plaintext. However, it doesn't seem to extend easily to more rounds.

But this attack is still new and work continues. It is possible that there may be a breakthrough some time during the next few years. Maybe there are benefits in combining this attack with higher-order differential cryptanalysis. Who knows?

Previous | Table of Contents | Next

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

12.5 The Real Design Criteria

After differential cryptanalysis became public, IBM published the design criteria for the S-boxes and the P-box [373,374]. The criteria for the S-boxes are:

— Each S-box has 6 input bits and 4 output bits. (This was the largest size that could be accommodated in a single chip with 1974 technology.)

— No output bit of an S-box should be too close to a linear function of the input bits.

— If you fix the left-most and right-most bits of an S-box and vary the 4 middle bits, each possible 4-bit output is attained exactly once.

— If two inputs to an S-box differ in exactly 1 bit, the outputs must differ in at least 2 bits.

— If two inputs to an S-box differ in the 2 middle bits exactly, the outputs must differ in at least 2 bits.

— If two inputs to an S-box differ in their first 2 bits and are identical in their last 2 bits, the two outputs must not be the same.

— For any nonzero 6-bit difference between inputs, no more than 8 of the 32 pairs of inputs exhibiting that difference may result in the same output difference.

— A criterion similar to the previous one, but for the case of three active S-boxes.

The criteria for the P-box are:

— The 4 output bits from each S-box in round i are distributed so that 2 of them affect the middle-bits of S-boxes at round$i$ + 1 and the other 2 affect end bits.

— The 4 output bits from each S-box affect six different S-boxes; no 2 affect the same S-box.

— If the output bit from one S-box affects a middle bit of another S-box, then an output bit from that other S-box cannot affect a middle bit of the first S-box.

The paper goes on to discuss the criteria. Generating S-boxes is pretty easy today, but was a complicated task in the early 1970s. Tuchman has been quoted as saying that they ran computer programs for months cooking up the S-boxes.

12.6 DES Variants

Multiple DES

Some DES implementations use triple-DES (see Figure 12.10) [55]. Since DES is not a group, then the resultant ciphertext is much harder to break using exhaustive search: 2^{112} attempts instead of 2^{56} attempts. See Section 15.2 for more details.

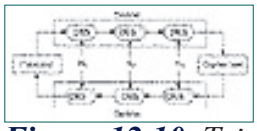


Figure 12.10 *Triple-DES.*

DES with Independent Subkeys

Another variation is to use a different subkey for each round, instead of generating them from a single 56-bit key [851]. Since 48 key bits are used in each of 16 rounds, this means that the key length for this variant is 768 bits. This variant would drastically increase the difficulty of a brute-force attack against the algorithm; that attack would have a complexity of 2^{768}.

However, a meet-in-the-middle attack (see Section 15.1) would be possible. This would reduce the complexity of attack to 2^{384}; still long enough for any conceivable security needs.

Although independent subkeys foil linear cryptanalysis, this variant is susceptible to differential cryptanalysis and can be broken with 2^{61} chosen plaintexts (see Table 12.15) [167,172]. It would seem that any modification of the key schedule cannot make DES much stronger.

DESX

DESX is a DES variant from RSA Data Security, Inc. that has been included in the MailSafe electronic mail security program since 1986 and the BSAFE toolkit since 1987. DESX uses a technique called whitening (see Section 15.6) to obscure the inputs and outputs to DES. In addition to a 56-bit DES key, DESX has an additional 64-bit whitening key. These 64 bits are XORed to the plaintext before the first round of DES. An additional 64 bits, computed as a one-way function of the entire 120-bit DES key, is XORed to the ciphertext

after the last round [155]. Whitening makes DESX much stronger than DES against a brute-force attack; the attack requires $(2^{120})/n$ operations with n known plaintexts. It also improves security against differential and linear cryptanalysis; the attacks require 2^{61} chosen plaintexts and 2^{60} known plaintexts, respectively [1338].

CRYPT(3)

CRYPT(3) is a DES variant found on UNIX systems. It is primarily used as a one-way function for passwords, but sometimes can also be used for encryption. The difference between CRYPT(3) and DES is that CRYPT(3) has a key-dependent expansion permutation with 2^{12} possible permutations. This was done primarily so that off-the-shelf DES chips could not be used to construct a hardware password-cracker.

Generalized DES

Generalized DES (GDES) was designed both to speed up DES and to strengthen the algorithm [1381,1382]. The overall block size increases while the amount of computation remains constant.

Figure 12.11 is a block diagram of GDES. GDES operates on variable-sized blocks of plaintext. Encryption blocks are divided up into q 32-bit sub-blocks; the exact number depends on the total block size (this was variable in the design, but must be fixed for each implementation). In general, q equals the block size divided by 32.

Function f is calculated once per round on the right-most block. The result is XORed with all the other parts, which are then rotated to the right. GDES has a variable number of rounds, n. There is a slight modification to the last round, so that the encryption and decryption processes differ only in the order of the subkeys (just like DES). In fact, if $q = 2$ and $n = 16$, this *is* DES.

Biham and Shamir [167,168] showed that, using differential cryptanalysis, GDES with $q = 8$ and $n = 16$ is breakable with only six chosen plaintexts. If independent subkeys are also used, 16 chosen plaintexts are required. GDES with $q = 8$ and $n = 22$ is breakable with 48 chosen plaintexts, and GDES with $q = 8$ and $n = 31$ requires only 500,000 chosen plaintexts to break. Even GDES with $q = 8$ and $n = 64$ is weaker than DES; 2^{49} chosen plaintexts are required to break it. In fact, any GDES scheme that is faster than DES is also less secure (see Table 12.15).

A variant of this scheme recently appeared [1591]. It is probably no more secure than the original GDES. In general, any large block DES variant that is faster than DES is probably also less secure than DES.

DES with Alternate S-Boxes

Other DES modifications centered around the S-boxes. Some designs made the order of the S-boxes variable. Other designers varied the contents of the S-boxes themselves. Biham and Shamir showed [170,172] that the design of the S-boxes, and even the order of the S-boxes themselves, were optimized

against differential cryptanalysis:

> The replacement of the order of the eight DES S-boxes (without changing their value) also makes DES much weaker: DES with 16 rounds of a particular replaced order is breakable in about 2^{38} steps.... DES with random S-boxes is shown to be very easy to break. Even a minimal change of one entry of one of the DES S-boxes can make DES easier to break.

The DES S-boxes were not optimized against linear cryptanalysis. There are better S-boxes than the ones that come with DES, but blindly choosing new S-boxes isn't a good idea.

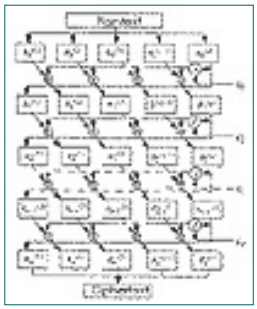


Figure 12.11 *GDES.*

Table 12.15 [167,169] lists some modifications to DES and the number of chosen plaintexts required for differential cryptanalysis. One change not listed, combining the left and right halves using addition mod 2^4 instead of XOR, is 2^{17} times harder to break than DES [689].

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

RDES

RDES is a variant that replaces swapping the left and right halves at the end of each round with a key-dependent swap [893]. The swappings are fixed, depending solely on the key. This means that the 15 key-dependent swaps occur with 2^{15} possible instances, and that the variant is not resistant to differential cryptanalysis [816,894,112]. RDES has a large number of weak keys. In fact, almost every key is weaker than a typical DES key. This variant should not be used.

A better idea is to swap only within the right half, at the beginning of each round. Another better idea is to make the swapping dependent on the input data and not a static function of the key. There are a number of possible variants [813,815]. In RDES-1, there is a data-dependent swap of the 16-bit words at the beginning of each round. In RDES-2, there is a data-dependent swap of the bytes at the beginning of each round after the 16-bit swappings as in RDES-1. And so on through RDES-4. RDES-1 is secure against both differential cryptanalysis [815] and linear cryptanalysis [1136]. Presumably RDES-2 and greater are as well.

Table 12.15
Differential Cryptanalysis Attacks against DES Variants

Modified Operation	Chosen Plaintexts
Full DES (no modification)	2^{47}
P permutation	Cannot strengthen
Identity permutation	2^{19}
Order of S-boxes	2^{38}

Replace XORs by additions	2^{39}, 2^{31}
S-boxes:	
Random	2^{18}–2^{20}
Random permutations	2^{33}–2^{41}
One entry	2^{33}
Uniform tables	2^{26}
Elimination of the E Expansion	2^{26}
Order of E and subkey XOR	2^{44}
GDES (width q = 8):	
16 rounds	6, 16
64 rounds	2^{49} (independent key)

s^n DES

A group of Korean researchers, led by Kwangjo Kim, has attempted to find a set of S-boxes that are optimally secure against both linear and differential cryptanalysis. Their first attempt, known as s^2DES, was presented in [834] and shown to be worse than DES against differential cryptanalysis in [855,858]. Their next attempt, s^3DES, was presented in [839] and shown to be worse than DES against linear cryptanalysis [856,1491,1527,858,838]. Biham suggested a minor change to make s^3DES secure against both linear and differential cryptanalysis [165]. The group went back to their computers and developed better techniques for S-box design [835,837]. They proposed s^4DES [836] and then s^5DES [838,944].

Table 12.16 gives the s^3DES S-boxes with S-box 1 and S-box 2 reversed, which are secure against both differential and linear cryptanalysis. Sticking this variant in a triple-DES mix is sure to irritate cryptanalysts.

DES with Key-Dependent S-Boxes

Linear and differential cryptanalysis work only if the analyst knows the composition of the S-boxes. If the S-boxes are key-dependent and chosen by a cryptographically strong method, then linear and differential cryptanalysis are much more difficult. Remember though, that randomly generated S-boxes have very poor differential and linear characteristics; even if they are secret.

Table 12.16
s^3DES S-Boxes (with S-box 1 and S-box 2 reversed)

S-box 1:															
13	14	0	3	10	4	7	9	11	8	12	6	1	15	2	5
8	2	11	13	4	1	14	7	5	15	0	3	10	6	9	12
14	9	3	10	0	7	13	4	8	5	6	15	11	12	1	2
1	4	14	7	11	13	8	2	6	3	5	10	12	0	15	9
S-box 2:															

15	8	3	14	4	2	9	5	0	11	10	1	13	7	6	12
6	15	9	5	3	12	10	0	13	8	4	11	14	2	1	7
9	14	5	8	2	4	15	3	10	7	6	13	1	11	12	0
10	5	3	15	12	9	0	6	1	2	8	4	11	14	7	13
S-box 3:															
13	3	11	5	14	8	0	6	4	15	1	12	7	2	10	9
4	13	1	8	7	2	14	11	15	10	12	3	9	5	0	6
6	5	8	11	13	14	3	0	9	2	4	1	10	7	15	12
1	11	7	2	8	13	4	14	6	12	10	15	3	0	9	5
S-box 4:															
9	0	7	11	12	5	10	6	15	3	1	14	2	8	4	13
5	10	12	6	0	15	3	9	8	13	11	1	7	2	14	4
10	7	9	12	5	0	6	11	3	14	4	2	8	13	15	1
3	9	15	0	6	10	5	12	14	2	1	7	13	4	8	11
S-box 5:															
5	15	9	10	0	3	14	4	2	12	7	1	13	6	8	11
6	9	3	15	5	12	0	10	8	7	13	4	2	11	14	1
15	0	10	9	3	5	4	14	8	11	1	7	6	12	13	2
12	5	0	6	15	10	9	3	7	2	14	11	8	1	4	13
S-box 6:															
4	3	7	10	9	0	14	13	15	5	12	6	2	11	1	8
14	13	11	4	2	7	1	8	9	10	5	3	15	0	12	6
13	0	10	9	4	3	7	14	1	15	6	12	8	5	11	2
1	7	4	14	11	8	13	2	10	12	3	5	6	15	0	9
S-box 7:															
4	10	15	12	2	9	1	6	11	5	0	3	7	14	13	8
10	15	6	0	5	3	12	9	1	8	11	13	14	4	7	2
2	12	9	6	15	10	4	1	5	11	3	0	8	7	14	13
12	6	3	9	0	5	10	15	2	13	4	14	7	11	1	8
S-box 8:															
13	10	0	7	3	9	14	4	2	15	12	1	5	6	11	8
2	7	13	1	4	14	11	8	15	12	6	10	9	5	0	3
4	13	14	0	9	3	7	10	1	8	2	11	15	5	12	6
8	11	7	14	2	4	13	1	6	5	9	0	12	15	3	10

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

Applied Cryptography, Second Edition: Protocols, Algorithms, and Source Code in C (cloth)
(Publisher: John Wiley & Sons, Inc.)
Author(s): Bruce Schneier
ISBN: 0471128457
Publication Date: 01/01/96

Here is a method to use 48 additional key bits to generate S-boxes that are resistant to both linear and differential cryptanalysis [165].

(1) Rearrange the DES S-boxes: 24673158.

(2) Select 16 of the remaining key bits. If the first bit is 1, swap the first two rows of S-box 1 with the last two rows of S-box 1. If the second bit is a 1, swap the first eight columns of S-box 1 with the second eight columns of S-box 1. Do the same to S-box 2 with the third and fourth key bits. Do the same with S-boxes 3 through 8.

(3) Take the remaining 32 key bits. XOR the first four with every entry of S-box 1, the second four with every entry of S-box 2 and so on.

The complexity of a differential cryptanalysis attack against this system is 2^{51}; the complexity of a linear cryptanalysis attack is 2^{53}. The complexity of exhaustive search is 2^{102}.

What is neat about this DES variant is that it can be implemented in existing hardware. Several DES chip vendors sell DES chips with loadable S-boxes. This S-box generation method can be done outside the chip and then loaded in. Differential and linear cryptanalysis require so much known or chosen plaintext as to be unworkable, and a brute-force attack is inconceivable—with no speed penalties.

12.7 How Secure Is DES Today?

The answer is both easy and hard. The easy answer just looks at key length (see Section 7.1). A brute-force DES-cracking machine that can find a key in an average of 3.5 hours cost only $1 million in 1993 [1597,1598]. DES is so widespread that it is na•ve to pretend that the NSA and its counterparts haven't built such a machine. And remember that cost will drop by a factor of 5 every 10 years. DES will only become less secure as time goes on.

The hard answer tries to estimate cryptanalytic techniques. Differential cryptanalysis was known by the NSA long before the mid-1970s, when DES first became a standard. It is na•ve to pretend that the NSA theoreticians have been idle since then; almost certainly they have developed newer cryptanalytic techniques that can be applied against DES. But there are no facts, only rumors.

Winn Schwartau writes that the NSA had built a massively parallel DES-cracking machine as early as the mid-1980s [1404]. At least one such machine was built by Harris Corp. with a Cray Y-MP as a front end. Supposedly there are a series of algorithms that can reduce the complexity of a DES brute-force search by several orders of magnitude. Contextual algorithms, based on the inner workings of DES, can scrap sets of possible keys based on partial solutions. Statistical algorithms reduce the effective key size even further. And other algorithms choose likely keys—words, printable ASCII, and so on (see Section 8.1)—to test. The rumor is that the NSA can crack DES in 3 to 15 minutes, depending on how much preprocessing they can do. And these machines cost only $50,000 each, in quantity.

A different rumor is that if the NSA has a large amount of plaintext and ciphertext, its experts can perform some kind of statistical calculation and then go out to an array of optical disks and retrieve the key.

These are just rumors, but they don't give me a warm, fuzzy feeling about DES. It has just been too big a target for too long. Almost any change to DES will be more annoying; maybe the resultant cipher will be easier to break, but the NSA might not have the resources to devote to the problem.

My recommendation is to use Biham's construction for key-dependent S-boxes. It is easy to implement in software and in hardware chips that have loadable S-boxes, and has no performance penalty over DES. It increases the algorithm's resistance to a brute-force attack, makes differential and linear cryptanalysis harder, and gives the NSA something at least as strong as DES—but different—to worry about.

Previous | Table of Contents | Next

Products | **Contact Us** | **About Us** | **Privacy** | **Ad Info** | **Home**

Use of this site is subject to certain Terms & Conditions, Copyright © 1996-2000 EarthWeb Inc.
All rights reserved. Reproduction whole or in part in any form or medium without express written permission of EarthWeb is prohibited. Read EarthWeb's privacy statement.

EXHIBIT F

US 20130276080A1

(19) **United States**

(12) **Patent Application Publication**

Nakonetzki et al.

(10) **Pub. No.: US 2013/0276080 A1**

(43) **Pub. Date: Oct. 17, 2013**

(54) **METHOD OF AUTHENTICATING A USER AT A SERVICE ON A SERVICE SERVER, APPLICATION AND SYSTEM**

(71) Applicant: **VODAFONE HOLDING GMBH**, Dusseldorf (DE)

(72) Inventors: **André Nakonetzki**, Erkrath (DE); **Wilhelm Joubert**, Dusseldorf (DE)

(21) Appl. No.: **13/860,670**

(22) Filed: **Apr. 11, 2013**

(30) **Foreign Application Priority Data**

Apr. 11, 2012 (DE) 10 2012 103 106.1

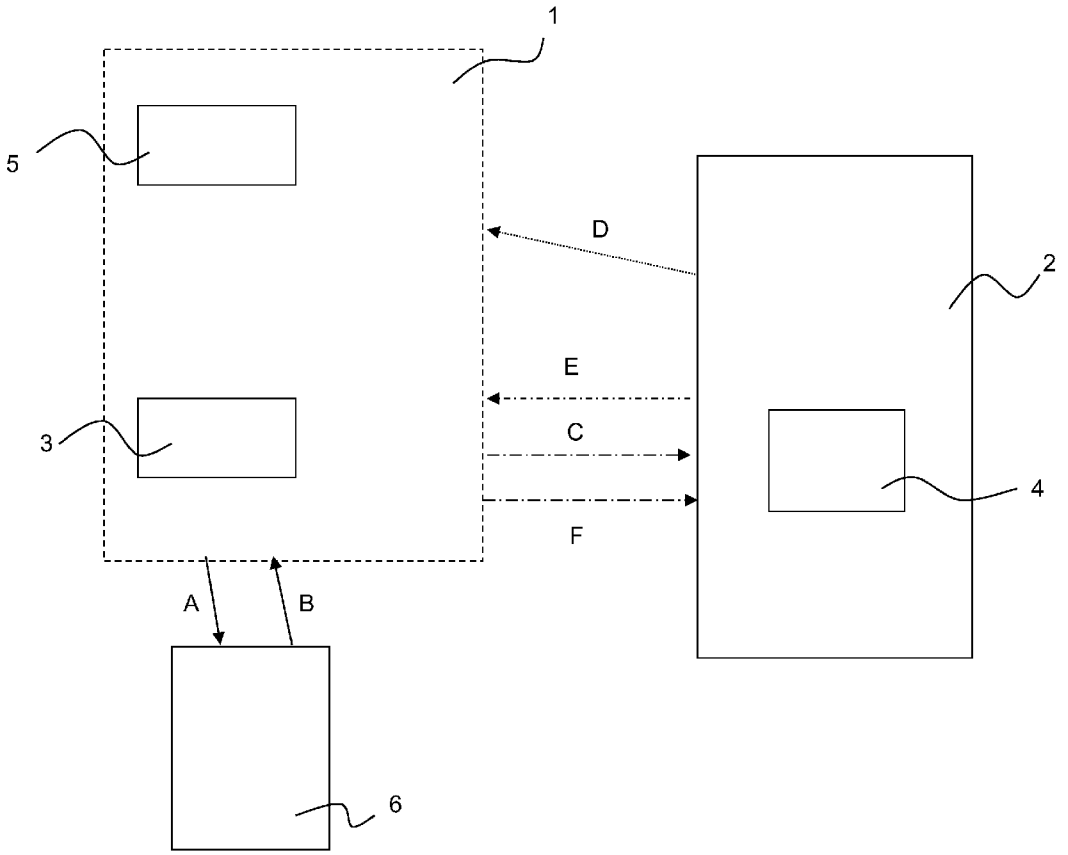
Publication Classification

(51) **Int. Cl.**
H04L 29/06 (2006.01)

(52) **U.S. Cl.**
CPC *H04L 63/08* (2013.01)
USPC **726/7**

(57) **ABSTRACT**

The present invention relates to a method of authenticating a user of a communication terminal (**1**), on which at least one application (**3**) is installed, at a service (**4**) on a service server (**2**). The method is characterized in that an initiation request, which contains an address identifier of the communication terminal (**1**), is generated by the application (**3**) for initiating a session and is transmitted to the service server (**2**), the service server (**2**) after receipt of the initiation request automatically generates and transmits a request specific key (OTP) to the communication terminal (**1**) and
the application (**3**) subsequently generates an access request in the session, wherein
the application (**3**) integrates the received request specific key (OTP) automatically in the access request, and
the access request is automatically transmitted from the communication terminal (**1**) to the service server (**2**) in the session.
Furthermore a respective application (**3**) and a respective system are described.



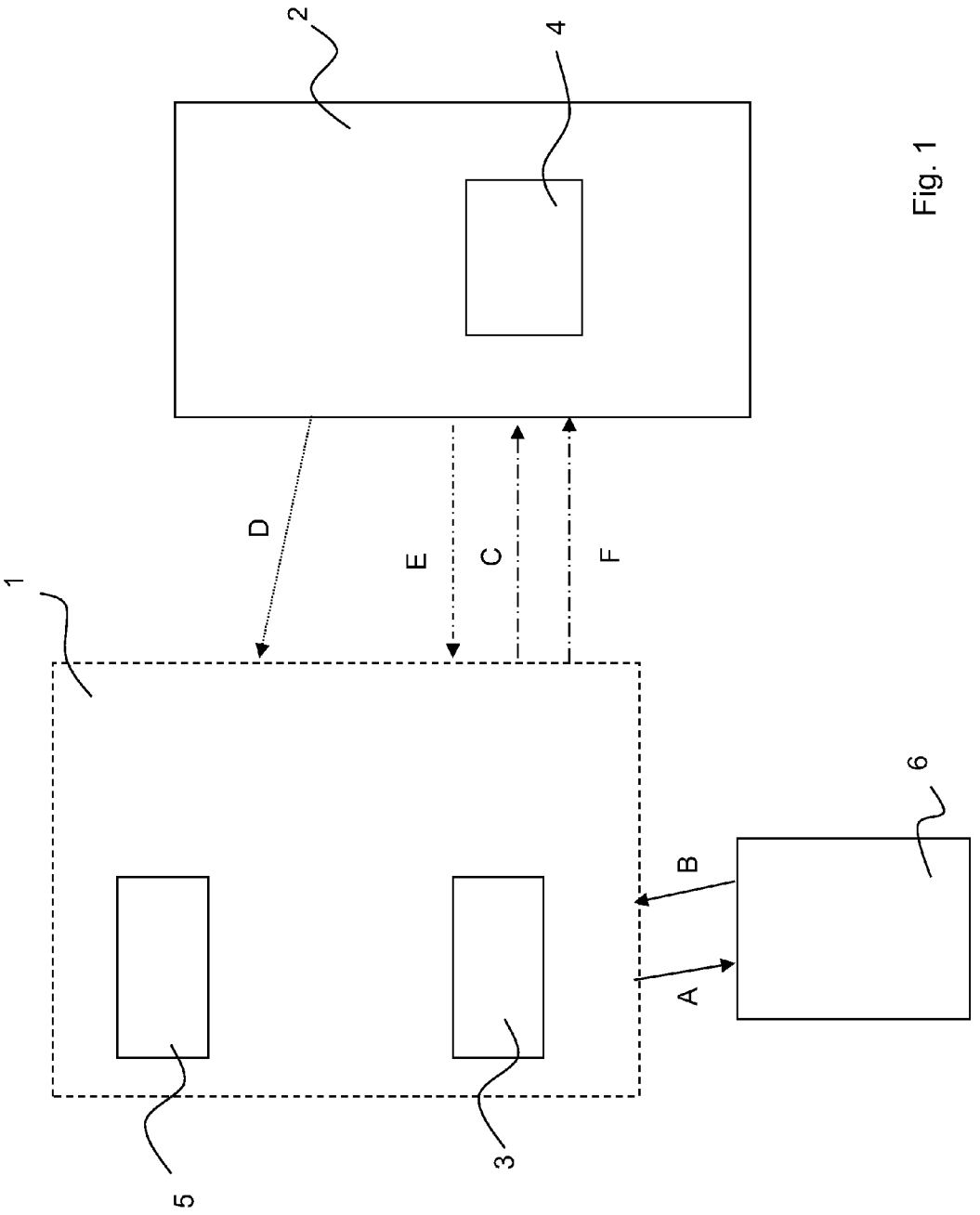


Fig. 1

METHOD OF AUTHENTICATING A USER AT A SERVICE ON A SERVICE SERVER, APPLICATION AND SYSTEM

REFERENCE TO PENDING PRIOR PATENT APPLICATION

[0001] This patent application claims benefit of German Patent Application No. 10 2012 103 106.1, filed Apr. 11, 2012, which patent application is hereby incorporated herein by reference.

FIELD OF THE INVENTION

[0002] The present invention relates to a method of authenticating a user as well as to a corresponding application and a corresponding system.

BACKGROUND OF THE INVENTION

[0003] In communication networks, as for example the Internet, it is well known, that users have to register themselves with a service for using this service. For this purpose, for example, a user name and a password are entered. This data is entered by the user at his terminal and are transmitted to the service. Furthermore, it is known to generate a password for authentication of a mobile network unit of a user at a wireless local network (WLAN). This password is then provided to the user at the mobile device which is associated to him. The user can then enter this password at the mobile network unit, in order to authenticate himself at the WLAN-Network and to use the WLAN Network. Such a method is, for example, described in EP 1 351 538 A1.

[0004] One disadvantage of the known methods is that they require a high degree of interaction on the part of the user. In addition, a previous registration of the user at the service is required, at least for the method with user name and password.

[0005] The problem underlying the present invention is thus to provide a solution, which allows for a simple yet secure usage of services, in particular on the internet, also from a mobile communication terminal.

SUMMARY OF THE INVENTION

[0006] The invention is based on the finding, that this problem can be solved by automatically recognizing a participant and being able to subsequently fully automatically authenticate the participant for a service.

[0007] According to a first aspect, the problem according to the invention is thus solved by a method of authenticating a user of a communication terminal, on which at least one application is installed, at a service on a service server. The method is characterized in that an initiation request, which contains an address identifier of the communication terminal, is generated by the application for initiating a session and is transmitted to the service server,

[0008] after receipt of the initiation request the service server automatically generates and transmits a request specific key to the communication terminal and

[0009] the application subsequently generates an access request in the session, wherein the application integrates the received request specific key automatically in the access request, and

[0010] the access request is automatically transmitted from the communication terminal to the service server in the session

[0011] Authenticating a user according the present invention is in particular understood as the authenticating of a user at a service on a service server, wherein the communication between the communication terminal, which the user uses for the communication with the service server, and the service server is authenticated by authentication of the user. Hence, authenticating a user according to the invention refers to authenticating or admitting the communication of the communication terminal, which is associated to the user, with the service server and in particular the communication of the application on the communication terminal with the service on the service server. The authentication is a confirmation of the entitlement or admittance for the communication and in particular for a session. The communication between the communication terminal and the service server and in particular between the application and the service is also referred to as data transfer or data exchange between these components or programs. This communication, which is authenticated, that means admitted, by the authentication of the user is hereinafter also referred to as session. In a session herein several accesses may be performed to a service at the service server. Access denotes a data transmission originating from the communication terminal, which can comprise data exchange, that means a transmission of data from the service server to the communication terminal. The session is preferably limited with respect to time. The time limitation can be bound to a mere time indication, that means the session may, for example, be terminated after a specific time period. According to the invention it is, however, preferred, that the session is terminated by one of the two components or programs, that means the communication terminal, the service server, the application or the service. The session is carried out via a communication connection between the communication terminal and the service server. This communication is preferably at least partially performed over a packet switched network, in particular the internet. The session is carried out in line with the protocol, which is preset for the respective network. For the Internet it may be, for example, the http-protocol. The session or communication according to the invention can be carried out via an access network and a core network. Herein, the service server is arranged in the core network. The core network may, for example, be the Internet and the access network may be a local or mobile network, for example a LAN, WLAN, GSM, UMTS or the like.

[0012] As far as reference is made to the service server, the steps which are carried out by the service server—if not indicated differently—are preferably carried out by the service, which runs on the service server and to which the application is connected.

[0013] A communication terminal according to the invention is preferably understood as a communication terminal, which can be used as a terminal or end device of a mobile network. The mobile network is also referred to as mobile communication network. Preferably, the communication terminal is thus a mobile phone. A communication terminal may be, however, also a different communication device, such as a PDA, Laptop or notebook, wherein this communication device is then specifically equipped for the communication via a mobile network. For example, the communication device has a corresponding or respective adapter or a corresponding or respective modem. In addition, the communication terminal, which may also be referred to as a mobile communication terminal, is preferably designed for the communication with units of the Internet. In particular, a commu-

nication according to the http-protocol can be performed with the communication terminal according to the invention. It is specifically preferred that also the communication according to the https-protocol is possible. In a specifically preferred embodiment, the communication terminal is thus a smartphone. Being designed for a purpose according to the invention can also be referred to as being adapted for that purpose.

[0014] An application, which is installed on the communication terminal, denotes a program which is hereinafter also referred to as App, application or client. This application according to the invention is in particular designed for the access to and the usage of a service on the service server. In that respect, the application is a client to the service on the service server. The service, at least partially, runs on the service server.

[0015] According to the invention an initiation request for initiating a session is generated by the application and is transmitted to the service server, in particular to the service. Session denotes—as already indicated above—a communication, which relates to a service, which is installed on the service server or runs thereon, respectively. One or several accesses to the service may occur in the session. In addition, the session runs via the communication connection, which exists between the communication terminal and the service server. Via this existing communication connection, the initiation request is transmitted to the service server. The initiation request is received at the service server and there effects the opening of a session. In the initiation request an address identifier of the communication terminal is contained. Thereby, the communication and thus the session are fixed to the communication terminal. The address identifier thus identifies at least the communication terminal.

[0016] After receipt of the initiation request, the service server automatically generates a request specific key. A request specific key denotes a key, which relates to the initiation request, that means which is associated to the session, which is to be initiated by the initiation request. The association of the key to the session is preferably stored at the service server. A key according to the invention denotes an indication which serves for the authentication of a user. Preferably, the key is used together with further information during the authentication. The key is hereinafter also referred to as password and is, for example, a code word or a key word, which consist of data, that means digits, letters and/or other characters. Since the request specific key is generated at the service server upon receipt of each initiation request, it may also be referred to as one-time-password, as it is only valid for the session to which the request relates. An automatic generation of the request specific key according to the invention is understood as an establishing, which is performed without further entries, in particular without further user input at the service server.

[0017] The request specific key is transmitted from the service server to the communication terminal. Thereby, the request specific key is available for further actions at the communication terminal, from which the initiation request had been generated and transmitted. The transmission of the request specific key to the communication terminal and the receipt of the request specific key are performed automatically. This means that no interaction by the user is necessary.

[0018] The application subsequently generates an access request in the session at the communication terminal. The generation in the session in this context means, that the access request is generated during the session, which has been started by the initiation request. The access request may be generated by the application itself or may be triggered by an action of the user. This action of the user may, for example, be the repeated or renewed selection of the service, for example, by a link on the communication terminal. The application automatically integrates the received request specific key into the access request. For automatic integration, the application may read the request specific key, for example, from a message, which has been received at the communication terminal. Alternatively, the request specific key may be, however, also transmitted automatically to the application by the unit in the communication terminal, which receives the request specific key. As the application, which receives the request specific key is also the application from which the initiation request for initiation of the session has been generated and transmitted, this application can associate the request specific key, which relates to this request, to the session and integrate the request specific key in the corresponding access request. In addition, the access request is automatically transmitted from the communication terminal to the service server in the session.

[0019] With the method according to the invention it is possible to authenticate the user at a service server, without an interaction of the user being absolutely mandatory. In particular, entering a key at the communication terminal by the user is not required. As the key according to the invention is specific to the initiation request, by which a session is established, the key, which is later received at the service server, can unambiguously be associated or assigned to the session.

[0020] The request specific key is transmitted to the communication terminal, from which the initiation request had been sent on the basis of the transmitted address identifier of the communication terminal. Thereby, the security of the authentication is increased, as the access of third parties can be prevented to the largest extend. Also the communication in the communication terminal for transmission of the request specific key from a receiving unit to the application is more secure in comparison to external transmissions from a third device to the communication terminal, as the access of third parties to such an internal transmission is excluded. As, in addition, the request specific key is included into the access request automatically, the security is further increased and the user interaction is also minimized.

[0021] According to the invention, an address identifier of the communication terminal is integrated into the initiation request at the communication terminal. With this address identifier on the one hand an additional communication connection from the service server with the communication terminal can be established. The address identifier is preferably an identifier of the communication terminal in a communication network, in particular in a mobile network. Thereby, a communication connection can be established from the service server to the communication terminal by means of the address identifier via the mobile network. Preferably, the address identifier of the communication terminal also serves for identifying the user at the service at the service server. For this purpose, preferably an address identifier, which in addition to the address of the communication terminal, in particular in a mobile network, also establishes a reference to the user, is used as address identifier. In a mobile network, this address identifier is in particular the Mobile Subscriber Integrated Services Digital Network Number (MSISDN). With this number on the one hand the communication terminal is identifiable and addressable in the communication network,

in particular in the mobile network. On the other hand it becomes possible to identify the user of the communication terminal with the address identifier of the communication terminal. This is in particular true, as in a mobile network for each MSISDN Number in general also the corresponding user is stored. Since this address identifier is contained in the initiation request to the service server, this information is available at the service server for further usage. Since the address identifier, in addition, is integrated in the initiation request and a request specific key is generated for this initiation request, also a relation between the request specific key and the communication terminal and preferably also the user exists.

[0022] Preferably, the application performs a query of an address identifier of the communication terminal prior to generating the initiation request. Thereby it can be ensure, even with a terminal, on which this address identifier might not be present, that this information is available prior to establishing or generating the initiation request. The address identifier, which is queried in this query, is preferably the current, that means the present, address identifier of the communication terminal. If, for example, the MSISDN is used as address identifier, it may change with different users of the communication terminal. Therefore, a query prior to establishing the initiation request is helpful, in order to ensure a reliable authentication.

[0023] The query of the address identifier is preferably performed internally in the communication terminal. In an internal query, the application will, for example, access storage units, which are exclusively used and written to by the application. Hence, in such a query entries of an address identifier, which originate from previous sessions of the application, would be read out and could be used. In addition or alternatively, it is, however, also possible, that the application in this query accesses other storage units in the communication terminal, for example a SIM-card. Only if this query delivers a negative query result, the query is subsequently performed externally to the communication terminal. An external query denotes a query, via which the information is queried from a unit, which is separate to the communication terminal and which is preferably a network side unit, in particular a network server or a network register. The network server may, for example, be assigned or associated to the network operator or network provider and may comprise the Home Location Register. Since the query for the address identifier is performed internally first and only in case of a negative query result is performed externally, the authentication process if further simplified. In particular, with an initiation request which is started from a communication terminal, from which previously already an initiation request had been started, a repeated or renewed external query of the address identifier can be omitted and the method can thus be simplified altogether.

[0024] It is particularly preferable, that the external query of the address identifier comprises the usage of a short dial. Herein a query short message, for example an empty short message (SMS), can be sent from the communication terminal to a server of the network provider. The address identifier, in particular the MSISDN of the sender, that means of the communication terminal, will be copied and will be inserted as message text into the response to this query message at the short dial application. This response message is received at the communication terminal. From this response message the address identifier can be read out at the communication terminal and preferably immediately at the application, and can be used for generating the initiation request. Furthermore, the received address identifier can also be stored for further or later usages.

[0025] Preferably, in addition to the request specific key at least one request specific identifier of the session is generated at the service server. Since the session is only started by the initiation request, also the request specific identifier is only generated after the receipt of the initiation request at the service server. The request specific identifier is also referred to as session token and serves for identifying the session. With the request specific identifier in particular a communication between the communication terminal and the service and in particular between the application and the service can be identified as belonging to a session and can be processed accordingly. If the communication between the communication terminal and the service server is performed via the Internet, the session may also be referred to as an http-session. For this session a unique request specific identifier, which can also be referred to as a session ID, is associated to the application, that means the client. The application sends this request specific identifier at each subsequent access to the service. The session is preferably generated with the first access by the application and remains valid for the entire communication duration. That means, that the access, the authentication, the data exchange as well as the termination of the session are contained in the session.

[0026] Preferably, the request specific identifier of the session is a temporary identifier. In particular, the request specific identifier is limited to the duration of the session, that means is valid only for the duration of the session.

[0027] In the method according to the invention, preferably the request specific identifier is transmitted from the service server to the communication terminal via the communication connection, via which the initiation request had been transmitted to the service server. This communication connection is preferably a communication connection, which at least partially is carried out according to the protocol of the Internet, preferably the http-protocol or https-protocol. Since this connection has already been established prior to the transmission of the initiation request, the establishment of a further communication connection is not necessary. In addition, the identifier may thereby be transmitted in the session, which has been opened by the initiation request and can thus easily be associated to the session at the service server and at the communication terminal.

[0028] The request specific identifier is integrated by the application into the access request at the communication terminal. Thereby, both the request specific key as well as the request specific identifier can be integrated into the access request and thus all information which is necessary for the authentication can automatically be provided by the communication terminal at the service server. Furthermore, in an access to the service, which occurs after the authentication of the user, the integration of the request specific key may be omitted and only the request specific identifier may be transmitted to the service, since at that point in time only the association to a session is necessary.

[0029] The request specific key and the request specific identifier are preferably transmitted from the service server to the communication terminal via different communication connections, which can also be referred to as communication channels. Thereby, the security of the authentication can be further increased. This is in particular true, since two connec-

tions have to be established for the communication. One connection is preferably the connection, via which the communication terminal has transmitted the initiation request to the service server and the other communication connection is generated using the address identifier, which the service server has received from the communication terminal.

[0030] According to a preferred embodiment, the request specific key is transmitted in a short message, in particular via a mobile network. It is particularly preferred that the request specific key is transmitted to the communication terminal in an SMS. in that case it is particularly preferred, that the application on the communication terminal can read the request specific key from the SMS. Alternatively, it is also possible, that the request specific key is read out from the SMS by a different unit and is subsequently provided to the application. One advantage of transmitting the request specific key via SMS is that the communication via SMS is already secured due to the applicable protocols and infrastructure, for example of the GSM network. With the transmission of the request specific key in an SMS it can be received at the SIM-card of the communication terminal, which may be a mobile phone, or it can reach the storage of the mobile phone. The application can access the SIM-card or the storage. This can, for example, be capsulated by the respective smartphone SDK-API, for example Android SDK.

[0031] The request specific identifier is at least partially transmitted via a packet switching network, in particular via the Internet. Herein, the request specific identifier may, for example, be transmitted in a cookie. Thereby, the request specific identifier is available at the time of a later access by the user to the service and can be delivered with the access to the service server. The application preferably reads the request specific key out of at least one message in the communication terminal, for obtaining the request specific key. The message preferably is a short message, in particular an SMS message. Since the reading out is performed by the application, an interaction from the user is not required.

[0032] It is, however, also provided, that the user of the communication terminal has access to the SMS storage of the device and in case of a proprietary smartphone system, for example Apple iPhone, this SMS message is read by the user and the user can enter the request specific key by manual input into a text field, which is provided by the application. Thereby the reading out by the application would be replaced by a manual step.

[0033] The service on the service server is preferably a value added service, which may also be referred to as premium service. Such service is thus not the mere establishment of the communication connection. Any service, such as for example social network services on the internet and the like may be used.

[0034] According to one embodiment, a communication connection is established between the communication terminal and the service server before or during the transmission of the initiation request from the communication terminal to the service server and the initiation request is transmitted via this communication connection. This means, that the authentication according the present invention is not an authentication of the communication connection as such but relates to a communication or session, which is performed in an existing communication connection.

[0035] Preferably, in addition to the steps which have already been mentioned the method comprises the step of verifying the request specific key and the request specific identifier and the link of the key and the identifier on the service server and in case of positive verification result, authenticating, that means confirming the entitlement or admission of the user for the communication between the application and the service or the authentication of the session.

[0036] According to one embodiment, the method comprises the following steps:

[0037] The step of determining the identity of the user,

[0038] The step of session initiation,

[0039] The step of authentication, and

[0040] The step of verifying the request specific key and the request specific identifier and the link of the key and the identifier on the service server and confirmation of the authentication of the user in case of a positive verification result.

[0041] The determining of the identity of the user can in particular be performed by determining the address identifier (MSISDN) of the communication terminal in a mobile network. This address identifier can de determined, as already described above, by means of an internal or an external query and may, if necessary, be stored or buffered in the communication terminal.

[0042] In this embodiment, the session initiation is preferably carried out by sending an initiation request according to the Internet protocol, in particular as an https-request, which contains the identity of the user, in particular by means of the MSISDN, to the service server. Furthermore, the session initiation comprises the generating of a request specific key, in particular a one-time-password (OTP), at the service server. This request specific key is sent via the mobile network to the communication terminal, which is assigned to the address identifier. Finally, the step of the session initiation in this embodiment comprises the generation of a temporary request specific identifier, in particular a session token, and transmitting the session token by means of a cookie to the application on the communication terminal.

[0043] In this embodiment, the step of authenticating, in particular, comprises the generating of an access request, in which the request specific key, in particular the OTP, and the request specific identifier, in particular the session token, are contained, and the transmission of the access request to the service on the service server.

[0044] Finally, the method according to this embodiment comprises the step of verifying the request specific key, in particular the OTP, and the request specific identifier, in particular the session token, and the link of the key and the identifier on the service server and in case of a positive verification result, confirmation of the authentication of the user.

[0045] With the present invention and in particular with this embodiment of the method, a fully automated authentication of a session after automated identification of the user can be performed. The application, which is used therein, preferably is a smartphone app and the session is an http-session. In the method according to the invention, the user does not have to enter a code or password; the authentication is instead carried out autarkic.

[0046] According to a further aspect, the invention relates to an application for a communication terminal for the access to at least one service on a service server. The application is characterized in that the user can be authenticated at the service by the application, that the application is designed for at least generating and transmitting an initiation request for initiation a session, for automatically obtaining a request

specific key and for generating an access request, within which the application integrates the request specific key

[0047] The application according to the invention can be implemented by hardware and/or software.

[0048] The application may also be referred to as a computer program product, which can be stored on and run on the communication terminal, for example a mobile phone, in particular a smart phone. The communication terminal with respect to the application functions as a processing unit and in this context can also be referred to as a computer. The invention thus also relates to a computer program product, with program medium which can be read on a computer, in particular the communication terminal, which, if the program is loaded, has program means for performing the method according to the invention. Also a program, which comprises commands, which are adapted to perform the method according to the invention, is subject of the present invention. Also a computer readable medium, on which a program is stored, wherein the program triggers the computer, that means the communication terminal, to perform the method according to the invention is also subject of the present invention.

[0049] With the application of the invention, or the computer program product, respectively, it is advantageous, that due to the improved method, the authentication of a user can be carried out in a simple way. in particular, the interaction of the user can be reduced to a minimum. An entry of the request specific key at the communication terminal is not required with the application according to the invention.

[0050] It is particularly preferred that the application is designed such that it is designed for querying the address identifier of the communication terminal, as already explained above. Thereby also the recognition of the user, that means the identification of the user can be performed automatically.

[0051] According to a preferred embodiment, the application is designed for carrying out the invention according to the invention.

[0052] According to a further aspect, the invention relates to a system for authenticating a user of a communication terminal, wherein the system has at least one communication terminal and at least one service server with a service which is at least partially run on the service server. The system is characterized in that at least one application for communication with the service on the service server is present on the communication terminal, that the application is designed for automatically establishing an initiation request, which comprises an address identifier of the communication terminal, that the service server is designed for generating at least one request specific key on the basis of the initiation request and for transmitting the at least on request specific key via different communication connections to the communication terminal, that the communication terminal is designed for automatically generating and transmitting an access request, within which the received request specific key is contained, to the service server and that the service server is designed for comparing the request specific key, which is contained in the access request with the generated request specific key and to authenticate the user in case of a positive comparison result. This authentication of the user is thus a confirmation of the authorization of the user for communication of the communication terminal, which is used by the user, with the service server and in particular of the session between the service and the application.

[0053] Preferably, the system comprises at least one short dial application for communication of at least the address identifier of the communication terminal to the communication terminal. Herein, the short dial application is in particular designed for receiving and processing of short messages, in particular empty SMS messages. The short dial application can, for example, be stored on a server of the provider of the mobile network within which the communication terminal serves as an end device or in a unit, which the short dial application can access. The short dial application, according to the invention serves for copying the address identifier of the sender of the short message and for inserting the address identifier this into a response message to the sender. Thereby, the communication terminal can automatically query its address identifier.

[0054] The system according to the invention is preferably designed for performing the method according to the invention and preferably comprises at least one application according to the invention which is installed on at least one communication terminal.

[0055] Advantages and features, which have been described with respect to the method, correspondingly apply—as far as applicable—to the application according to the invention and the system according to the invention and vice versa and are thus only described once.

BRIEF DESCRIPTION OF THE DRAWING

[0056] The invention will now be described again with reference to the enclosed FIG. **1**, which shows an embodiment of the system according to the invention.

DETAILED DESCRIPTION OF THE INVENTION

[0057] In FIG. **1** a communication terminal **1** is embodied as a mobile phone. In particular, the communication terminal is a smartphone. Furthermore, a service server **2** is shown, with which the communication terminal **1** can communicate. On the service server **2** at least one service **4** is provided, that means is at least partially performed by the service server **2**. Furthermore, a short dial application **6** is shown. The communication terminal **1** can communicate with this short dial application **8**. In the communication terminal **1**, an identification module **5**, preferably as a Subscriber Identity Module (SIM), is contained and preferably inserted into the communication terminal **1**. Furthermore, an application **3** is shown in the communication terminal **1**, which is installed thereon and which can be run by the communication terminal **1**. The application **3** is designed such, that it can communicate with the service **4**. The service **4** may be, for example, a web application, WAP application or a different online application. If the application **3** is opened at the communication terminal **1**, a connection to the service server **2** is established automatically from the communication terminal **1**. Herein, for example, a browser is opened, which allows for access to the Internet, and a specific address, for example a URL, which is stored in the application **3**, is dialed or contacted. Thereby, a communication connection is established between the communication terminal **1** and the service server **2**, which may for example be a WiFi or WLAN connection, respectively.

[0058] As indicated in FIG. **1**, a query or validation may be carried out within the communication terminal **1** the first time that the application **3** is opened. In this query an address identifier of the communication terminal is searched for. This address identifier is an address identifier from the mobile

network. In particular, the address identifier of the communication terminal **1** is an identifier of the communication terminal **1** in GSM, UMTS, HSDPA, LTE or further mobile network technologies. Hereinafter, the address identifier is also referred to as MSISDN, which uniquely identifies the communication terminal **1** and thus also the user of the communication terminal **1** in the mobile network. If the address identifier or MSISDN is not stored in the communication terminal **1**, a query to a short dial application **8** is transmitted, as indicated by arrow A in FIG. **1**. As a response to this query, the address identifier is automatically transmitted to the communication terminal **1** and thus to the application **3** (arrow B). The address identifier can be stored in the communication terminal **1**, for example in the identification module **5**. Subsequently, that means with the address identifier being present, the application **3** establishes an initiation request for initiating a session with the service **4**, This initiation request may be contained in an https-request and the application **3** transmits with the request also the address identifier of the communication terminal **1** (arrow C). Subsequently, a request specific key, in particular a one-time-password (OTP) is transmitted to the communication terminal **1** from the service **4**. This is indicated in FIG. **1** by arrow D. This transmission does not occur within the communication connection, which had been established between the communication terminal **1** and the service server **2**, in which the communication is performed according to the http-protocol, but via a different communication channel. In particular, a communication connection in a mobile network, for example a GSM network, is used for transmission of short messages. It is particularly preferred that the transmission is thus performed in an SMS message, which hereinafter will also be referred to as SMS. The SMS is sent from the service server **2** or the service **4**, respectively, to the MSISDN which was contained in the initiation request. After receipt of this message, which contains the request specific key, in particular the OTP, at the communication terminal **1**, this key is automatically read out at the communication terminal **1** and is provided to the application **3**. The application **3** in addition receives the request specific identifier or session identifier, which can also be referred to as session token, as is indicated by the arrow E, from the service **4**.

[0059] In step F an access request which contains both the request specific identifier as well as the request specific key is then transmitted to the service **4**. The thus received data can then be compared by the service **4** with data which is stored on the service server **2**, in particular in the service **4**, that means with a request specific key or a request specific identifier, and the service can then decide on the authentication of the user at the service. Therein the request specific key and the request specific identifier, which have been generated with respect to a single initiation request, are compared with the received data. If the indications transmitted in the access request F, namely the request specific key (OTP) and the request specific identifier (session token) match with the ones which had been generated at the service server **2** for the address identifier of the communication terminal **1**, which was contained in the initiation request, the user and thus the session is authenticated with the service. In a further access within the session, this authentication is recognized and no further key is requested. For further accesses within the session, the transmission of the request specific identifier, that means the session token to the service server **2** is thus sufficient, as the service server **2** can determine by means of the session token whether the session has been authenticated or not. If the user is not authenticated, that means if his entitlement for the communication or session is not confirmed, a respective message can be transmitted to the communication terminal **1**, which informs the user, that he could not be authenticated for this service. Further access to the service **4** is then no longer possible.

[0060] The present invention will now be described again with respect to a use case.

[0061] If a smartphone app access an internet service via the internet, this communication, that means the user, is not authenticated at first. According to the state of the art, the user would then authenticate himself at a value added service by means of user identification (identity) and a password (secret) which has been set by the user. With the present invention the storing of such a secret can be omitted. Instead, the method according to the invention according to one embodiment comprises the following steps:

[0062] 1) Determining the Identity

[0063] Upon starting the smartphone app, it is checked whether the identity of the user has already been determined. The identity is for example the MSISDN (Mobile Subscriber ISDN Number) of the SIM-card which used in the smartphone, that means it identifies the user. If this identity has not yet been determined, the app initiates an empty SMS transmission in the background to a short dial in the network of the mobile network provider. An application, which merely copies the MSISDN of the sender and which includes the MSISDN as message text into a response SMS to the same number, serves as the recipient. The app can evaluate the response SMS and receives its own MSISDN, which it can use as identity and which it can store locally.

[0064] 2) Session Initiation

[0065] As soon as the app now accesses the value added service, which is also referred to as service, via the Internet, a so called https-request is triggered. In this request also the MSISDN is transported as an identification feature. The service, which is run on the server, identifies, that the connection has not yet been authenticated and now sends an SMS with a randomly generated secret key (one-time-password OTP) to the transmitted MSISDN. At the same time a temporary identifier is generated, which can also be referred to as session token, and is announced by means of an http-cookie to the client, which is the app which is installed on the smartphone. The cookie process requires that the session token is transmitted with each further access from the app.

[0066] 3) Authentication

[0067] The smart phone app receives the SMS with the OTP from the value added service and stores it temporarily. Then, a further http access is addressed to the value added service and the following information is transmitted therewith:

[0068] One-time-password (received per SMS)

[0069] Session token (received from the server via the Internet)

[0070] The value added service knows the connection of the two features already, as it has generated them itself. Only the recipient of the SMS, namely the user of the SIM card, can be in possession of both features. Consequently, the smartphone app has thus successfully authenticated itself at the value added service.

[0071] 4) Further Accesses

[0072] In every subsequent access only the session token which has been authenticated by the value added service is transmitted.

[0073] 5) Session Termination

[0074] After, for example, 20 minutes the session token automatically looses its validity with the value added service. A repeated or renewed session initiation according to step 2) is thus necessary.

[0075] With the present invention a simplified method of authentication of a user, in particular by authenticating a mobile terminal of the user via the Internet using an SMS-channel is provided.

[0076] With the present invention in particular the problem of the state of the art can be solved, that authentication features of an access from a mobile device via the Internet to a value added service is to be provided in a secure manner. Thereby, the situation, which was present so far, that mobile network subscriber, who access value added services via the Internet, for example a web application, cannot be reliably identified, is prevented.

[0077] Preferably, the authentication according to the present invention is lead through an SMS-channel, which due to the security means of the network which is used for this channel, for example the GSM network, is secure. By using an SMS-channel a non-authenticated access via the Internet by means of a secret key for a SIM-card which was transmitted via the SMS-channel, can be identified as an identity feature and can be authenticated.

[0078] In particular, the method according to the invention, for example, works on smartphones and in particular without interaction of the user. In addition, not only one single transaction can be authenticated, but the entire communication traffic, preferably within one session, with a value added service via the Internet can be authenticated.

[0079] The present invention has a number of advantages. In particular the present invention offers the possibility, to authenticate users, who are also referred to as subscriber or participants, of a mobile network for using any smartphone app and preferably to beforehand identify the user. Therein it is irrelevant, whether the user uses, for example, a WLAN connection for the access to the Internet or gets on the Internet, for example, via UMTS, in addition, according to the present invention an http-session is unlocked, that means authenticated, for a specific user, who is identified by the address identifier.

[0080] When using a WLAN or WiFi connection of the communication terminal with the service server, the method according to the invention can also be referred to as WiFi Auto Authentication per MSISDN. Herein, an automated identification and authentication of a user via MSISDN, that means by using the MSISDN in case of WiFi usage, is preferably performed. With the method according to the invention a secure usage of customer related data/transactions can be carried out without extensive additional registration.

[0081] The method according to the invention is, for example, usable for apps, integrated and extendable software and operating systems (cell phone, netbook, notebook, tablet, further devices with the option of identification via mobile communication (GSM, UMTS, HSDPA, LTE, further future mobile communication technologies), Web/Wap Logins, all further online registrations/authentifications, which can be authenticated by MSISDN, any further form, in which an identification of the customer is not possible due to the WiFi usage, which, however, becomes possible due to parallel mobile communication technology.

[0082] In addition, the invention may, for example, also be used by third party providers, that means by providers, who are not mobile network operators. Also users may conveniently verify themselves with the third party providers via their MSISDN, without having to log in with user name and password. For example, the authentication or registration according to the present invention can be carried out with services, such as, for example, facebook, ebay, Amazon, E-mail provider, online banking or further providers.

[0083] Thereby, the invention can be used for any online usage, which requires a password/login.

[0084] In case of usage of a mobile data connection, via the mobile network, the MSISDN can be also used for authentication without any additional verification process. Also there the user has an excellent user experience, which does not require further login, as the customer can identify himself via his MSISDN securely as himself.

REFERENCE NUMBERS

[0085] **1** Communication terminal
[0086] **2** Service server
[0087] **3** Application
[0088] **4** Value added service
[0089] **5** Identification module (SIM)
[0090] **6** Short dial application

1. Method of authenticating a user of a communication terminal (**1**), on which at least one application (**3**) is installed, at a service (**4**) on a service server (**2**), characterized in that an initiation request, which contains an address identifier of the communication terminal (**1**), is generated by the application (**3**) for initiating a session and is transmitted to the service server (**2**), the service server (**2**) after receipt of the initiation request automatically generates and transmits a request specific key (OTP) to the communication terminal (**1**) and the application (**3**) subsequently generates an access request in the session, wherein the application (**3**) integrates the received request specific key (OTP) automatically in the access request, and the access request is automatically transmitted from the communication terminal (**1**) to the service, server (**2**) in the session.

2. Method according to claim **1**, characterized in that the application (**3**) performs a query of an address identifier of the communication terminal (**1**) prior to generating the initiation request.

3. Method according to claim **2**, characterized in that the query of the address identifier is performed internally within the communication terminal (**1**) and only in case of a negative query result is performed externally to the communication terminal (**1**).

4. Method according to claim **3**, characterized in that the external query comprises the usage of a short dial.

5. Method according to claim **1**, characterized in that in addition to the request specific key at least one request specific identifier of the session is generated at the service server (**2**).

6. Method according to claim **5**, characterized in that the request specific identifier of the session is a temporary identifier.

7. Method according to claim **5**, characterized in that the request specific identifier is transmitted from the service server (**2**) to the communication terminal (**1**) via the communication connection, via which the initiation request had been transmitted to the service server (**2**).

8. Method according to claim **5**, characterized in that the request specific identifier is integrated into the access request by the application (**3**).

9. Method according to claim 5, characterized in that the request specific key and the request specific identifier are transmitted from the service server (2) via different communication connections to the communication terminal (1).

10. Method according to claim 9, characterized in that the request specific key is transmitted in a short message, in particular via a mobile network.

11. Method according to claim 9, characterized in that the request specific identifier is at least partially transmitted via a packet switched network, in particular the Internet, preferably within a cookie,

12. Method according to claim 1, characterized in that the application (3) reads the request specific key from at least one message in the communication terminal (1) for obtaining of the a request specific identifier.

13. Method according to claim 1, characterized in that the service (4) is a value added service.

14. Method according to claim 1, characterized in that before or during the transmission of the initiation request from the communication terminal (1) to the service server (2) a communication connection is established between the communication terminal (1) and the service server (2) and the initiation request is transmitted via this communication connection.

15. Method according to claim 1, characterized in that the method comprises:

a step of determining the identity of the user, in particular by determining an address identifier (MSISDN) of the communication terminal (1) in a mobile network,

a step of session initiation, in particular by sending an initiation request according to the Internet protocol, in particular an HTTPS request, within which the identity of the user, in particular by means of the MSISDN, is contained, to the service server (2), generating a request specific key, in particular an OTP, at the service server (2), sending of the request specific key to the communication terminal (1) which is associated to the address identifier via the mobile network, generating a temporary request specific identifier, in particular a session token, transmitting the session token by means of a cookie to the application (3) on the communication terminal (1),

a step of authenticating, in particular by generating an access request, within which the request specific key, in particular the OTP, and the request specific identifier, in particular the session token, are contained, and transmission of the access request to the service (4) on the service server (2), and

a step of verifying the request specific key, in particular the OTP, and the request specific identifier, in particular the session token, and the link of the key and the identifier on the service server (2) and confirmation of the authorization of the user in case of a positive verification result.

16. Application for a communication terminal (1) for access to at least one service (4) on a service server (2), characterized in that a user can be authenticated at the service by the application (3), that the application (3) is designed for at least generating and transmitting an initiation request for initiating a session, for automatically obtaining a request specific key, for generating an access request, within which the application (3) integrates the request specific key.

17. Application for a communication terminal (1) for access to at least one service (4) on a service server (2), characterized in that a user can be authenticated at the service by the application (3), that the application (3) is designed for at least generating and transmitting an initiation request for initiating a session, for automatically obtaining a request specific key, for generating an access request, within which the application (3) integrates the request specific key, characterized in that the application is designed for performing the method according to claim 1.

18. System for authenticating a user of a communication terminal, wherein the system has at least one communication terminal (1) and at least one service server (2) with a service (4) which is at least partially run on the service server (2), characterized in that at least one application (3) for communication with the service (4) on the service server (2) is present on the communication terminal (1), that the application (3) is designed for automatically establishing an initiation request, which comprises an address identifier of the communication terminal (1), that the service server (3) is designed for generating at least one request specific key on the base of the initiation request and for transmitting the at least one request specific key via different communication connections to the communication terminal (1), that the communication terminal (1) is designed for automatically generating and transmitting to the service server (2) an access request, within which the received request specific key is contained, and that the service server (2) is designed for comparing the request specific key, which is contained in the access request with the generated request specific key and to authenticate the user in case of a positive comparison result.

19. System according to claim 18, characterized in that the system comprises at least one short dial application (6) for communication of at least the address identifier of the communication terminal (1) to the communication terminal (1).

20. System for authenticating a user of a communication terminal, wherein the system has at least one communication terminal (1) and at least one service server (2) with a service (4) which is at least partially run on the service server (2), characterized in that at least one application (33) for communication with the service (4) on the service server (2) is present on the communication terminal (1), that the application (3) is designed for automatically establishing an initiation request, which comprises an address identifier of the communication terminal (1), that the service server (3) is designed for generating at least one request specific key on the base of the initiation request and for transmitting the at least one request specific key via different communication connections to the communication terminal (1), that the communication terminal (1) is designed for automatically generating and transmitting to the service server (2) an access request, within which the received request specific key is contained, and that the service server (2) is designed for comparing the request specific key, which is contained in the access request with the generated request specific key and to authenticate the user to case of a positive comparison result, characterized in that the system is designed for performing the method according to claim 1 and at which preferably an application (3) is installed on at least one communication terminal (1).

* * * * *

EXHIBIT G

Authentication in an **internet banking** environment

FFIE Council - FFIEC gencies (August 2001 Guidance), 2005 - profitguruonline.com

… reports, board resolutions identifying officers and **authorized** signers, and other business credentials. However, in an **Internet banking** environment, reliance on these traditional forms of …

☆ Save 99 Cite Cited by 177 Related articles All 39 versions »

Secure **internet banking authentication**

A Hiltgen, T Kramp, T Weigold - IEEE security & privacy, 2006 - ieeexplore.ieee.org

This article classifies common **Internet banking authentication** methods regarding potential threats and their level of security against common credential stealing and channel breaking …

☆ Save 99 Cite Cited by 298 Related articles All 15 versions

A multi-layer of multi factors **authentication** model for online **banking** services

MM Mohammed, M Elsadig - 2013 International Conference on …, 2013 - ieeexplore.ieee.org

… multi factor **authentication** with multi layer **authentication** … factor **authentication** model suitable for different **internet banking** … Therefore, **authorized** users are **authenticated** based on their …

☆ Save 99 Cite Cited by 43 Related articles All 2 versions

A study of the security of **Internet banking** and financial private information in South Korea

JH Lee, WG Lim, JI Lim - Mathematical and Computer Modelling, 2013 - Elsevier

… does this through a secure **authorization** Code service. In … a user trying to log in to **Internet banking** for the first time with … designated when subscribing to **Internet banking**. For example, …

☆ Save 99 Cite Cited by 35 Related articles All 4 versions

What is your husband's name? Sociological dimensions of **internet banking authentication**

S Singh, A Cabraal, G Hermansson - … of the 18th Australia conference on …, 2006 - dl.acm.org

… In our sample of 28 married persons using **Internet banking**, we find there are a variety of … Every time you want to make any changes, he has to ring and **authorize** you to talk to me …

☆ Save 99 Cite Cited by 16 Related articles All 7 versions

A formal classification of **internet banking** attacks and vulnerabilities

L Peotta, MD Holtz, BM David, FG Deus… - International Journal of …, 2011 - flyccs.com

… its role in the whole **authentication** and **authorization** process. In most cases, the security of the **authentication** and **authorization** process in online **banking** systems is based on secret …

☆ Save 99 Cite Cited by 93 Related articles All 7 versions »

An Experimental Investigation of the Usability of Transaction **Authorization** in Online **Bank** Security Systems.

M Al Zomai, B AlFayyadh, A Jøsang, A McCullagh - AISC, 2008 - academia.edu

… for data origin **authentication** used by online **bank** systems is … The term "**authorization** code" is often used by online **banks** … of OTP when discussing online **bank** security below. A good …

☆ Save 99 Cite Cited by 53 Related articles All 10 versions »

Online **banking authentication** system using mobile-OTP with QR-code

YS Lee, NH Kim, H Lim, HK Jo… - … Conference on Computer …, 2010 - ieeexplore.ieee.org

… tasks are also engaged in **internet** field. However, the existing **internet banking** system was exposed to … system performed the user **authentication** and digital signatures using **authorized** …

☆ Save 99 Cite Cited by 119 Related articles All 5 versions

On the security of **internet banking** in south korea

H Kim, J Huh, R Anderson - 2010 - ora.ox.ac.uk

South Korean **Internet banking** systems have a unique way of enforcing security controls. Users are obliged to install proprietary security software – typically an ActiveX plugin that …

☆ Save 99 Cite Cited by 33 Related articles »»

Security analysis for **internet banking** models

O Dandash, PD Le, B Srinivasan - Eighth ACIS International …, 2007 - ieeexplore.ieee.org

… the **Internet**. In this paper, we propose an **Internet Banking** Model (IBM) for fraudulent **Internet banking** … systems as they enforce strong **authentication** and **authorization** methods. We also …

☆ Save 99 Cite Cited by 29 Related articles All 5 versions

Related searches

sociological dimensions internet banking authentication

security of internet banking

online banking authentication

authentication **in an** internet banking **environment**

internet banking **mutual** authentication

trust in internet banking

internet banking **client side attacks**

internet banking **biometrics technologies**

internet banking **service**

internet banking **digital signature law**

internet banking **matter of risk**

online banking **services** authentication **model**

internet banking **south korea**

internet banking **protection mechanisms**

internet banking **two factor**

internet banking **preliminary evidence**

EXHIBIT H

The Wayback Machine - https://web.archive.org/web/20110504083053/http://www.businessinsider.com:80/google-stackoverflow-2011-3

Google Is Indexing Stack Overflow At 10 Times Per Second

Pascal-Emmanuel Gobry | Mar. 30, 2011, 6:54 AM | 1,346

Share Email A A A

Popular Q&A site Stack Overflow is getting indexed by Google 10 times per second, which is the maximum, co-founder Jeff Atwood wrote as a response to a question about the site's performence.



Stack Exchange co-founders Jeff Atwood and Joel Spolsky

In English, this means that Google's robots, which crawl the web to give you search results, are visiting Stack Overflow 10 times per second. This level of attention from Google's algorithm is a pretty impressive testament to how active and big the site is now.

Stack Overflow recently renamed itself to Stack Exchange and raised a boatload of venture capital to expand its popular Q&A site for programmers across many verticals like photography and cooking.

The news should be great for Stack Overflow's business over the long term. Because the site provides specific answers to specific questions, as opposed to the most open-ended, bloggy questions on hyped rival Quora, most of its traffic comes via long tail search engine queries. So having plenty of attention from Google's robots should mean, all else being equal, that Stack Overflow will get more traffic, which it can eventually monetize.

We'd be curious to know how often Quora is indexed by Google; if we had to bet, we'd say it's less. Part of it because the site is younger but part of it, we'd surmise, is because Quora's more open-ended questions overlap less with what people are actually searching for. And although it's early days for both companies, it's a hint that the most hyped company is not necessarily the one that ends up with the most success or the best business.

Related: Our Exclusive Q&A With Stack Exchange CEO Jeff Spolsky →

Follow Pascal-Emmanuel Gobry

For the latest tech news, visit SAI: Silicon Alley Insider. Follow us on Twitter and Facebook.

Tags: Stack Overflow, Startups, Quora | **Get Alerts for these topics »**

Advertisement:

Share: Short URL http://read.bi/icJIct

Twitter Facebook Buzz Digg StumbleUpon Buzz

Reddit LinkedIn Email Embed Alerts Newsletter

2011 STARTUP SAI May 10, NYC Get Tickets Now »

Get SAI Emails & Alerts SIGN-UP

Learn More »

Advertisement

Your Money

Economic Update



MAY DAY, MAY DAY: The Ultimate Guide To The Slowing Global Economy

Hussman: This Market Is Setting Up Just Like Some Of The Worst Markets Of All Time

The St. Louis Fed Presents Its Guide To Our Unemployment Nightmare

SILVER: Everything You Need To Know About The World's Hottest Investment

A New Guide To How The Middle Class Is Being Crushed By Rising Prices, Falling Wages, And Declining Security

5 Huge Economic Events That Will Define The Coming Week

Business Insider Mobile GET IT NOW

Blackboard Home »

Quora

Edit This »

Summary

Quora is an online collection of user-created questions and answers. It founded in June 2009, launched in private beta in December 2009, and made available to the public on June 21, 2010. Quora aggregates questions and... More »

Stack Overflow

Edit This »

Summary

Stack Overflow is a programming question and answer site that is built and maintained collaboratively. More »



Pascal-Emmanuel Gobry is a Paris-based entrepreneur and writer for Business Insider.

Contact:
e-mail: **pegobry@businessinsider.com**
URL: **http://pegobry.tumblr.com/**
Subscribe to his **RSS feed** | **twitter feed**

Recent Posts

Today In Ridiculous French ...
Meet MyHabit, Amazon's Gilt...
Sorry Bloomberg: RenRen Is ...

Comments on this post are now closed.

Read Me Become a Contributor

On The Web, Bin Laden News Is Big--But Not As Big As Soccer



No debate that Osama bin Laden's death is...
– Peter Kafka



How Yahoo Lost Its Grip 11
– Marc Cenedella



The Most Interesting Tweets About Osama Bin Laden's Death 42
– Pascal-Emmanuel Gobry



Television Ownership Drops In U.S. For First Time In 20 Years
– Brian Stelter



10 Secrets Apple Doesn't Want You To Know About Mac OS X Lion 4
– Steve Kovach

Five social media presence strategies
– Tristan Louis

Flying Software Engineers To Boulder For Startup Week
– Brad Feld

Why We May Be In Another Tech Bubble
– Adam Piore

How Facebook, Google, And Twitter Were Born
– Greg Voakes

Feedly Takes RSS Into The Tablet Wars
– Robert Scoble

Beyonce Sued By Game Developer For $100 Million In Dropped Deal
– IndustryGamers

Facebook Testing, Promoting Around Proximity
– Ryan Spoon

More »

The Hive

What Smart People Are Reading Right Now

TechStars » Funding and Mentorship from the #1 Startup Accelerator in the World 18

The Situation Room Meme: The Shortest Route From Bin Laden to Lulz - Alexis Madrigal - Technology - The Atlantic 10

Bin Laden Aides Bought Big Orders of Pepsi, Coke, Grocer Says - Bloomberg 9

More All »

Most Read

Read Commented


Meet The 'Seal Team 6', The Bad-Asses Who Killed Osama Bin Laden
431,482 Views


Here's Who Was Watching The Bin Laden Raid In The White House Situation Room 373,704 Views


The Most Interesting Tweets About Osama Bin Laden's Death
336,860 Views


PHOTOS: See How The World Reacted To Bin Laden's Death
318,843 Views


The 20 Most Expensive Domain Names Of All Time 313,553 Views

Jobs

.Net Development Manager 15404
Los Angeles, CA

More Jobs > **Post a Job >**

Advertisement

Thanks to our partners

DATAPIPE Your Competitive Edge
OpenX
catchpoint
OOYALA
Ad-Juster
Financial CONTENT

Hot Questions

What Smart People Are Discussing Right Now

when will this issue with playstion network be resolved **0 Answers**

Does anyone at Business Insider actually proof the articles you print **0 Answers**

Trust Companies in Mauritius **0 Answers**

More | Ask A Question » All »

A-Z INDEX
COMPANIES
AUTHORS
TAGS
SITE MAP
LATEST
CONTRIBUTORS
VIDEO

TOOLS
STOCK QUOTES
JOB LISTINGS

LISTS & RANKINGS
DIGITAL 100
SILICON ALLEY 100
SILICON VALLEY 100
AMERICA'S BEST COLLEGES
BEST BUSINESS SCHOOLS
SEXIEST CEOS
MORE

YOUR ACCOUNT
REGISTER
CHANGE YOUR EMAIL
PREFERENCES

ABOUT BI
ABOUT
JOBS AT BI
MASTHEAD
CONTACT
ADVERTISE
MOBILE

FOLLOW BI
EMAIL NEWSLETTERS
ALERTS
RSS
TWITTER
LINKEDIN
FACEBOOK

VERTICALS
TECH
MEDIA
WALL STREET
MARKETS
STRATEGY
SPORTS
TRAVEL
TOOLS
POLITICS
PR
MISC.
LIFESTYLE

* Copyright © 2011 Business Insider, Inc. All rights reserved. Registration on or use of this site constitutes acceptance of our Terms of Service and Privacy Policy. | Disclaimer

Powered by MongoDB | Hosted by Datapipe | Web analytics by Empirical Path

EXHIBIT I

multi-factor authentication phone number internet banking

Sign in

AI Mode | All | Images | Videos | Shopping | Short videos | Books | More | Tools

Before Dec 31, 2013
All results
Advanced Search
Clear

About 5,750,000 results (0.29s)

TD Bank
https://www.td.com › about-td › how-we-protect-you

Protecting your accounts with Two-Step Verification

Sep 10, 2011 — If you do not have access to a previously verified **phone number**, please contact EasyLine **Telephone banking** at 1-866-222-3456. Tip: You can adjust the ... Read more
Missing: ~~internet~~ | Show results with: internet

Cryptomathic
https://www.cryptomathic.com › hubfs › docs

Two-Factor Authentication for Banking PDF

May 7, 2009 — As a leading security vendor to the financial sector, Cryptomathic have extensive experience in developing and delivering **two-factor authentication** solutions. Read more
19 pages

FFIEC BSA/AML (.gov)
https://bsaaml.ffiec.gov › resources › new_5_2007

OCC BULLETIN PDF

Oct 12, 2005 — Used in a multifactor authentication process, the customer first enters his or her user name and password in the established manner.
18 pages

Provident Bank
https://www.provident.bank › out-of-band-authentication

Authentication Tools for Business eBanking: Good, Bad, ...

Oct 16, 2013 — Out-of-Band Authentication Phone Call- You first select a phone number from the list on record with your bank (this may include work lines and mobile numbers). ... Read more

First Tech Federal Credit Union
https://www.firsttechfed.com › help › security › prevent

Preventing Security Issues

Dec 12, 2004 — Use two-factor authentication. · **Log on to Online Banking** · Click the “Settings” option on the top of the right screen · Click the “Security” tab · Under the “Two- ... Read more

BDO USA
https://www.bdo.com › getmedia › BDOUSA_MF...

Multi-factor Authentication Client Enrollment PDF

Aug 31, 1982 — your bdo.com account reset due to replacing a phone, new phone number etc., please contact the BDO Client Support help desk at **800-546-2191** from 9:00 am to 5:00 ... Read more
10 pages

SolidPass
http://www.solidpass.com › solutions › two-factor-authen...

Two-Factor Authentication - 2FA

Mar 16, 2010 — Two Factor Authentication (2FA) is **using a combination of two different "factors"** to gain access to or authenticate online banking. Read more

Authy
https://authy.com

Authy: Two-factor Authentication (2FA) App & Guides | Authy

May 25, 2012 — **Two-factor authentication (2FA)** is the best way to protect yourself online. Defeat cyber criminals & avoid account takeovers with stronger security, for free! Read more

Download | Two-Factor Authentication (2FA) | Reset account | App features

Missing: ~~internet~~ | Show results with: internet

Bank of America
https://www.bankofamerica.com › security-center › faq

Online Banking Security Features FAQs

Oct 5, 2012 — However, if a U.S. based **mobile number** is not available, you may choose to use a USB security key. We do not support international **phone numbers** at this time. Read more

Information Security Stack Exchange
https://security.stackexchange.com › questions › could-...

Could my bank's two-factor authentication be hacked?

May 29, 2013 — Clone your **phone's** sim, and request a **banking** code to be sent to your **phone's number**. they could also possibly clone a non-sim **phone** as well · Steal your **phone**. Read more

92037, San Diego, CA - From your IP address - Update location

Help Send feedback Privacy Terms

EXHIBIT J

The Wayback Machine - https://web.archive.org/web/20141107231653/http://betabeat.com/2011/01/forget-quora-new-yorks-stack-overflow-is-killing-it/

Q&A

Forget Quora, New York's Stack Overflow Is Killing It

BY ADRIANNE JEFFRIES | 1/25/11 11:03AM

Like 0
Tweet
Share
Email
More



As the tech blogosphere continues to gush and agonize over the question-and-answer site Quora, New York's question-and-answer site Stack Overflow has been quietly catching fire.

Stack Overflow is primarily a message board where programmers can share tips and tricks and crowdsource solutions to problems they run into while coding. The name is a reference to aprogramming error.

Stack Overflow grew from seven million visitors to over 16 million in 2010, putting the site in Quantcast's top 400, CEO Joel Spolsky wrote in a blog post. The company raised $6 million from Union Square Ventures and angel investors last spring and went from three full-time employees to 27.

It's been so successful that the company launched several new sites in 2010, including a Q&A site forcooking, math, photography and English.

Over 80 percent of questions get a good answer, Mr. Spolsky wrote, and many of the new Stack Exchange sites have 100 percent answer rates. One of the issues with Quora, a well-funded Bay Area startup founded by former Facebook employees, is the high number of unanswered questions.

One way to solve this problem, Mr. Spolsky realized, was to separate Q&A sites by topic—that way each site attracts a more engaged group.

"We learned a long time ago that the only way to get questions answered promptly is to have a critical mass of knowledgeable users, so we have an onerous process called Area 51 where sites are proposed, discussed, and voted on. If a proposed site doesn't have critical mass, we just won't create it. Even if it does get created, it has to maintain a certain level of traffic and quality or we'll close it down," he said.

Another New York company has had success by targeting developers—Forrst, which is like a Tumblr for programmers, has been growing rapidly since founder Kyle Bragger started it as a side

CONNECT WITH US

Sign up for our Newsletter SEND

Send an anonymous tip SEND

Popular on Betabeat

Plaid Ex1003
Page 369 of 372

project.

UPDATE: Mr. Spolsky pointed out on the tech news forum Hacker News that Quora and Stack Overflow are targeting different kinds of questions. "We're the reference section of the library, they're an awesome salon where smart people are shootin' the shit," he wrote. The questions on Stack Overflow and its associated sites are the sort of questions that have definite answers, like "Is there a word or phrase for the feeling you get after looking at a word for too long?" from the English Language & Usage site (answer: semantic satiation. Those questions can be found on Quora, too, but it also attracts questions with subjective answers, i.e. "What tech company will we all be talking about at SXSW 2011?"

Check out the 25 Most Popular Gawker User Passwords >>

adrjeffries [at] observer.com | @adrjeffries

FOLLOW **ADRIANNE JEFFRIES** **ON TWITTER** OR VIA **RSS**. AJEFFRIES@OBSERVER.COM

- Like 0
- Tweet
- Share
- Email
- More

TAGS: Q&A, TECH, JOEL SPOLSKY, STACK EXCHANGE, STACK OVERFLOW

EXHIBIT K

Q&A websites like Quora and Stack Exchange take off

Updated | Comment | Recommend



Enlarge By Karen Bleier, AFP/Getty Images file

Hot on the Web these days: Question-and-answer websites.

By Jon Swartz, USA TODAY

SAN FRANCISCO — What's one of the hottest new areas for technology investment? The answer: question-and-answer websites.

Stack Overflow is the latest to join the fray on the heels of new Silicon Valley darling Quora.

Today, Stack publicly launches Stack Exchange, a network of 33 Q&A sites on topics ranging from computer science to cooking. The service, in beta testing for seven months, has more than 1.5 million users, up 50% from November. It is looking to add one topic site a week.

"This is a specialist's site," says Stack Overflow CEO Joel Spolsky. "We're going to be the reference section of the library, where you can reach experts and get definitive answers."

There seems to be no limit to websites devoted to answering questions for consumers. Many are not only adding millions of customers but millions of dollars from tech investors.

Last year, Stack raised $6 million from Union Square Ventures.

Meanwhile, recent buzz over Quora has been positively Twitter-like.

The Q&A site — founded by former Facebook employees and already valued at $86 million, based on undisclosed funding from Benchmark Capital— combines elements of Facebook, Yahoo Answers, Wikipedia, Twitter, and two of its competitors, Ask.com and JustAnswer.

There have been Q&A websites for years, such as Answers.com. Texting and website service ChaCha started in 2005. But newer services such as Stack and Quora are narrower in focus, with experts providing the answers.

Spolsky divides the fledgling field into two categories: horizontal sites, where anything can be asked and anyone can answer, and vertical sites, such as Stack Exchange. "Listen, if you are a math researcher at UC-Berkeley, you're not going to go to Yahoo Answers," Spolsky says. "Our motto is to link experts together, so they can get definitive answers on specific topics."

Stack Overflow makes its money through targeted advertising.

Meanwhile, better-established Q&A sites say there's a healthy market for competing players.

"There is a lot of interest because people have a discrete need to have questions answered," says Doug Leeds, CEO of Ask.com, which combines search and Q&A services. Ask has 90 million users worldwide.

At least one company is getting people to pay for its expertise. Millions have paid about $20 per question to get medical, legal, financial and other advice on JustAnswer, started seven years ago by CEO Andy Kurtzig.

Posted

Updated

To report corrections and clarifications, contact Standards Editor **Brent Jones**. For publication consideration in the newspaper, send comments to letters@usatoday.com. Include name, phone number, city and state for verification. To view our corrections, go to corrections.usatoday.com. We've updated the Conversation Guidelines. Changes include a brief review of the moderation process and an explanation on how to use the "Report Abuse" button. Read more.