

UNITED STATES PATENT AND TRADEMARK OFFICE

BEFORE THE PATENT TRIAL AND APPEAL BOARD

GOOGLE LLC,
Petitioner,

v.

HUMAN DIFFERENTIAL INTELLIGENCE, LLC,
Patent Owner.

Case No. TBD
Patent No. 8,752,141

DECLARATION OF MARY K. BOLIN, PH.D.

Google Exhibit 1035 Google v. HDI

I, Mary K. Bolin, declare as follows:

1. I am Emeritus Professor at the University of Nebraska, Lincoln Libraries. I have held the following positions: Professor and Catalog and Metadata Librarian, Digital Initiatives and Special Collections Department, University of Nebraska, Lincoln Libraries (2015-2020); Professor and Chair, Discovery and Resource Management and Coordinator of Collection Development, University of Nebraska, Lincoln Libraries (2004-2015); Lecturer, San José State University School of Information (2008-present); Head of the Technical Services Department, University of Idaho Library (1993-2003; promoted to Professor in 1999); Head of the Cataloging Department, University of Idaho Library (1986-1993; promoted to Associate Professor and tenured in 1991); Head of Records Maintenance Section, Cataloging Department, University of Georgia Libraries (1981-1986).
2. Over the course of my work as a librarian, professor, researcher, student research supervisor, and author of numerous scholarly papers and other publications, I have had extensive experience with cataloging records and online library management systems built around Machine-Readable Cataloging (MARC) standards. I also have substantial experience in authenticating printed documents and establishing the date when they were accessible to ordinarily skilled researchers.
3. Exhibit A is my full resume.

4. I have been retained by Wolf, Greenfield, & Sacks, P.C., counsel for Petitioner Google LLC, to determine the authenticity and dates of public availability of certain documents in the *inter partes* review proceedings of U.S. Patent No. 8,752,141 (“the ’141 patent”). For this service, I am being paid my usual hourly fee of \$100 per hour, plus actual expenses. My compensation is not dependent in any way upon the outcome of the *inter partes* review of the ’141 patent.

I. DOCUMENTS OF INTEREST

5. The documents that are the subject of this declaration are the following:

Document 1. IMAGINATION: A Robust Image-based CAPTCHA Generation System. *MM’05, November 6–11, 2005, Singapore MM’05, November 6–11, 2005, Singapore*

Document 2. Luis von Ahn, Manuel Blum, Nicholas J. Hopper, John Langford. CAPTCHA: Using Hard AI Problems. *International Conference on the Theory and Applications of Cryptographic Techniques. EUROCRYPT 2003: Advances in Cryptology.*

Document 3. G. Kochanski, D. Lopresti, and C. Shih. “A Reverse Turing Test Using Speech,” *Proceedings of the Seventh International Conference on Spoken Language Processing*, September 2002, Denver, CO, pp. 1357-1360.

Document 4. Luis von Ahn. *Human Computation*. Dissertation, Carnegie Mellon University, 2005.

II. METHODOLOGY

6. I searched for each of these documents (1) in subscription databases (large repositories of reliable scholarly content from well-known publisher) such as *IEEE Xplore*, *SpringerLink*, and *ACM Digital Library*, and (2) in *Google Scholar* (an open access database that aggregates content from subscription sites as well as other freely available repositories of scholarly and research data). I searched for Document 4, a dissertation, in the Online Computer Library Center (“OCLC”) database. I searched the conference proceeding volumes for Documents 1-3 in OCLC as well. I also searched the library catalogs of institutions such as the University of Nebraska, Lincoln and other large research libraries. My goal was to find a record in one or more of these databases that contained a date indicating when the work in question was available to the public.

7. *Use of MARC Records.* The “Machine Readable Cataloging” standard, also known as “MARC,” is an industry-wide, standard method of organizing library catalog information. MARC was first developed in the 1960s by the Library of Congress. A MARC-compatible library is one that has a catalog consisting of individual MARC records for each of its works. Today, MARC is the primary

communication protocol for the transfer and storage of bibliographic metadata in libraries.

8. When an OCLC participating institution acquires a work, it creates a MARC record for this work in its computer catalog system in the ordinary course of its business and OCLC automatically supplies the date of creation. The MARC record creation date reflects the date on which (or shortly after which) the item was first acquired and catalogued. Once the MARC record is created by a cataloger at an OCLC participating member institution, it is instantaneously available to any OCLC participating members, and, therefore, available to the public.

9. A MARC record comprises several fields, each of which contains specific data about the work. Each field is identified by a standardized, unique, three-digit code corresponding to the type of data that follow. For example, a work's title is recorded in field 245, the primary author of the work is recorded in field 100, an item's International Standard Book Number ("ISBN") is recorded in field 020, an item's Library of Congress call number is recorded in field 050, and the publication date is recorded in field 264 under the subfield "c." If a work is a periodical, then its publication frequency is recorded in field 310, and the publication dates (*e.g.*, the first and last publication) are recorded in field 362, which is also referred to as the enumeration/chronology field. The library that

created the record is recorded in field 040 in subfield “a” with a unique library code.

10. OCLC provides its members online access to MARC records through its OCLC Connexion database. When viewing the MARC record online via the OCLC Connexion database, hovering over this code with the mouse reveals the name of the library. I used this method of “mousing over” the library codes in the OCLC database to identify the originating library for all the MARC records in this declaration. In the MARC records in the OCLC Connexion database, MARC field 040 contains the OCLC symbols of institutions transcribing, creating, and modifying the record.

11. OCLC was created “to establish, maintain and operate a computerized library network and to promote the evolution of library use, of libraries themselves, and of librarianship, and to provide processes and products for the benefit of library users and libraries, including such objectives as increasing availability of library resources to individual library patrons and reducing the rate of rise of library per-unit costs, all for the fundamental public purpose of furthering ease of access to and use of the ever-expanding body of worldwide scientific, literary and

educational knowledge and information.”¹ Among other services, OCLC and its members are responsible for maintaining the WorldCat database (<http://www.worldcat.org/>), used by independent and institutional libraries throughout the world. OCLC is the international cooperative to which libraries catalog records and others pay to access MARC records. The MARC records referenced in this declaration, extracted from the OCLC database, were generated in the ordinary course of business. Further, I personally located and downloaded each of these records.

12. MARC records created at the Library of Congress are tape-loaded into the OCLC database through a subscription to MARC Distribution Services daily or weekly. Once the MARC record is created by a cataloger at an OCLC member institution or is tape-loaded from the Library of Congress, it is then made available to any other OCLC members online, and, therefore, made available to the public. As noted before, when an OCLC member institution creates a new record, OCLC automatically supplies the date of creation for that record. The MARC record

¹ Third Article, Amended Articles of Incorporation of OCLC Online Computer Library Center, Incorporated (available at <http://www.oclc.org/en-US/membership/councils.html>).

creation date reflects the date on which, or shortly after which, the item was first acquired or cataloged.

13. The date of creation for the MARC record appears in the fixed field (008), characters 00 through 05. Field 005 is a 16-digit field that record the Date characters yyyymmdd include 4 for the year, 2 for the month, and 2 for the day.

The Date entered on file never changes. These dates are automatically supplied by the OCLC system and cannot be changed by the user. When the MARC record is created in OCLC, it is available throughout the world to other libraries and in WorldCat, the public version of OCLC Connexion. Thus, the date of creation of the MARC record by a cataloger at an OCLC member institution reflects when the underlying item is accessible to the public.

14. For each of Documents 1-4, I searched OCLC, www.oclc.org, for the conference proceedings (Documents 1-3) or the dissertation (Document 4).

15. For those documents for which there is a record in OCLC, I determined when the record was created and which libraries own the item. I then visited the website of the library or libraries that own the item, searched their library catalog, and determined when the item was added to the library's collection.

16. I am familiar with many online catalogs, databases, and search engines. In preparing for this declaration I used authoritative information systems such as WorldCat (<https://www.worldcat.org>), the online catalog of the Library of

Congress (<https://catalog.loc.gov>), the online catalog of a number of large research libraries such as the University of Nebraska, Lincoln, the public catalog of the United States Copyright Office (<https://cocatalog.loc.gov/>), LexisNexis, IEEEExplore, Google Books, and Google Scholar search for records. These records are identified and discussed in this declaration.

III. AUTHENTICATION OF DOCUMENT 1

17. Document 1 was found at: https://link.springer.com/chapter/10.1007/3-540-39200-9_18 with the full text PDF at <http://infolab.stanford.edu/~wangz/project/imsearch/IMAGINATION/ACM05/datta.pdf>. The face of the document accurately reflects the title, author, and publication information listed on the webpage at that URL, and document appears to be complete.

18. The Properties of the PDF state that it was created September 16, 2005. There is a MARC record for the conference proceedings. It was added to the OCLC database on February 16, 2007.

19. Attachment 1 is a PDF of the article from Stanford Infolab.

20. Attachment 2 is a screen capture of the Properties for this PDF.

21. Attachment 3 is a MARC record for the conference proceedings.

IV. PUBLIC AVAILABILITY OF DOCUMENT 1

22. Based on the fact that the metadata for Document 1 states that it was created in 2005 and that there is a MARC record for the conference proceedings where this paper was presented that was created in OCLC on February 16, 2007, this document would have been available publicly in late 2005, and available in OCLC in 2007, and could have been retrieved by a skilled researcher.

V. AUTHENTICATION OF DOCUMENT 2

23. Document 2 was found in SpringerLink at https://link.springer.com/chapter/10.1007/3-540-39200-9_18 where it states that it was first online 13 May 2003. The PDF of the document is available at: https://link-springer-com.libproxy.unl.edu/content/pdf/10.1007%2F3-540-39200-9_18.pdf. The face of the document accurately reflects the title, author, and publication information listed on the webpage at that URL, and document appears to be complete.

24. Document 2 was cited in: Shyong K Lam, John Riedl. Shilling Recommender Systems for Fun and Profit. *WWW '04: Proceedings of the 13th international conference on World Wide Web*. May 2004. Pages 393–402. <https://doi.org/10.1145/988672.988726>. That paper was found in *ACM Digital Library*: <https://dl-acm-org.libproxy.unl.edu/doi/pdf/10.1145/988672.988726>.

25. A MARC record for the print conference proceedings for WWW 04 is available in OCLC. The MARC record was created on June 1, 2005.
26. A patent application that cites this article, filed August 16, 2007, was found in the US Patent and Trademark Office (USPTO) database at [http://appft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&u=%2Fnetacgi%2FPTO%2Fsearch-adv.html&r=14&p=1&f=G&l=50&d=PG01&S1=\(%22physical+token%22.TTL.\)&OS=ttl/%22physical+token%22&RS=TTL/%22physical+token%22](http://appft.uspto.gov/netacgi/nph-Parser?Sect1=PTO2&Sect2=HITOFF&u=%2Fnetacgi%2FPTO%2Fsearch-adv.html&r=14&p=1&f=G&l=50&d=PG01&S1=(%22physical+token%22.TTL.)&OS=ttl/%22physical+token%22&RS=TTL/%22physical+token%22).
27. A European patent application that cites this article, filed February 10, 2006, was found on Google Patents at <https://patentimages.storage.googleapis.com/fc/80/39/3923492bf5f5af/EP1845468A1.pdf>.
28. A European patent that cites this article, filed February 28, 2008, was found on Google Patents at <https://patentimages.storage.googleapis.com/1f/5b/dc/d75817754bf678/EP2130159B1.pdf>.
29. Attachment 4 is a PDF of the article from SpringerLink.
30. Attachment 5 is a PDF of the Lam and Riedl article from *ACM Digital Library*.
31. Attachment 6 is a MARC record for the conference proceedings.

32. Attachment 7 is a PDF of the search results page from SpringerLink showing the availability date of May 2003.
33. Attachment 8 is a PDF of the patent application from the USPTO database.
34. Attachment 9 is a PDF of the European patent application from Google Patents.
35. Attachment 10 is a PDF of the European patent from Google Patents.
36. Attachment 11 is a list of articles, books, and chapters that cited this article between 2004 and 2008.

VI. PUBLIC AVAILABILITY OF DOCUMENT 2

37. Based on the fact that the article was available in SpringerLink and was first online in 2003, the MARC record for the conference proceedings was available in 2005, the pre-June 2008 patent publications cite to Document 2, and the Lam and Riedl article that cites Document 2 was published in 2004, Document 4 was publicly available starting in 2003 and could have been retrieved by a skilled researcher.

VII. AUTHENTICATION OF DOCUMENT 3

38. This article was retrieved from https://www.isca-speech.org/archive/archive_papers/icslp_2002/i02_1357.pdf. The Properties of this PDF file show that it was created on July 30, 2002. The conference proceedings are found at: <https://www.semanticscholar.org/paper/Proceedings-of->

[the-7th-International-Conference-on-Tan-](#)

[Dalsgaard/f2d898947f5f0c7ae107a005386c1301ddc662bf](#). That page states that the proceedings were published in 2002. The PDF link from the conference proceedings to the paper is

[http://www.cse.lehigh.edu/~lopresti/Publications/2002/icslp02.pdf](#), the faculty website of one of the authors. Apart from the blue text at the top of the first page in the isca-speech.org PDF, the two documents are substantively identical. The properties of that PDF show that it was created on April 1, 2002. That website was archived by the Internet Archive (web.archive.org) at least as early as September 1, 2004, indicating that the paper was publicly accessible on that website by that date.

See

[http://web.archive.org/web/20040901075237/http://www.cse.lehigh.edu/~lopresti/Publications/2002/icslp02.pdf](#). The face of the document accurately reflects the title, author, and publication information listed on the webpage at that URL, and document appears to be complete.

39. Attachment 12 is the PDF from isca-speech.org.

40. Attachment 13 is the PDF Lehigh University.

41. Attachment 14 is a screenshot of the PDF from isca-speech.org, showing the creation date.

42. Attachment 15 is a screenshot of the PDF from Lehigh University, showing the creation date.

43. Attachment 16 is a screenshot of the web page for the conference proceedings, where it states that the proceedings were published in 2002.

44. Attachment 17 is a screenshot of the web.archive.org web page for the Lehigh University website, showing that the paper was available on Lehigh University's website by September 1, 2004.

VIII. PUBLIC AVAILABILITY OF DOCUMENT 3

45. Because PDFs of this document in multiple locations were created in 2002, because the conference proceedings state that they were published in 2002, and because web.archive.org indicates the document was available online by September 1, 2004, this document was publicly available in 2002, or at the latest in September 2004, and could have been found by a skilled researcher.

IX. AUTHENTICATION OF DOCUMENT 4

46. Document 4 was found in the online library catalog of Carnegie Mellon university at MARC Record:

https://cmu.primo.exlibrisgroup.com/discovery/sourceRecord?vid=01CMU_INST:01CMU&docId=alma991000901079704436&recordOwner=01CMU_INST where it states that the record was created October 20, 2006.

47. Document 4 was also found in Dissertation Abstracts: <https://search-proquest-com.libproxy.unl.edu/pqdt/docview/305006687/7F821E62E4614922PQ/4?accountid=8116> where it states that it was made available by ProQuest dissertation publishing in 2005.

48. Attachment 18 is a PDF of the MARC record for the dissertation from the Carnegie Mellon University online library catalog.

49. Attachment 19 is a PDF of record from the Dissertation Abstracts database.

X. PUBLIC AVAILABILITY OF DOCUMENT 4

50. Based on the fact that the MARC record in the Carnegie Mellon library catalog was created in October 2006, and the record in the Dissertation Abstracts database was created in 2005, Document 4 was available sometime in 2005, and at the latest by October 2006, and could have been retrieved by a skilled researcher.

I hereby 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 are made with the knowledge that willful false statements and the like are punishable by fine, imprisonment, or both, under Section 1001 of Title 18 of the U.S. Code.

I declare under penalty of perjury under the laws of the United States of America that the foregoing is true and correct.

Executed on SEPTEMBER 13, 2020



Mary K. Bolin, Ph.D.

EXHIBIT A

Mary K. Bolin, PhD

Professor Emeritus
University Libraries
University of Nebraska—Lincoln
mbolin2@unl.edu

Lecturer,
School of Information
San José State University
mary.bolin@sjsu.edu

EDUCATION

PhD, Educational Studies (Educational Leadership and Higher Education), 2007, University of Nebraska

Dissertation: *A Typology and Discourse Analysis of the Status and Appointments of Librarians at Land Grant Universities* (available at digitalcommons.unl.edu)

MA, English, 1999, University of Idaho
Thesis: *Grace: A Contrastive Analysis of a Biblical Semantic Field* (available at digitalcommons.unl.edu)

MSLS, 1981, University of Kentucky

BA, Linguistics, 1976, University of Nebraska—Lincoln

WORK EXPERIENCE

Center for Digital Research in the Humanities. Fellow, 2019- Emeritus, 2020-

University of Nebraska--Lincoln Libraries: Professor and Catalog and Metadata Librarian, Digital Initiatives and Special Collections, 2015-2020. Emeritus, 2020-

University of Nebraska—Lincoln Libraries: Professor and Chair, Discovery and Resource Management, 2004-2015; Coordinator of Collection Development, 2012-2015

University of Idaho Library: Head, Technical Services Department, 1993-2003. Promoted to

Professor in 1999

University of Idaho Library: Head, Cataloging Department, 1986-1993. Promoted to Associate Professor and tenured, 1991

University of Georgia Libraries: Head, Records Maintenance Section, Cataloging Department, 1981-1986

Teaching:

San José State University School of Information:

LIBR 281, Metadata. 2009-

LIBR 248, Introduction to Cataloging and Classification. 2013-

LIBR 202, Information Retrieval. 2008-2013

American Library Association

Cataloging for Non-Catalogers, Online Continuing Education Course, 2017-

Intermediate Cataloging for Non-Catalogers, Online Continuing Education Course, 2018-

Independent Study in Idaho

Library Science C415, Technical Services for Small Libraries, 2004-2013

Library Science C416, Library Services to Rural Libraries, 2004-2013

Library Science C422, Use of the School Library, 1999-2013

Library Science C425/525, Organization and Management of Small Libraries, 1988-2013

Library Science C427, Library or Media Center Practicum, 1995-2013

Gateway PhD Program (San Jose State University and Queensland University of Technology). SJSU supervisor for two PhD students, 2015-

Consulting

Expert witness for multiple attorneys employed by Wolf, Greenfield & Sacks, P.C., Hughes Hubbard & Reed LLP, and other firms. 2017-

SCHOLARSHIP

Refereed Publications:

(Publications available from digitalcommons.unl.edu)

Pytlík Zillig, Brian, and Mary K. Bolin. (Submitted) "Shakespeare and the Enumeration of Semantic Primes." *Japanese Journal of Digital Humanities*.

Wintermute, Harriet E., and Bolin, Mary K. (2018). "Metadata Generators for Backlog Reduction: Metadata Maker Streamlines Cataloging and Facilitates Transition. *Technical Services Quarterly* 35(2) doi: <https://doi.org/10.1080/07317131.2018.1422879>

Bolin, Mary K. (2018). "Metadata, Catalog Systems, Linked Data, and the Evolving ILS." In: Sandra Hirsh (Ed.), *Information Services Today: An Introduction*. 2nd ed. New York: Rowman and Littlefield.

Bolin Mary K. (2017). *The 21-century Academic Library: Global Patterns of Organization and Discourse*. Oxford: Chandos Publishing. (Chandos Information Professional Series)

Bolin, Mary K. (2017). Linguistics and LIS: A Research Agenda." *SLIS Student Research Journal*, 7:1.

Retrieved from <http://scholarworks.sjsu.edu/slissrj/vol7/iss1/2>

Bolin, Mary K. (2014). "Open Access Web Resources for Library Continuing Education and Training." In S. Hines (Ed.), *Revolutionizing the Development of Library and Information Professionals: Planning for the Future* (pp. 107- 124). Hershey, PA: . doi:10.4018/978-1-4666-4675-9.ch008.

Bolin, Mary K. (2014). "The Language of Academic Librarianship: The Discourse of Promotion and Tenure." *Advances in Library Administration and Organization* 32.

Bolin, Mary K. "Librarian Status at US Research Universities: Extending the Typology." *Journal of Academic Librarianship* 34:4 (September 2008)

Bolin, Mary K. "A Typology of Librarian Status at Land Grant Universities." *Journal of Academic Librarianship*, 34: 3 (May 2008)220:230.

Bolin, Mary K. "The Library and the Computer Center: Organizational Patterns at Land Grant Universities." *Journal of Academic Librarianship*, 31:1 (2005)

Eckwright, Gail Z. and Mary K. Bolin. "The Hybrid Librarian: The Affinity of Collection Management with Technical Services and the Organizational Benefits of an Individualized Assignment." *Journal of Academic Librarianship*, 27:6 (2001)452:457.

Bolin, Mary K. "Catalog Design, Catalog Maintenance, Catalog Governance," *Library Collections, Acquisitions, and Technical Services*. 24: (2000)53-63.

Bolin, Mary K. "The Collegial Environment and the Functional Organization," *Journal of Library Administration*. 29:2(1999):49-61.

Bolin, Mary K. "Cataloging Environmental Impact Statements and Related Documents," *Resource Sharing and Library Networks*. 14:1(1999): 21-40.

Bolin, Mary K. "Waiting for the Electrician or Someone like Him: Client-Centered Technical Services" Proceedings of the Association of College and Research Libraries National Conference, Nashville, April 1997.

< http://www.ala.org/Content/ContentGroups/ACRL1/Nashville_1997_Papers/Bolin.htm >

Grants:

Electronic Theses and Dissertations. Seed Grant from University of Idaho Office of Research, 2002. Co PI with Gail Z. Eckwright. \$6500.

Interoperability of Metadata for Thematic Research Collections. Grant to the Center for Digital Research in the Humanities from the Institute of Museum and Library Services 2005-2008.

Project Team Member.

http://unlcms.unl.edu/cas/center-for-digital-research-in-the-humanities/projects/pages/iom_files/IOM_Final_Narrative_Report_July_2008.pdf

Other Publications:

Library Philosophy & Practice. Peer-reviewed electronic library science journal. Mary K. Bolin and Gail Z. Eckwright, founders and editors. <http://digitalcommons.unl.edu/libphilprac>

Library Science C415, Technical Services for Small Libraries, through Independent Study in Idaho. 2004.

Library Science C416, Library Services to Rural Libraries, through Independent Study in Idaho. 2004.

Library Science C422, Use of the School Library, through Independent Study in Idaho. Revised 1999.

Library Science C425/525, Organization and Management of Small Libraries through Independent Study in Idaho. Revised 2005.

Library Science C427, Library or Media Center Practicum, Independent Study in Idaho. Revised 1999.

Presentations:

Bolin, Mary K. "BibFrame and the MARC Cataloging Environment: an Introduction." Lyris Webinar, June 16, 2020. Recording available <http://lyrasis.adobeconnect.com/paufo9e0dm30/>

Pytlik Zillig, Brian, and Mary K. Bolin. "A Little More than Kin, a Little Less than Kind": the Semantic Field Grace in Early Modern English" Poster Session, Text Encoding Initiative Meeting, Graz, Austria, September 18, 2019

Pytlik Zillig, Brian, and Mary K. Bolin. "Shakespeare and the Enumeration of Semantic Primes." Research Paper Presentation, Text Encoding Initiative Meeting, Tokyo, Japan, September 13, 2018

"Library Philosophy and Practice: Creating and Maintaining an Open-Access E-Journal," Presentation to UNCL meeting (libraries at campuses of the University of Nebraska), Lincoln, Nebraska, March 20, 2008.

Presentation to "Thinking outside the Borders" participants on the open-access electronic journal Library Philosophy and Practice and its role in the international library community. University of Nebraska—Lincoln Libraries, October 24, 2007

Panel discussion on open access electronic journals. GWLA directors' meeting. Linda K. Hall Library, Kansas City, Missouri. April 2007.

"Waiting for the Electrician or Someone Like Him: Client-Centered Technical Services." Association of College and Research Libraries National Conference, Nashville, April 5, 1997.

" 'Some Problems of an Editor': the Idaho Librarian from Manuscript to Published Issue," Joint Annual Conference of the Idaho Library Association and the Washington Library Association, Spokane, Washington, April 6, 1995.

"Automating Idaho's Libraries: a Historical Perspective." Idaho Library Association Annual Conference, Idaho Falls, October 4, 1990.

"Small Library Management." Idaho Library Association Spring Conference, Lewiston, Idaho, May 13, 1989.

"No More 'Good Workers': Excellence and Academic Library Classified Staff." Association of College and Research Libraries 5th Annual Conference, Cincinnati, April 7, 1989.

"Communication to and from Cataloging Departments." Heads of Cataloging Discussion Group, American Library Association Midwinter Meeting, San Antonio, January 1988.

Discussion Leader at the LAMA (Library Administration and Management) Management Exchange. American Library Association Annual Meeting, San Francisco, June 1987.

"Staff Turnover." Heads of Cataloging Discussion Group, American Library Association Midwinter Meeting, Chicago, January 1987.

SERVICE

Reviewer. *Resources for College Libraries*. Linguistics bibliography, Spring 2020.

Peer reviewer for ALISE National Conference presentations, Spring 2020.

Peer reviewer for *Journal of Academic Librarianship*, 2012-

Peer reviewer for refereed journal, *Western Nigerian Journal of Medical Sciences* (<http://www.wnjms.org/>), a publication of Ladoke Akintola University of Technology (LAUTECH) Teaching Hospital, 2008-

PhD Reviewer for students from Aligarh Muslim University, India, 2010-

Outside reviewer for numerous library faculty promotion and tenure cases in US universities.

Faculty Committee Assignments:

University of Nebraska – Lincoln

Chair, University of Nebraska Libraries Faculty, 2018

Academic Rights and Responsibilities Committee, 2016- Chair, 2018-19

Faculty Senate – 2004-2008; 2008-2011

Faculty Senate Executive Committee 2004-2008

Academic Planning Committee (APC), 2005-2008; Vice Chair, 2006-2007; Chair, 2007-2008

Aesthetic Review Committee 2006-2007

APC monitor for academic program reviews: Department of Political Science, 2005; Department of Educational Psychology 2006

Chancellor's Committee on Wellness, 2007-

University of Idaho:

University of Idaho NASC Accreditation Committee: Chair, Standard 5, Library and Information Resources Subcommittee, 2002-2004.

Faculty Affairs Committee, 2001-2004; 1992-1995; Chair, 1994-1995

Committee on Committees, 2000-2003, Chair, 1991-92; 1995-96

Budget Liaison Committee, 1990-92, 1996-98; Chair, 1991-92; 1997-98

University of Idaho Presidential Search Committee, 1995-96

University of Idaho Provost Search Committee, 1996-97

University of Idaho Strategic Planning Task Force, 1997-98

Search Committee Chair, Lionel Hampton School of Music Director, 1998-99

Disability Affairs Committee, 1998-; Chair, 1999-2000

Faculty Council, Faculty-at-Large representative, 1989-92, 1995-98; Vice-Chair, 1991-92, 1995-96; Acting Chair, Spring 1996

Honors Program Committee, 1988-93; Chair, 1991-92

Student Employee Grievance Committee, 1988-89

Faculty-at-Large (Chair), 1987-89

Pacific Northwest Library Association:

Editor, *PNLA Quarterly*, December 2002-December 2013

Idaho Library Association:

Editor, *The Idaho Librarian*, September 1990-January 1996

Professional Memberships

American Library Association

ATTACHMENT 1

IMAGINATION: A Robust Image-based CAPTCHA Generation System

Ritendra Datta, Jia Li, and James Z. Wang

The Pennsylvania State University
University Park, PA 16802, USA

datta@cse.psu.edu, jiali@stat.psu.edu, jwang@ist.psu.edu

ABSTRACT

We propose IMAGINATION (IMAge Generation for INternet AuthenticaTION), a system for the generation of attack-resistant, user-friendly, image-based CAPTCHAs. In our system, we produce controlled distortions on randomly chosen images and present them to the user for annotation from a given list of words. The distortions are performed in a way that satisfies the incongruous requirements of low perceptual degradation and high resistance to attack by content-based image retrieval systems. Word choices are carefully generated to avoid ambiguity as well as to avoid attacks based on the choices themselves. Preliminary results demonstrate the attack-resistance and user-friendliness of our system compared to text-based CAPTCHAs.

Categories and Subject Descriptors

K.6.5 [Management of Computing and Information Systems]: Security and Protection.

General Terms

Verification, Security.

Keywords

Automated Turing test, CAPTCHA, image retrieval.

1. INTRODUCTION

A way to tell apart a human from a computer by a test is known as a Turing Test [10]. When a computer program is able to generate such tests and evaluate the result, it is known as a CAPTCHA (Completely Automated Public test to Tell Computers and Humans Apart) [1]. In the past, Websites have often been *attacked* by malicious programs that register for service on massive scale. Programs can be written to automatically consume large amount of Web resources or bias results in on-line voting. This has driven researchers to the idea of CAPTCHA-based

security, to ensure that such attacks are not possible without human intervention, which in turn makes them ineffective. CAPTCHA-based security protocols have also been proposed for related issues, e.g., countering Distributed Denial-of-Service (DDoS) attacks on Web servers [6]. A CAPTCHA acts as a security mechanism by requiring a correct answer to a question which only a human can answer any better than a random guess. Humans have speed limitation and hence cannot replicate the impact of an automated program. Thus the basic requirement of a CAPTCHA is that computer programs must be slower than humans in responding correctly. To that purpose, the *semantic gap* [9] between human understanding and the current level of machine intelligence can be exploited. Most current CAPTCHAs are text-based.

Commercial text-based CAPTCHAs have been broken using object-recognition techniques [7], with accuracies of up to 99% on EZ-Gimpy. This reduces the reliability of security protocols based on text-based CAPTCHAs. There have been attempts to make these systems harder to break by systematically adding noise and distortion, but that often makes them hard for humans to decipher as well. Image-based CAPTCHAs such as [1, 3, 8] have been proposed as alternatives to the text media. More robust and user-friendly systems can be developed. State-of-the-art content-based image retrieval (CBIR) and annotation techniques have shown great promise at automatically finding semantically similar images or naming them, both of which allow means of attacking image-based CAPTCHAs. User-friendliness of the systems are potentially compromised when repeated responses are required [3] or deformed face images are shown [8].

One solution is to randomly distort the images before presenting them. However, current image matching techniques are robust to various kinds of distortions, and hence a systematic distortion is required. Here, we present IMAGINATION, a system for generating user-friendly image-based CAPTCHAs robust against automated attacks. Given a database of images of simple concepts, a two-step user-interface allows quick testing for humans while being expensive for machines. Controlled composite distortions on the images maintain visual clarity for recognition by humans while making the same difficult for automated systems.

Requiring the user to type in the annotation may lead to problems like misspelling and polysemy [3]. In our system, we present to the user a set of word choices, and the user must choose the most suitable image descriptor. A problem with generating word choices is that we might end up having,

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

MM'05, November 6–11, 2005, Singapore.

Copyright 2005 ACM 1-59593-044-2/05/0011 ...\$5.00.

say, the word “dog” and the word “wolf” in the list, and this may cause ambiguity in labeling. To avoid this problem, we propose a WordNet-based [5] algorithm to generate a semantically non-overlapping set of word choices while preventing *odd-one-out* attacks using the choices themselves. Because the number of choices are limited, the location of the mouse-click on the composite image acts as additional user input, and together with the annotation, it forms the two-step mechanism to reduce the rate of random attacks.

2. THE IMAGINATION SYSTEM

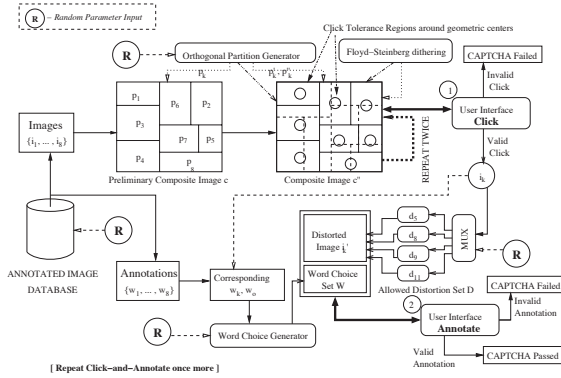


Figure 1: The IMAGINATION system architecture.

A reason for naming our system IMAGINATION is that it aims to exploit human imagination power gained through exposure/experience, allowing interpretation of pictures amidst distortion/clutter. The overall system architecture is shown in Fig. 1. We have a two-round *click-and-annotate* process in which a user needs to click on the interface 4 times in all. The system presents the user with a set of 8 images tiled to form a single *composite image*. The user must then select an image she wants to annotate by clicking near its geometric center. If the location of the click is near one of the centers, a controlled distortion is performed on the selected image and displayed along with a set of *word choices* pertaining to it, and the user must choose the appropriate one. If the click is not near any of the centers or the choice is invalid, the test restarts. Otherwise, this *click-and-annotate* process is repeated one more time, passing which the CAPTCHA is considered cleared. The reason for having the *click* phase is that the word choices are limited, making random attack rate fairly high. Instead of having numerous rounds of *annotate*, user clicks tend to make the system more user-friendly, while decreasing the attack rate.

The first step is the *composite image* generation. Given an annotated database of images I consisting of simple concepts and objects, the system randomly selects a set of 8 images $\{i_1, \dots, i_8\}$ with their corresponding annotations $\{w_1, \dots, w_8\}$. A rectangular region is divided into 8 random orthogonal partitions $\{p_1, \dots, p_8\}$ and by a one-to-one mapping $i_k \rightarrow p_k$, each image is placed into a partition, scaled as necessary, forming a *preliminary composite image c*. A two-stage *dithering* using the *Floyd-Steinberg error-diffusion algorithm* is then performed. The image c is randomly divided into two different sets of 8 orthogonal partitions $\{p'_1, \dots, p'_8\}$ and $\{p''_1, \dots, p''_8\}$, and dithering is applied on these two sets sequentially, forming

the required *composite image c''*. Dithering parameters that are varied independently over each partition include the base colors used (18, randomly chosen in RGB space), resulting in different color gamuts, and the coefficients used for spreading the quantization error. The same ratio of coefficients $7/16$, $1/16$, $5/16$ and $3/16$ are used for neighboring pixels, but they are multiplied by a factor α_k , which is chosen randomly in the range of $0.5 - 1.5$. These steps ensure that the task of automatically determining the geometric centers of the images remain challenging, while human imagination continues to steer rough identification. The difficulty in automated detection arises from the fact that partitioning and subsequent dithering cuts the original image tiling arbitrarily, making techniques such as edge/rectangle detection generate many false boundaries (see example in Fig. 2 for an idea). Let the location of the actual user click be (X, Y) . Suppose the corner co-ordinates of the 8 images within the composite image be $\{(x_1^k, y_1^k, x_2^k, y_2^k), k = 1, \dots, 8\}$. The user's click is considered valid if $\min_k \left\{ \left(X - \frac{x_1^k + x_2^k}{2} \right)^2 + \left(Y - \frac{y_1^k + y_2^k}{2} \right)^2 \right\} \leq R^2$ where *tolerance* R is a constant determining the radius around the actual geometric centers of each image up to which this validity holds. Note that this parameter adjusts the wall between user-friendliness and reliability (larger tolerance R also means higher random attack rate).



Figure 2: Example composite image.

Suppose the response is valid and the minimum is achieved for image i_k . Then a randomly chosen *composite distortion* from among an *allowed distortion set D* is performed on i_k and displayed in its original size and aspect ratio. Based on the corresponding annotation w_k , a *word choice set W* is generated. Generation of D and W are described below.

2.1 Determining the Allowed Distortion Set

Images can be distorted in various ways. Our design of an *allowed distortion set D* requires the inclusion of distortions that maintains good visual clarity for recognition by humans while making automated recognition hard. CAPTCHA requires that the annotated database and relevant code be publicly available, for added security. If undistorted images from the database were presented as CAPTCHAs, attacks would be trivial. Previous systems proposed [3] are liable to such attacks. If the images are randomly distorted before being presented to the user [1], it may still be possible to perform attacks using computer vision techniques such as affine/scale invariant features and CBIR.

We aim at building image-based CAPTCHAs secure

against such attacks. Certain assumptions about possible attack strategies are needed in order to design attack-resistant distortions. Here, we assume that the only feasible way is to use CBIR to perform inexact matches between the distorted image and the set of images in the database, and use the label associated with an appropriately matched one for attack. This assumption is reasonable since attack strategy needs to work on the entire image database in real-time in order to be effective, and image retrieval usually scales better than other techniques. Suppose $d(i_k)$ indicates the application of distortion d on image i_k , and $S_p(i_j, i_k)$ denotes the similarity measure between images i_j and i_k using image retrieval system S_p . Considering the worst-case scenario where the attacker has access to the database I , the CBIR system S_p , and the distortion algorithms in D , a good attack strategy can be as follows: The attacker studies the *distribution* of the distances between (1) a distorted image and its original, $f_1(x)$, and (2) a distorted image and all other images in I , $f_2(x)$. For a given distorted image $d(i_j)$, she can then compute $S_p(d(i_j), i_k) \forall i_k \in I$. If there are significant differences between $f_1(x)$ and $f_2(x)$, the attacker can exploit this to eliminate images in I that are unlikely to be i_j . One way to do this is to set a confidence interval $[a, b]$ at say 90% level around the mean of distribution f_1 and then eliminating all images i_k except those with $a \leq S_p(d(i_j), i_k) \leq b$. With N images contained in I , and a random guess, $P(\text{Attack}) = N^{-1}$, while after elimination,

$$P(\text{Attack}) = \left(0.9N \int_a^b f_2(x) dx \right)^{-1}.$$

This idea is illustrated in Fig. 3. Our goal is to counter such attacks by choosing distortions d that minimize $P(\text{Attack})$, i.e. maximize $\int_a^b f_2(x) dx$. Although $f_2(x)$ is dependent on $d(i_j)$, there is no easy way to control f_2 directly through a choice of d . Instead, we design D by choosing distortions d that give a value for $P(\text{Attack})$ below a chosen threshold T . In this way, we ensure that probabilistically, given distorted image $d(i_j)$ and all data/code, the attacker can identify the original image i_j in I (and hence successfully attack) with a probability of at most T . We found through experiments that while $f_2(x)$ tends to be a *wider* distribution, $f_1(x)$ is usually a *narrow* band with mean closer to the origin, and both are only slightly skewed from Gaussian distributions. Intuitively, under such circumstances, if $\delta = |f_1 - f_2|$, $P(\text{Attack})$ decreases as $\delta \rightarrow 0$ (see Fig. 3). One underlying assumption for our probabilistic criteria is that distributions $f_1(x)$ and $f_2(x)$ are invariant to the choice of i_j . Though this does not hold precisely, it does so for a majority of the i_j in I , allowing us the liberty to make the assumption to get a significantly simpler criteria.

For experiments, our choice of S_p is a state-of-the-art similarity measure (or image distance), the *Integrated Region Matching (IRM)* used in the SIMPLIcity system [11]. While other image comparison methods exist [9], IRM produces relatively fast (speed of attack is critical here) and accurate inexact matches. Note that the actual features or systems to be used by the attacker is *unknown*, but for the purpose of launching effective attacks, alternate choices seem unlikely. If there are better ways to attack the system, then these in turn improve the state-of-the-art in retrieving distorted images, and new sets of distortions need to be included in D . We have not considered attacks based on interest points or other such features.

Our experiments revealed that isolated distortions are

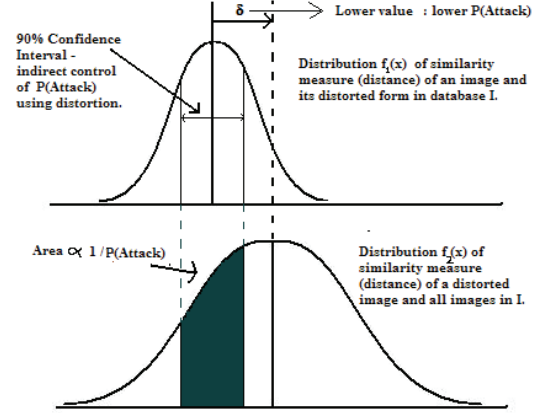


Figure 3: Criteria for including distortions into D .

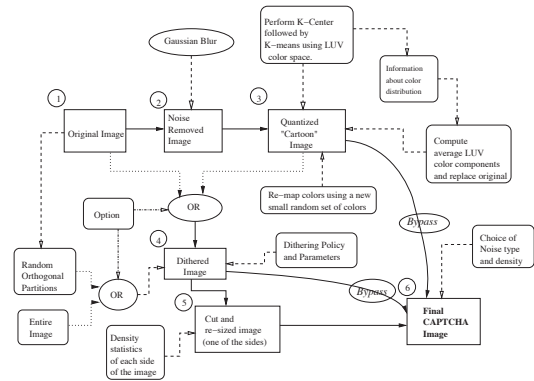


Figure 4: Framework for composite distortions.

insufficient in fooling the retrieval systems. Considering attack chances and visual clarity after distortion, we came up with a set of 11 *candidate composite distortions* $\{d_1, \dots, d_{11}\}$ along the framework shown in Fig. 4. Due to brevity of space, detailed descriptions are not possible. In short, each one is composed of a combination of *dithering*, *partitioning*, *quantization*, *noise addition*, *color re-mapping*, and *selective cut-and-resize*. Dithering seemed particularly suitable since clarity was retained while low-level feature extraction (and thus image matching) was affected. We applied the distortions to 300 Corel images and used IRM to calculate $f_1(x)$ and $f_2(x)$ for each d_k . Based on our criteria, a suitable threshold T , and a 90% confidence interval around $\bar{f}_1$, distortions d_5 , d_8 , d_9 and d_{11} were chosen as part of the *allowed distortion set D*. Note that we define here a formal procedure for choosing composite distortions, and select 4 acceptable ones out of a set of 11 ad-hoc distortions. Details of these distortions is not critical to the novelty of our work. Other distortions can be added to D by this procedure.

2.2 Determining the Word Choice Set

For word choice generation, factors related to image-based CAPTCHAs that have not been previously addressed are (1) it may be possible to remove ambiguity in labeling images (hence making annotation easier for humans) by the choices themselves, (2) the images might seem to have multiple valid labels (e.g. a tiger in a lake can be seen as “tiger” and “lake” as separate entities), and this may cause

ambiguity, and (3) the choices themselves may result in *odd-one-out* attacks if the correct choice is semantically different from all others. We propose an algorithm to generate the word choice set W containing unambiguous choices for the ease of users, while ensuring that word-based attacks are ineffective. For this we use a WordNet-based [5] semantic word similarity measure [4], denoted by $d(w_1, w_2)$ where w_1 and w_2 are English words. Given the correct annotation w_k (e.g. “tiger”) of image i_k , and optionally, other words W_o (e.g. {“lake”}) with the requirement of N_w choices, the algorithm for determining W is as follows:

1. Set $W \leftarrow \{w_k\} + W_o$, $t \leftarrow 1$.
2. Choose a word $w_l \notin W$ randomly from the database.
3. $flag = 0$.
4. For each word $w_i \in W$
If $d(w_k, w_i) < \theta$ then $flag = 1$.
5. If $flag = 1$ then go to step 2.
6. $W \leftarrow W + \{w_l\}$; $t \leftarrow t + 1$
7. If $t < N_w$ then go to step 2.
8. $W \leftarrow W - W_o$

The value of θ depends on what range of values the word similarity measure yields and can be determined empirically or based on user surveys (i.e. what values of θ causes ambiguity). Geometrically speaking, this method yields word choices like as if all the words lie beyond the boundaries of a (N_w) -dimensional *simplex* or *hyper-tetrahedron*.

3. RESULTS AND CONCLUSION

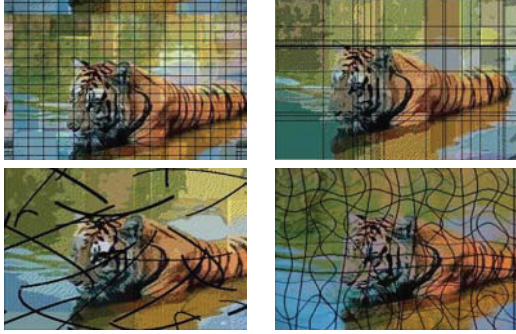


Figure 5: Clockwise from top-left: Distortion results using methods d_5 , d_8 , d_{11} , and d_9 .

Distorted images produced using the 4 chosen methods in D are shown in Fig. 5. Clearly, perceptual quality of the images have not deteriorated beyond recognition. User-friendliness of image-based CAPTCHAs has been studied before [3]. Hence we conducted a user survey only on the ease of use of our *click-and-annotate* process. We chose 8 distorted images each of 8 different concepts from the Corel database, and arbitrarily chose 5 users and asked them to annotate the images (40 responses per concept). On an average, 95% were correct responses. Another survey was conducted on the ease of clicking near geometric centers in our *composite images*, using an 800×600 composite image consisting of 8 images ($R = 15$), yielding 90% accuracy in user clicks. An appropriate choice of threshold T in choosing distortion set D ensures that automated annotation is not noticeably better than a random guess among the N_w possible word choices. With $N_w = 15$, the random attack success rate for two rounds of *click-and-annotate* is thus $\left(\frac{8\pi R^2}{800 \times 600} \times \frac{1}{N_w}\right)^2$, or 0.000062%. This is significantly lower

than the attack rates of up to 99% on current text-based CAPTCHAs. Without the *click* phase, attack rate would still be pretty high at $1/N_w^2$ or 0.44%, which justifies the need for the *click* phase. Because cracking our proposed system will require solving two distinct hard AI problems, with our design being aimed at ensuring attack-resistance to state-of-the-art image matching, we do not expect this CAPTCHA to be broken to any sizable extent in the near future, unless there is considerable progress in image understanding technology. Our system generates distortions in less than 1 sec. on a 450 MHz Sun Ultra 60 Server. *Word choice set* takes about 20 sec. to generate using a Perl interface to WordNet (the algorithm makes iterative calls to the word similarity interface, which is slow), but that can be sped up easily using pre-processing.

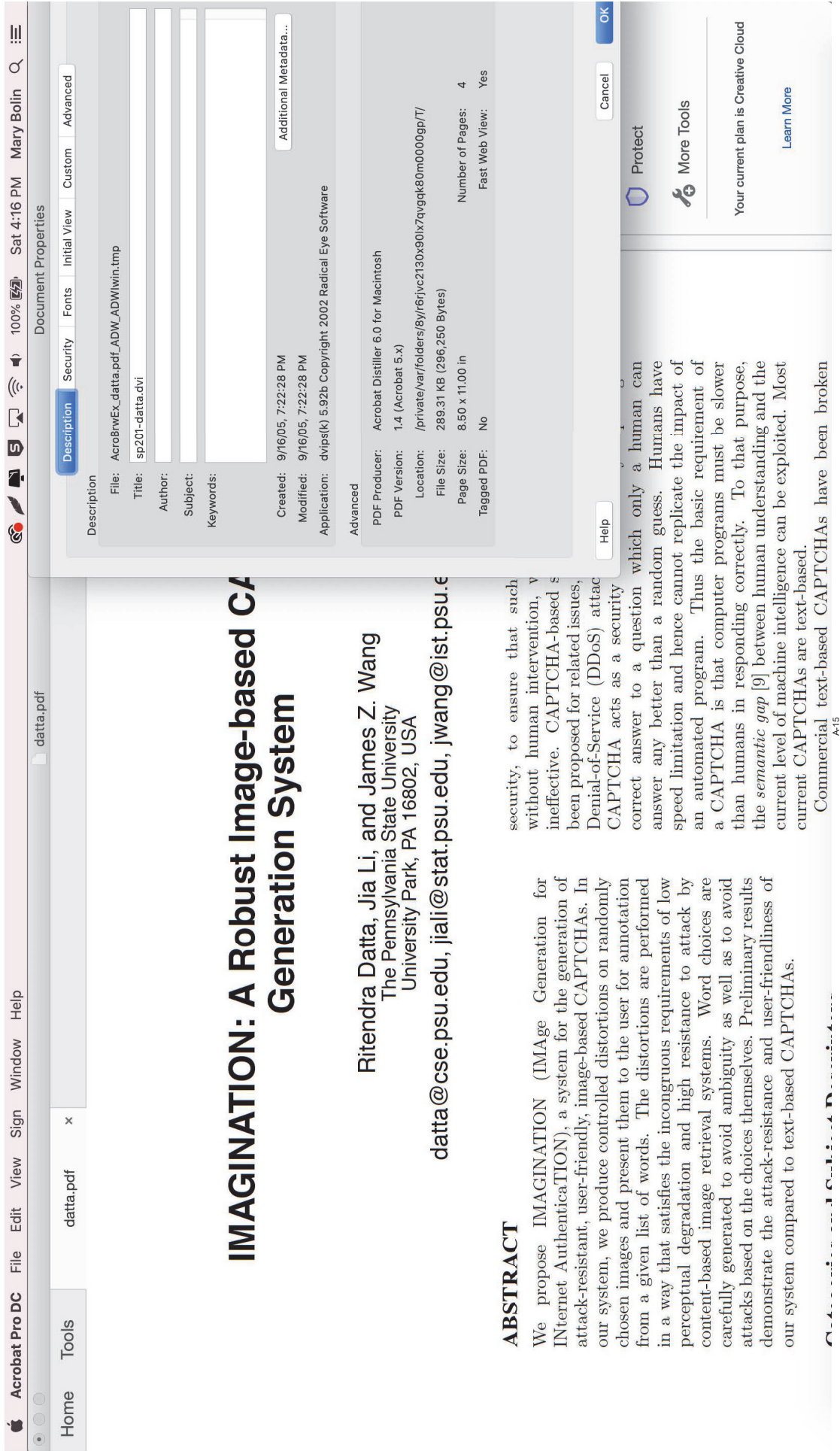
In conclusion, we have proposed a new CAPTCHAs generation system using a considerable amount of pseudo-randomness. A novel word-choice generation algorithm is proposed that tackles issues related to user-friendliness and security. A formal method for choosing composite distortion for inclusion in the *allowed distortions set* is proposed, and four such distortions are obtained through experimentation. Under certain assumptions about the *best possible* feasible attack strategy, our system is much more secure compared to text-based CAPTCHAs. User-friendliness has been carefully considered in our design, and preliminary results suggest that a simple interface and just four mouse-clicks make it favorable. In the future, we plan to carry out large-scale user-studies on the ease of use, build a Web interface to the IMAGINATION system, and generate greater attack-resistance by considering other possible attack strategies such as interest points, scale/affine invariants, and other object-recognition techniques.

Acknowledgments: The material is based upon work supported by the NSF (IIS-0219272 and IIS-0347148). Riemann.ist.psu.edu provides more information.

4. REFERENCES

- [1] L. von Ahn et al., “Telling Humans and Computers Apart (Automatically) or How Lazy Cryptographers do AI,” *Comm. of the ACM*, 47(2):57-60, 2004.
- [2] “The CAPTCHA Project,” <http://www.captcha.net>.
- [3] M. Chew et al., “Image Recognition CAPTCHAs,” *Proc. 7th Info. Security Conf.*, 2004.
- [4] J. J. Jiang et al., “Semantic Similarity Based on Corpus Statistics and Lexical Taxonomy,” *Proc. Intl. Conf. Research in Computational Linguistics*, 1997.
- [5] G. Miller, “WordNet: A Lexical Database for English,” *Comm. of the ACM*, 38(11):39-41, 1995.
- [6] W. G. Morein et al., “Using Graphic Turing Tests To Counter Automated DDoS Attacks Against Web Servers,” *Proc. ACM Conf. Computer and Comm. Security*, 2003.
- [7] G. Mori et al., “Recognizing Objects in Adversarial Clutter: Breaking a Visual CAPTCHA,” *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 2003.
- [8] Y. Rui et al., “ARTiFACIAL: Automated Reverse Turing Test using FACIAL features,” *Multimedia Systems*, 9(6):493-502, 2004.
- [9] A. W. M. Smeulders et al., “Content-Based Image Retrieval at the End of the Early Years,” *IEEE Trans. Pattern Analysis and Machine Intell.*, 22(12):1349-1380, 2000.
- [10] A. Turing, “Computing Machinery and Intelligence,” *Mind*, 59(236):433-460, 1950.
- [11] J.Z. Wang et al., “SIMPLcity: Semantics-Sensitive Integrated Matching for Picture Libraries,” *IEEE Trans. Pattern Analysis and Machine Intell.*, 23(9):947-963, 2001.

ATTACHMENT 2



IMAGINATION: A Robust Image-based CAPTCHA Generation System

Ritendra Datta, Jia Li, and James Z. Wang
The Pennsylvania State University
University Park, PA 16802, USA
datta@cse.psu.edu, jiali@stat.psu.edu, jwang@ist.psu.edu

ABSTRACT

We propose IMAGINATION (IMAGE Generation for INternet AuthenticaTION), a system for the generation of attack-resistant, user-friendly, image-based CAPTCHAs. In our system, we produce controlled distortions on randomly chosen images and present them to the user for annotation from a given list of words. The distortions are performed in a way that satisfies the incongruous requirements of low perceptual degradation and high resistance to attack by content-based image retrieval systems. Word choices are carefully generated to avoid ambiguity as well as to avoid attacks based on the choices themselves. Preliminary results demonstrate the attack-resistance and user-friendliness of our system compared to text-based CAPTCHAs.

security, to ensure that such without human intervention, v ineffective. CAPTCHA-based s been proposed for related issues, Denial-of-Service (DDoS) attac CAPTCHA acts as a security correct answer to a question which only a human can answer any better than a random guess. Humans have speed limitation and hence cannot replicate the impact of an automated program. Thus the basic requirement of a CAPTCHA is that computer programs must be slower than humans in responding correctly. To that purpose, the *semantic gap* [9] between human understanding and the current level of machine intelligence can be exploited. Most current CAPTCHAs are text-based.

Commercial text-based CAPTCHAs have been broken

A-15

ATTACHMENT 3

WorldCat: ACM Multimedia 2005

OCLC 83979463									
No holdings in LDL - 18 other holdings No LDL holdings in GLIMIR cluster; 18 other holdings in GLIMIR cluster									
Books	Rec Stat	c	Entered	20070216	Replaced	20181115143511.3			
Type	a	ELvl	I	Srce	d	Audn		Ctrl	
BLvl	m	Form	o	Conf	1	Biog		MRec	
		Cont	b	GPub		LitF	0	Indx	1
Desc	a	Ills	a	Fest	0	DtSt	s	Dates	2005.

007 c b r d m e n f u

040 GAT ‡b eng ‡c GAT ‡d C\$A ‡d OCLCQ ‡d OCLCO ‡d OCLCF ‡d COO ‡d OCLCO ‡d OCLCQ ‡d OCL ‡d OCLCO
‡d OCLCQ ‡d VT2 ‡d WYU

020 ±z 1595930442020 [tz 9781595930446](#)

050 4 QA76.575 [tb](#) .A36 2005c Online only

090

049 I D I I

1112 ACM International Conference on Multimedia [in](#) (13th : [td](#) 2005 : [tc](#) Singapore)

24510 ACM Multimedia 2005 : [fb](#) proceedings of the 13th ACM International Conference on Multimedia, November 6-11, 2005. Singapore / [fc](#) sponsored by the ACM special interest groups SIGMM and SIGGRAPH.

24630 Multimedia 2005

24630 Proceedings of the 13th ACM International Conference on Multimedia

2463 Proceedings of the 13th Annual ACM International Conference on Multimedia

2463 International Multimedia Conference

2461 **±i** Also known as: **±a** Multimedia '05

260 New York, N.Y. : **ib** Association for Computing Machinery, **ic** ©2005.

300 1 online resource (xxiii, 1088 pages) : [4b](#) illustrations

```
336 text ±b txt ±2 rdacontent
```

337 computer ±b c ±2 rdamedia

338 online resource $\pm b$ cr ± 2 rdacarrier

500 Title from PDF title page (ACM Digital Library, viewed Aug. 27, 2009).

500 "ACM order number 433051"--Title page verso.

504 Includes bibliographical references and author index.

500 "ACM order number 433051"--Page ii.

650 0 Multimedia systems ±v Congresses.

650 0 Information storage and retrieval systems [tv](#) Congresses.

650_7 Information storage and retrieval systems. ± 2 fast ± 0 (OCOLC)fst00972781

650_7 Multimedia systems. ± 2 fast ± 0 (OCoLC)fst01028920

655_7 Conference papers and proceedings. ± 2 fast ± 0 (OCoLC)fst01423772

7102 ACM Special Interest Group on Multimedia.

7102 SIGGRAPH.

77608 [📖](#) Print version: [📖](#) ACM International Conference on Multimedia (13th : 2005 : Singapore). [📖](#) ACM Multimedia 2005 : proceedings of the 13th ACM International Conference on Multimedia : November 6-11, 2005, Singapore. [📖](#) New York : ACM, 2005 [📖](#) 9781595930446

85640 [fu http://portal.acm.org/toc.cfm?](http://portal.acm.org/toc.cfm?)

id=1101149&coll=portal&dl=ACM&type=proceeding&idx=SERIES316&part=series&WantType=Proceedings&title=MM

85640 [tu https://dl.acm.org/citation.cfm?id=1101149](https://dl.acm.org/citation.cfm?id=1101149)

85640 **⚡** VIEW FULL TEXT **⚡** <http://VH7QX3XE2P.search.serialssolutions.com/?V=1.0&L=VH7QX3XE2P&S=JCs&C=TC0001139182&T=marc&tab=BOOKS>

0291 AU@ ±b 000053037201

0291_NZ1 ~~14~~ 14779024

0291 AU@ ±b 000061403382

006 fields for Computer Files Computer File						
Audn	Form o	File d	GPub			
Action Status	Delete Holdings	Export	Label	Produce	Update Holdings	Validate

ATTACHMENT 4

CAPTCHA: Using Hard AI Problems for Security

Luis von Ahn¹, Manuel Blum¹, Nicholas J. Hopper¹, and John Langford²

¹ Computer Science Dept., Carnegie Mellon University, Pittsburgh PA 15213, USA

² IBM T.J. Watson Research Center, Yorktown Heights NY 10598, USA

Abstract. We introduce CAPTCHA, an automated test that humans can pass, but current computer programs can't pass: any program that has high success over a CAPTCHA can be used to solve an unsolved Artificial Intelligence (AI) problem. We provide several novel constructions of CAPTCHAs. Since CAPTCHAs have many applications in practical security, our approach introduces a new class of hard problems that can be exploited for security purposes. Much like research in cryptography has had a positive impact on algorithms for factoring and discrete log, we hope that the use of hard AI problems for security purposes allows us to advance the field of Artificial Intelligence. We introduce two families of AI problems that can be used to construct CAPTCHAs and we show that solutions to such problems can be used for steganographic communication. CAPTCHAs based on these AI problem families, then, imply a win-win situation: either the problems remain unsolved and there is a way to differentiate humans from computers, or the problems are solved and there is a way to communicate covertly on some channels.

1 Introduction

A CAPTCHA is a program that can generate and grade tests that: (A) most humans can pass, but (B) current computer programs can't pass. Such a program can be used to differentiate humans from computers and has many applications for practical security, including (but not limited to):

- **Online Polls.** In November 1999, *slashdot.com* released an online poll asking which was the best graduate school in computer science (a dangerous question to ask over the web!). As is the case with most online polls, IP addresses of voters were recorded in order to prevent single users from voting more than once. However, students at Carnegie Mellon found a way to stuff the ballots by using programs that voted for CMU thousands of times. CMU's score started growing rapidly. The next day, students at MIT wrote their own voting program and the poll became a contest between voting "bots". MIT finished with 21,156 votes, Carnegie Mellon with 21,032 and every other school with less than 1,000. Can the result of any online poll be trusted? Not unless the poll requires that only humans can vote.

- **Free Email Services.** Several companies (Yahoo!, Microsoft, etc.) offer free email services, most of which suffer from a specific type of attack: “bots” that sign up for thousands of email accounts every minute. This situation can be improved by requiring users to prove they are human before they can get a free email account. Yahoo!, for instance, uses a CAPTCHA of our design to prevent bots from registering for accounts. Their CAPTCHA asks users to read a distorted word such as the one shown below (current computer programs are not as good as humans at reading distorted text).



Fig. 1. The Yahoo! CAPTCHA.

- **Search Engine Bots.** Some web sites don’t want to be indexed by search engines. There is an *html* tag to prevent search engine bots from reading web pages, but the tag doesn’t guarantee that bots won’t read the pages; it only serves to say “no bots, please”. Search engine bots, since they usually belong to large companies, respect web pages that don’t want to allow them in. However, in order to truly *guarantee* that bots won’t enter a web site, CAPTCHAs are needed.
- **Worms and Spam.** CAPTCHAs also offer a plausible solution against email worms and spam: only accept an email if you know there is a human behind the other computer. A few companies, such as *www.spamarrest.com* are already marketing this idea.
- **Preventing Dictionary Attacks.** Pinkas and Sander [11] have suggested using CAPTCHAs to prevent dictionary attacks in password systems. The idea is simple: prevent a computer from being able to iterate through the entire space of passwords by requiring a human to type the passwords.

The goals of this paper are to lay a solid theoretical foundation for CAPTCHAs, to introduce the concept to the cryptography community, and to present several novel constructions.

Lazy Cryptographers Doing AI

Note that from a mechanistic point of view, there is no way to *prove* that a program cannot pass a test which a human can pass, since there is a program — the human brain — which passes the test. All we can do is to present evidence that it’s hard to write a program that can pass the test. In this paper, we take an approach familiar to cryptographers: investigate state-of-the-art algorithmic developments having to do with some problem, assume that the adversary does

not have algorithms for that problem that are much better than the state-of-the-art algorithms, and then prove a reduction between passing a test and exceeding the performance of state-of-the-art algorithms. In the case of ordinary cryptography, it is assumed (for example) that the adversary cannot factor 1024-bit integers in any reasonable amount of time. In our case, we assume that the adversary cannot solve an Artificial Intelligence problem with higher accuracy than what's currently known to the AI community. This approach, if it achieves widespread adoption, has the beneficial side effect of inducing security researchers, as well as otherwise malicious programmers, to advance the field of AI (much like computational number theory has been advanced since the advent of modern cryptography).

A CAPTCHA is a cryptographic protocol whose underlying hardness assumption is based on an AI problem.

An important component of the success of modern cryptography is the practice of stating, very precisely and clearly, the assumptions under which cryptographic protocols are secure. This allows the rest of the community to evaluate the assumptions and to attempt to break them. In the case of Artificial Intelligence, it's rare for problems to be precisely stated, but using them for security purposes forces protocol designers to do so. We believe that precisely stating unsolved AI problems can accelerate the development of Artificial Intelligence: most AI problems that have been precisely stated and publicized have eventually been solved (take chess as an example). For this reason it makes practical sense for AI problems that are used for security purposes to also be *useful*. If the underlying AI problem is useful, a CAPTCHA implies a win-win situation: either the CAPTCHA is not broken and there is a way to differentiate humans from computers, or the CAPTCHA is broken and a useful AI problem is solved. Such is not the case for most other cryptographic assumptions: the primary reason algorithms for factoring large numbers are useful is because factoring has applications in cryptanalysis.

In this paper we will present constructions of CAPTCHAs based on certain AI problems and we will show that solving the CAPTCHAs implies solving the AI problems. The AI problems we chose have several applications, and we will show that solutions to them can be used, among other things, for steganographic communication (see Section 5).

Related Work

The first mention of ideas related to “Automated Turing Tests” seems to appear in an unpublished manuscript by Moni Naor [10]. This excellent manuscript contains some of the crucial notions and intuitions, but gives no proposal for an Automated Turing Test, nor a formal definition. The first practical example of an Automated Turing Test was the system developed by Altavista [8] to prevent “bots” from automatically registering web pages. Their system was based on the difficulty of reading slightly distorted characters and worked well in practice,

but was only meant to defeat off-the-shelf Optical Character Recognition (OCR) technology. (Coates et al [5], inspired by our work, and Xu et al [14] developed similar systems and provided more concrete analyses.) In 2000 [1], we introduced the notion of a CAPTCHA as well as several practical proposals for Automated Turing Tests.

This paper is the first to conduct a rigorous investigation of Automated Turing Tests and to address the issue of *proving* that it is difficult to write a computer program that can pass the tests. This, in turn, leads to a discussion of using AI problems for security purposes, which has never appeared in the literature. We also introduce the first Automated Turing Tests not based on the difficulty of Optical Character Recognition. A related general interest paper [2] has been accepted by *Communications of the ACM*. That paper reports on our work, without formalizing the notions or providing security guarantees.

2 Definitions and Notation

Let $\mathcal{C}$ be a probability distribution. We use $[\mathcal{C}]$ to denote the support of $\mathcal{C}$. If $P(\cdot)$ is a probabilistic program, we will denote by $P_r(\cdot)$ the deterministic program that results when P uses random coins r .

Let (P, V) be a pair of probabilistic interacting programs. We denote the output of V after the interaction between P and V with random coins u_1 and u_2 , assuming this interaction terminates, by $\langle P_{u_1}, V_{u_2} \rangle$ (the subscripts are omitted in case the programs are deterministic). A program V is called a *test* if for all P and u_1, u_2 , the interaction between P_{u_1} and V_{u_2} terminates and $\langle P_{u_1}, V_{u_2} \rangle \in \{\text{accept}, \text{reject}\}$. We call V the *verifier* or *tester* and any P which interacts with V the *prover*.

Definition 1. Define the *success* of an entity A over a test V by

$$\text{Succ}_A^V = \Pr_{r, r'}[\langle A_r, V_{r'} \rangle = \text{accept}].$$

We assume that A can have precise knowledge of how V works; the only piece of information that A can't know is r' , the internal randomness of V .

CAPTCHA

Intuitively, a CAPTCHA is a test V over which most humans have success close to 1, and for which *it is hard to write a computer program that has high success over V* . We will say that it is hard to write a computer program that has high success over V if *any program that has high success over V can be used to solve a hard AI problem*.

Definition 2. A test V is said to be (α, β) -human executable if at least an α portion of the human population has success greater than β over V .

Notice that a statement of the form “ V is (α, β) -human executable” can only be proven empirically. Also, the success of different groups of humans might depend on their origin language or sensory disabilities: color-blind individuals, for instance, might have low success on tests that require the differentiation of colors.

Definition 3. An *AI problem* is a triple $\mathcal{P} = (S, D, f)$, where S is a set of problem instances, D is a probability distribution over the problem set S , and $f : S \rightarrow \{0, 1\}^*$ answers the instances. Let $\delta \in (0, 1]$. We require that for an $\alpha > 0$ fraction of the humans H , $\Pr_{x \leftarrow D}[H(x) = f(x)] > \delta$.

Definition 4. An AI problem $\mathcal{P}$ is said to be (δ, τ) -solved if there exists a program A , running in time at most τ on any input from S , such that

$$\Pr_{x \leftarrow D, r}[A_r(x) = f(x)] \geq \delta.$$

(A is said to be a (δ, τ) solution to $\mathcal{P}$.) $\mathcal{P}$ is said to be a (δ, τ) -hard AI problem if no current program is a (δ, τ) solution to $\mathcal{P}$, and the AI community agrees it is hard to find such a solution.

Definition 5. A (α, β, η) -CAPTCHA is a test V that is (α, β) -human executable, and which has the following property:

There exists a (δ, τ) -hard AI problem $\mathcal{P}$ and a program A , such that if B has success greater than η over V then A^B is a (δ, τ) solution to $\mathcal{P}$. (Here A^B is defined to take into account B ’s running time too.)

We stress that V should be a *program* whose code is publicly available.

Remarks

1. The definition of an AI problem as a triple (S, D, f) should not be inspected with a philosophical eye. We are not trying to capture all the problems that fall under the umbrella of Artificial Intelligence. We want the definition to be easy to understand, we want some AI problems to be captured by it, and we want the AI community to agree that these are indeed hard AI problems. More complex definitions can be substituted for Definition 3 and the rest of the paper remains unaffected.
2. A crucial characteristic of an AI problem is that a certain fraction of the human population be able to solve it. Notice that we don’t impose a limit on how long it would take humans to solve the problem. All that we require is that some humans be able to solve it (even if we have to assume they will live hundreds of years to do so). The case is not the same for CAPTCHAs. Although our definition says nothing about how long it should take a human to solve a CAPTCHA, it is preferable for humans to be able to solve CAPTCHAs in a very short time. CAPTCHAs which take a long time for humans to solve are probably useless for all practical purposes.

AI Problems as Security Primitives

Notice that we define *hard* in terms of the consensus of a community: an AI problem is said to be hard if the people working on it agree that it's hard. This notion should not be surprising to cryptographers: the security of most modern cryptosystems is based on assumptions agreed upon by the community (e.g., we *assume* that 1024-bit integers can't be factored). The concept of a hard AI problem as a foundational assumption, of course, is more questionable than $P \neq NP$, since many people in the AI community agree that all hard AI problems are eventually going to be solved. However, hard AI problems may be a more reasonable assumption than the hardness of factoring, given the possibility of constructing a quantum computer. Moreover, even if factoring is shown to be hard in an asymptotic sense, picking a concrete value for the security parameter usually means making an assumption about current factoring algorithms: we only *assume* that current factoring algorithms that run in current computers can't factor 1024-bit integers. In the same way that AI researchers believe that all AI problems will be solved eventually, we believe that at some point we will have the computational power and algorithmic ability to factor 1024-bit integers. (Shamir and Tromer [13], for instance, have proposed a machine that could factor 1024-bit integers; the machine would cost about ten million dollars in materials.)

An important difference between popular cryptographic primitives and AI problems is the notion of a security parameter. If we believe that an adversary can factor 1024-bit integers, we can use 2048-bit integers instead. No such concept exists in hard AI problems. AI problems, as we have defined them, do not deal with asymptotics. However, as long as there is a small gap between human and computer ability with respect to some problem, this problem can potentially be used as a primitive for security: rather than asking the prover to solve the problem once, we can ask it to solve the problem twice. If the prover gets good at solving the problem twice, we can ask it to solve the problem three times, etc.

There is an additional factor that simplifies the use of hard AI problems as security primitives. Most applications of CAPTCHAs require the tests to be answered within a short time after they are presented. If a new program solves the hard AI problems that are currently used, then a different set of problems can be used, and the new program cannot affect the security of applications that were run before it was developed. Compare this to encryption schemes: in many applications the information that is encrypted must remain confidential for years, and therefore the underlying problem must be hard against programs that run for a long time, and against programs that will be developed in the future.¹

We also note that not all hard AI problems can be used to construct a CAPTCHA. In order for an AI problem to be useful for security purposes, there needs to be an *automated* way to generate problem instances along with their solution. The case is similar for computational problems: not all hard computational problems yield cryptographic primitives.

¹ We thank one of our anonymous Eurocrypt reviewers for pointing this out.

Who Knows What?

Our definitions imply that an adversary attempting to write a program that has high success over a CAPTCHA knows exactly how the CAPTCHA works. The only piece of information that is hidden from the adversary is a small amount of randomness that the verifier uses in each interaction.

This choice greatly affects the nature of our definitions and makes the problem of creating CAPTCHAs more challenging. Imagine an Automated Turing Test that owns a large secret book written in English and to test an entity A it either picks a paragraph from its secret book or generates a paragraph using the best known text-generation algorithm, and then asks A whether the paragraph makes sense (the best text-generation algorithms cannot produce an entire paragraph that would make sense to a human being). Such an Automated Turing Test might be able to distinguish humans from computers (it is usually the case that the best text-generation algorithms and the best algorithms that try to determine whether something makes sense are tightly related). However, this test cannot be a CAPTCHA: an adversary with knowledge of the secret book could achieve high success against this test without advancing the algorithmic state of the art. *We do not allow CAPTCHAs to base their security in the secrecy of a database or a piece of code.*

Gap Amplification

We stress that *any* positive gap between the success of humans and current computer programs against a CAPTCHA can be amplified to a gap arbitrarily close to 1 by *serial* repetition. The case for parallel repetition is more complicated and is addressed by Bellare, Impagliazzo and Naor in [3].

Let V be an (α, β, η) -CAPTCHA, and let V_k^m be the test that results by repeating V m times in series (with fresh new randomness each time) and accepting only if the prover passes V more than k times. Then for any $\epsilon > 0$ there exist m and k with $0 \leq k \leq m$ such that V_k^m is an $(\alpha, 1 - \epsilon, \epsilon)$ -CAPTCHA. In general, we will have $m = O(1/(\beta - \eta)^2 \ln(1/\epsilon))$ and sometimes much smaller. Since CAPTCHAs involve human use, it is desirable to find the smallest m possible. This can be done by solving the following optimization problem:

$$\min_m \left\{ \exists k : \sum_{i=k+1}^m \binom{m}{i} \beta^i (1 - \beta)^{m-i} \geq 1 - \epsilon \ \& \ \sum_{i=0}^k \binom{m}{i} \eta^i (1 - \eta)^{m-i} \leq \epsilon \right\}$$

Notice that amplifying a gap can roughly be thought of as increasing the security parameter of a CAPTCHA: if the best computer program now has success 0.10 against a given CAPTCHA (for example), then we can ask the prover to pass the CAPTCHA twice (in series) to reduce the best computer program's success probability to 0.01.

3 Two AI Problem Families

In this section we introduce two families of AI problems that can be used to construct CAPTCHAs. The section can be viewed as a precise statement of the kind of hardness assumptions that our cryptographic protocols are based on. We stress that solutions to the problems will also be shown to be useful.

For the purposes of this paper, we define an *image* as an $h \times w$ matrix (h for height and w for width) whose entries are *pixels*. A pixel is defined as a triple of integers (R, G, B) , where $0 \leq R, G, B \leq M$ for some constant M . An *image transformation* is a function that takes as input an image and outputs another image (not necessarily of the same width and height). Examples of image transformations include: turning an image into its black-and-white version, changing its size, etc.

Let $\mathcal{I}$ be a distribution on images and $\mathcal{T}$ be a distribution on image transformations. We assume for simplicity that if $i, i' \in [\mathcal{I}]$ and $i \neq i'$ then $T(i) \neq T'(i')$ for any $T, T' \in [\mathcal{T}]$. (Recall that $[\mathcal{I}]$ denotes the support of $\mathcal{I}$.)

Problem Family ($\mathcal{P1}$). Consider the following experiment: choose an image $i \leftarrow \mathcal{I}$ and choose a transformation $t \leftarrow \mathcal{T}$; output $t(i)$. $\mathcal{P1}_{\mathcal{I}, \mathcal{T}}$ consists of writing a program that takes $t(i)$ as input and outputs i (we assume that the program has precise knowledge of $\mathcal{T}$ and $\mathcal{I}$). More formally, let $S_{\mathcal{I}, \mathcal{T}} = \{t(i) : t \in [\mathcal{T}] \text{ and } i \in [\mathcal{I}]\}$, $D_{\mathcal{I}, \mathcal{T}}$ be the distribution on $S_{\mathcal{I}, \mathcal{T}}$ that results from performing the above experiment and $f_{\mathcal{I}, \mathcal{T}} : S_{\mathcal{I}, \mathcal{T}} \rightarrow [\mathcal{I}]$ be such that $f_{\mathcal{I}, \mathcal{T}}(t(i)) = i$. Then $\mathcal{P1}_{\mathcal{I}, \mathcal{T}} = (S_{\mathcal{I}, \mathcal{T}}, D_{\mathcal{I}, \mathcal{T}}, f_{\mathcal{I}, \mathcal{T}})$.

Problem Family ($\mathcal{P2}$). In addition to the distributions $\mathcal{I}$ and $\mathcal{T}$, let L be a finite set of “labels”. Let $\lambda : [\mathcal{I}] \rightarrow L$ compute the label of an image. The set of problem instances is $S_{\mathcal{I}, \mathcal{T}} = \{t(i) : t \in [\mathcal{T}] \text{ and } i \in [\mathcal{I}]\}$, and the distribution on instances $D_{\mathcal{I}, \mathcal{T}}$ is the one induced by choosing $i \leftarrow \mathcal{I}$ and $t \leftarrow \mathcal{T}$. Define $g_{\mathcal{I}, \mathcal{T}, \lambda}$ so that $g_{\mathcal{I}, \mathcal{T}, \lambda}(t(i)) = \lambda(i)$. Then $\mathcal{P2}_{\mathcal{I}, \mathcal{T}, \lambda} = (S_{\mathcal{I}, \mathcal{T}}, D_{\mathcal{I}, \mathcal{T}}, g_{\mathcal{I}, \mathcal{T}, \lambda})$ consists of writing a program that takes $t(i)$ as input and outputs $\lambda(i)$.

Remarks

1. Note that a (δ, τ) solution A to an instance of $\mathcal{P1}$ also yields a $(\delta', \tau + \tau')$ solution to an instance of $\mathcal{P2}$ (where $\delta' \geq \delta$ and $\tau' \leq \log |\mathcal{I}|$ is the time that it takes to compute λ), specifically, computing $\lambda(A(x))$. However, this may be unsatisfactory for small δ , and we might hope to do better by restricting to a smaller set of labels. Conversely, $\mathcal{P1}$ can be seen as a special case of $\mathcal{P2}$ with λ the identity function and $L = [\mathcal{I}]$. Formally, problem families $\mathcal{P1}$ and $\mathcal{P2}$ can be shown to be isomorphic. Nonetheless, it is useful to make a distinction here because in some applications it appears unnatural to talk about labels.
2. We stress that in all the instantiations of $\mathcal{P1}$ and $\mathcal{P2}$ that we consider, $\mathcal{I}, \mathcal{T}$ and, in the case of $\mathcal{P2}$, λ , will be such that humans have no problem solving $\mathcal{P1}_{\mathcal{I}, \mathcal{T}}$ and $\mathcal{P2}_{\mathcal{I}, \mathcal{T}, \lambda}$. That is, $[\mathcal{T}]$ is a set of transformations that humans

can easily undo in $[\mathcal{I}]$. Additionally, it has to be possible to perform all the transformations in $[\mathcal{T}]$ using current computer programs.

3. There is nothing specific to *images* about these problem definitions; any other space of objects which humans recognize under reasonable transformations (e.g., organized sounds such as music or speech, animations, *et cetera*) could be substituted without changing our results.
4. It is easy to build a $(\delta_{\mathcal{I}}, \tau_{\mathcal{I}})$ -solution to $\mathcal{P1}_{\mathcal{I}, \mathcal{T}}$, where $\delta_{\mathcal{I}} = \max\{\Pr_{j \leftarrow \mathcal{I}}[j = i] : i \in [\mathcal{I}]\}$ and $\tau_{\mathcal{I}}$ is the time that it takes to describe an element of $[\mathcal{I}]$, by always guessing the image with the highest probability in $\mathcal{I}$. Similarly, it is easy to build a $(\delta_{\mathcal{I}, \lambda}, \tau_{\mathcal{I}, \lambda})$ -solution to $\mathcal{P2}_{\mathcal{I}, \mathcal{T}, \lambda}$, where $\delta_{\mathcal{I}, \lambda} = \max\{\Pr_{j \leftarrow \mathcal{I}}[\lambda(j) = \lambda(i)] : i \in [\mathcal{I}]\}$ and $\tau_{\mathcal{I}, \lambda}$ is the time that it takes to describe a label in L . Therefore, we restrict our attention to finding solutions to $\mathcal{P1}_{\mathcal{I}, \mathcal{T}}$ where $\delta > \delta_{\mathcal{I}}$ and solutions to $\mathcal{P2}_{\mathcal{I}, \mathcal{T}, \lambda}$ where $\delta > \delta_{\mathcal{I}, \lambda}$.

Hard Problems in $\mathcal{P1}$ and $\mathcal{P2}$

We believe that $\mathcal{P1}$ and $\mathcal{P2}$ contain several hard problems. For example, the CAPTCHA shown in Section 1 (and other CAPTCHAs based on the difficulty of reading slightly distorted text) could be defeated using solutions to $\mathcal{P2}$. To see this, let W be a set of images of words in different fonts. All the images in W should be undistorted and contain exactly one word each. Let $\mathcal{I}_W$ be a distribution on W , let $\mathcal{T}_W$ be a distribution on image transformations, and let λ_W map an image to the word that is contained in it. A solution to $\mathcal{P2}_{\mathcal{I}_W, \mathcal{T}_W, \lambda_W}$ is a program that can defeat a CAPTCHA such as the one that Yahoo! uses (assuming $\mathcal{T}_W$ is the same set of transformations they use). So the problem of determining the word in a distorted image is an instantiation of $\mathcal{P2}$ (it can be easily seen to be an instantiation of $\mathcal{P1}$ too). Reading slightly distorted text has been an open problem in machine vision for quite some time. (For a good overview of the difficulties of reading slightly distorted text, see [12].)

But $\mathcal{P1}$ and $\mathcal{P2}$ are much more general, and reading slightly distorted text is a somewhat easy instance of these problems. In general it will not be the case that the problem is reduced to matching $26 * 2 + 10$ different characters (upper and lowercase letters plus the digits).

The hardness of problems in $\mathcal{P1}$ and $\mathcal{P2}$ mostly relies on $\mathcal{T}$. In particular, it should be computationally infeasible to enumerate all of the elements of $[\mathcal{T}]$, since $\mathcal{I}$ will normally be such that enumeration of $[\mathcal{I}]$ is feasible. Thus we are mainly interested in (δ, τ) solutions where $\tau \ll |[\mathcal{T}]|$, while $\tau > |[\mathcal{I}]|$ may sometimes be acceptable. In addition to the size of the transformation set, the character of the transformations is also important: it is necessary to defeat many simple checks such as color histogram comparisons, frequency domain checks, etc.

Since instantiations of $\mathcal{P1}$ and $\mathcal{P2}$ have never been precisely stated and published as challenges to the AI and security communities, there is no way to tell if they will withstand the test of time. For now we refer the reader to www.captcha.net for examples of $\mathcal{I}$'s and $\mathcal{T}$'s which are believed to be good candidates. Any instantiation of $\mathcal{P1}$ and $\mathcal{P2}$ for security purposes requires that the precise $\mathcal{I}$ and $\mathcal{T}$ be published and thoroughly described.

4 Two Families of CAPTCHAS

We now describe two families of CAPTCHAS whose security is based on the hardness of problems in $\mathcal{P}1$ and $\mathcal{P}2$. Notice that if $\mathcal{P}1_{\mathcal{I},\mathcal{T}}$ is (δ, τ) -hard then $\mathcal{P}1_{\mathcal{I},\mathcal{T}}$ can be used to construct a CAPTCHA trivially: the verifier simply gives the prover $t(i)$ and asks the prover to output i . According to our definition, this would be a perfectly valid CAPTCHA. However, it would also be a very impractical one: if $[\mathcal{I}]$ is large, then humans would take a long time to answer. The CAPTCHAS we present in this section can be quickly answered by humans. The first family of CAPTCHAS, MATCHA, is somewhat impractical, but the second family, PIX, is very practical and in fact several instantiations of it are already in use.

4.1 MATCHA

A MATCHA instance is described by a triple $M = (\mathcal{I}, \mathcal{T}, \tau)$, where $\mathcal{I}$ is a distribution on images and $\mathcal{T}$ is a distribution on image transformations that can be easily computed using current computer programs. MATCHA is a CAPTCHA with the following property: *any program that has high success over $M = (\mathcal{I}, \mathcal{T})$ can be used to solve $\mathcal{P}1_{\mathcal{I},\mathcal{T}}$.*

The MATCHA verifier starts by choosing a transformation $t \leftarrow \mathcal{T}$. It then flips a fair unbiased coin. If the result is heads, it picks $k \leftarrow \mathcal{I}$ and sets $(i, j) = (k, k)$. If the result is tails, it sets $j \leftarrow \mathcal{I}$ and $i \leftarrow U([\mathcal{I}] - \{j\})$ where $U(S)$ is the uniform distribution on the set S . The MATCHA verifier sends the prover $(i, t(j))$ and sets a timer to expire in time τ ; the prover responds with $res \in \{0, 1\}$. Informally, $res = 1$ means that $i = j$, while $res = 0$ means that $i \neq j$. If the verifier's timer expires before the prover responds, the verifier rejects. Otherwise, the verifier makes a decision based on the prover's response res and whether i is equal to j :

- If $i = j$ and $res = 1$, then MATCHA accepts.
- If $i = j$ and $res = 0$, then MATCHA rejects.
- If $i \neq j$ and $res = 1$, then MATCHA rejects.
- If $i \neq j$ and $res = 0$, then MATCHA plays another round.

In the last case, MATCHA starts over (with a fresh new set of random coins): it flips another fair unbiased coin, picks another pair of images (i, j) depending on the outcome of the coin, etc.

Remarks

1. It is quite easy to write a computer program that has success probability $1/2$ over MATCHA by simply answering with $res = 1$. For most applications, a distinguishing probability of $1/2$ is unacceptable. In order for MATCHA to be of practical use, the test has to be repeated several times.
2. Our description of MATCHA contains an obvious asymmetry: when $i = j$, MATCHA presents the prover with $(i, t(i))$ for $i \leftarrow \mathcal{I}$, and when $i \neq j$ MATCHA presents $(i, t(j))$, where i is chosen *uniformly* from the set $[\mathcal{I}] - \{j\}$. This gives the prover a simple strategy to gain advantage over M : if i seems to

come from $\mathcal{I}$, guess that $t(j)$ is a transformation of i ; otherwise guess that it isn't. The reason for the asymmetry is to make the proof of Lemma 1 easier to follow. We note that a stronger CAPTCHA can be built by choosing i from the distribution $\mathcal{I}$ restricted to the set $[\mathcal{I}] - \{j\}$.

3. The intuition for why MATCHA plays another round when $i \neq j$ and $res = 0$ is that we are trying to convert high success against MATCHA into high success in solving $\mathcal{P}1$; a program that solves $\mathcal{P}1$ by comparing $t(j)$ to every image in $[\mathcal{I}]$ will encounter that most of the images in $[\mathcal{I}]$ are different from $t(j)$.
4. In the following Lemma we assume that a program with high success over M always terminates with a response in time at most τ . Any program which does not satisfy this requirement can be rewritten into one which does, by stopping after τ time and sending the response 1, which never decreases the success probability. We also assume that the unit of time that MATCHA uses is the same as one computational step.

Lemma 1. *Any program that has success greater than η over $M = (\mathcal{I}, \mathcal{T}, \tau)$ can be used to $(\delta, \tau|[\mathcal{I}])$ -solve $\mathcal{P}1_{\mathcal{I}, \mathcal{T}}$, where*

$$\delta \geq \frac{\eta}{1 + 2|[\mathcal{I}](1 - \eta)}.$$

Proof. Let B be a program that runs in time at most τ and has success $\sigma_B \geq \eta$ over M . Using B we construct a program A^B that is a $(\delta, \tau|[\mathcal{I}])$ -solution to $\mathcal{P}1_{\mathcal{I}, \mathcal{T}}$.

The input to A^B will be an image, and the output will be another image. On input j , A^B will loop over the entire database of images of M (i.e., the set $[\mathcal{I}]$), each time feeding B the pair of images (i, j) , where $i \in [\mathcal{I}]$. Afterwards, A^B collects all the images $i \in [\mathcal{I}]$ on which B returned 1 (i.e., all the images in $[\mathcal{I}]$ that B thinks j is a transformation of). Call the set of these images S . If S is empty, then A^B returns an element chosen uniformly from $[\mathcal{I}]$. Otherwise, it picks an element ℓ of S uniformly at random.

We show that A^B is a $(\delta, \tau|[\mathcal{I}])$ -solution to $\mathcal{P}1$. Let $p_0 = \Pr_{\mathcal{T}, \mathcal{I}, r}[B_r(i, t(i)) = 0]$ and let $p_1 = \Pr_{\mathcal{T}, j \leftarrow \mathcal{I}, i, r}[B_r(i, t(j)) = 1]$. Note that

$$\Pr_{r, r'}[\langle M_r, B_{r'} \rangle = reject] = 1 - \sigma_B = \frac{p_0}{2} + \frac{p_1 + (1 - p_1)(1 - \sigma_B)}{2} = \frac{p_0 + p_1}{1 + p_1},$$

which gives $\sigma_B \leq 1 - p_0$ and $p_1 \leq 2(1 - \sigma_B)$. Hence:

$$\Pr_{\mathcal{T}, \mathcal{I}, r}[A_r^B(t(j)) = j] \geq \sum_{s=1}^n \Pr_{\mathcal{T}, \mathcal{I}, r}[A_r^B(t(j)) = j | |S| = s] \Pr_{\mathcal{T}, \mathcal{I}}[|S| = s] \quad (1)$$

$$= \sum_{s=1}^n \frac{1 - p_0}{s} \Pr_{\mathcal{T}, \mathcal{I}}[|S| = s] \quad (2)$$

$$\geq \frac{1 - p_0}{E_{\mathcal{T}, \mathcal{I}}[|S|]} \quad (3)$$

$$\geq \frac{1 - p_0}{1 + |\mathcal{I}|p_1} \quad (4)$$

$$\geq \frac{\sigma_B}{1 + 2|\mathcal{I}|(1 - \sigma_B)} \quad (5)$$

(2) follows by the definition of the procedure A^B , (3) follows by Jensen's inequality and the fact that $f(x) = 1/x$ is concave, (4) follows because

$$E_{\mathcal{T}, \mathcal{I}}[|S|] \leq 1 + \sum_{i \neq j} \Pr(B(i, t(j)) = 1)$$

and (5) follows by the inequalities for p_0 , p_1 and σ_B given above. This completes the proof.

Theorem 1. *If $\mathcal{P}1_{\mathcal{I}, \mathcal{T}}$ is $(\delta, \tau|\mathcal{I}|)$ -hard and $M = (\mathcal{I}, \mathcal{T}, \tau)$ is (α, β) -human executable, then M is a $(\alpha, \beta, \frac{(2-|\mathcal{I}|\delta)}{2\delta-|\mathcal{I}|})$ -CAPTCHA.*

4.2 PIX

An instance $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$ can sometimes be used almost directly as a CAPTCHA. For instance, if $\mathcal{I}$ is a distribution over images containing a single word and λ maps an image to the word contained in it, then $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$ can be used directly as a CAPTCHA. Similarly, if all the images in $[\mathcal{I}]$ are pictures of simple concrete objects and λ maps an image to the object that is contained in the image, then $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$ can be used as a CAPTCHA.

Formally, a PIX instance is a tuple $X = (\mathcal{I}, \mathcal{T}, L, \lambda, \tau)$. The PIX verifier works as follows. First, V draws $i \leftarrow \mathcal{I}$, and $t \leftarrow \mathcal{T}$. V then sends to P the message $(t(i), L)$, and sets a timer for τ . P responds with a label $l \in L$. V accepts if $l = \lambda(i)$ and its timer has not expired, and rejects otherwise.

Theorem 2. *If $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$ is (δ, τ) -hard and $X = (\mathcal{I}, \mathcal{T}, L, \lambda, \tau)$ is (α, β) -human executable, then X is a (α, β, δ) -CAPTCHA.*

Various instantiations of PIX are in use at major internet portals, like Yahoo! and Hotmail. Other less conventional ones, like ANIMAL-PIX, can be found at www.captcha.net. ANIMAL-PIX presents the prover with a distorted picture of a common animal (like the one shown in Figure 2) and asks it to choose between twenty different possibilities (monkey, horse, cow, *et cetera*).

5 An Application: Robust Image-Based Steganography

We detail a useful application of (δ, τ) -solutions to instantiations of $\mathcal{P}1$ and $\mathcal{P}2$ (other than reading slightly distorted text, which was mentioned before). We hope to convey by this application that our problems were not chosen just because they can create CAPTCHAs but because they in fact have applications related to security. Our problems also serve to illustrate that there is a need



Fig. 2. ANIMAL-PIX.

for better AI in security as well. Areas such a Digital Rights Management, for instance, could benefit from better AI: a program that can find slightly distorted versions of original songs or images on the world wide web would be a very useful tool for copyright owners.

There are many applications of solutions to $\mathcal{P}1$ and $\mathcal{P}2$ that we don't mention here. $\mathcal{P}1$, for instance, is interesting in its own right and a solution for the instantiation when $\mathcal{I}$ is a distribution on images of works of art would benefit museum curators, who often have to answer questions such as “what painting is this a photograph of?”

Robust Image-Based Steganography

Robust Steganography is concerned with the problem of covertly communicating messages on a public channel which is subject to modification by a restricted adversary. For example, Alice may have some distribution on images which she is allowed to draw from and send to Bob; she may wish to communicate additional information with these pictures, in such a way that anyone observing her communications can not detect this additional information. The situation may be complicated by an adversary who transforms all transmitted images in an effort to remove any hidden information. In this section we will show how to use (δ, τ) -solutions to instantiations of $\mathcal{P}1_{\mathcal{I}, \mathcal{T}}$ or $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$ to implement a secure robust steganographic protocol for image channels with distribution $\mathcal{I}$, when the adversary chooses transformations from $\mathcal{T}$. Note that if we require security for arbitrary $\mathcal{I}, \mathcal{T}$, we will require a (δ, τ) -solution to $\mathcal{P}1$ for arbitrary $\mathcal{I}, \mathcal{T}$; if no solution works for arbitrary $(\mathcal{I}, \mathcal{T})$ this implies the existence of specific $\mathcal{I}, \mathcal{T}$ for which $\mathcal{P}1$ is still hard. Thus either our stegosystem can be implemented by computers for arbitrary image channels or there is a (non-constructive) hard AI problem that can be used to construct a CAPTCHA.

The results of this subsection can be seen as providing an implementation of the “supraliminal channel” postulated by Craver [6]. Indeed, Craver's observation that the adversary's transformations should be restricted to those which do not significantly impact human interpretation of the images (because the adversary should not unduly burden “innocent” correspondents) is what leads to the applicability of our hard AI problems.

Steganography Definitions

Fix a distribution over images $\mathcal{I}$, and a set of keys $\mathcal{K}$. A *steganographic protocol* or *stegosystem* for $\mathcal{I}$ is a pair of efficient probabilistic algorithms (SE, SD) where $SE : \mathcal{K} \times \{0, 1\} \rightarrow [\mathcal{I}]^\ell$ and $SD : \mathcal{K} \times [\mathcal{I}]^\ell \rightarrow \{0, 1\}$, which have the additional property that $\Pr_{K, r, r'}[SD_r(K, SE_{r'}(K, \sigma)) \neq \sigma]$ is negligible (in ℓ and $|K|$) for any $\sigma \in \{0, 1\}$. We will describe a protocol for transmitting a single bit $\sigma \in \{0, 1\}$, but it is straightforward to extend our protocol and proofs by serial composition to any message in $\{0, 1\}^*$ with at most linear decrease in security.

Definition 6. A stegosystem is *steganographically secret* for $\mathcal{I}$ if the distributions $\{SE_r(K, \sigma) : K \leftarrow \mathcal{K}, r \leftarrow \{0, 1\}^*\}$ and $\mathcal{I}^\ell$ are computationally indistinguishable for any $\sigma \in \{0, 1\}$.

Steganographic secrecy ensures that an eavesdropper cannot distinguish traffic produced by SE from $\mathcal{I}$. Alice, however, is worried about a somewhat malicious adversary who transforms the images she transmits to Bob. This adversary is restricted by the fact that he must transform the images transmitted between many pairs of correspondents, and may not transform them in ways so that they are unrecognizable to humans, since he may not disrupt the communications of legitimate correspondents. Thus the adversary's actions, on seeing the image i , are restricted to selecting some transformation t according to a distribution $\mathcal{T}$, and replacing i by $t(i)$. Denote by $t_{1 \dots \ell} \leftarrow \mathcal{T}^\ell$ the action of independently selecting ℓ transformations according to $\mathcal{T}$, and denote by $t_{1 \dots \ell}(i_{1 \dots \ell})$ the action of element-wise applying ℓ transformations to ℓ images.

Definition 7. A stegosystem (SE, SD) is *steganographically robust against $\mathcal{T}$* if it is steganographically secret and

$$\Pr_{t_{1 \dots \ell} \leftarrow \mathcal{T}^\ell, r, r', K}[SD_r(K, t_{1 \dots \ell}(SE_r(K, \sigma))) \neq \sigma]$$

is negligible (in $|K|$ and ℓ) for any $\sigma \in \{0, 1\}$.

Let $F : \mathcal{K} \times \{1, \dots, \ell\} \times L \rightarrow \{0, 1\}$ be a pseudorandom function family. We assume that $\lambda : [\mathcal{I}] \rightarrow L$ is efficiently computable and $A : S_{\mathcal{P}2} \rightarrow L$ is a (δ, τ) -solution to $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$ as defined above (recall that $\mathcal{P}1$ is a special case of $\mathcal{P}2$), i.e. A operates in time τ and

$$\Pr_{t, i, r}[A_r(t(i)) = \lambda(i)] \geq \delta.$$

Let $c = \Pr_{i, j \leftarrow \mathcal{I}}[\lambda(i) = \lambda(j)]$. We *require* that $c < 1$ (that is, we require that there is enough variability in the labels of the images to be useful for communication). Notice that L can simply be equal to $[\mathcal{I}]$ and λ can be the identity function (in case the images in $[\mathcal{I}]$ have no labels as in $\mathcal{P}1$). We prove in the Appendix that the following construction is an efficient, robust stegosystem for $\mathcal{T}$.

Construction 1**Procedure SE :**

Input: $K \in \mathcal{K}$, $\sigma \in \{0, 1\}$
 for $j = 1 \dots \ell$ do
 draw $d_0 \leftarrow \mathcal{I}$, $d_1 \leftarrow \mathcal{T}$
 if $F_K(j, \lambda(d_0)) = \sigma$ then
 set $i_j = d_0$
 else
 set $i_j = d_1$
Output: $i_1, i_2, \dots, i_\ell$

Procedure SD :

Input: $K \in \mathcal{K}$, $i'_1 \dots i'_\ell \in [\mathcal{I}]^\ell$
 for $j = 1 \dots \ell$ do
 set $\sigma_j = F_K(j, A(i'_j))$
Output: $\text{majority}(\sigma_1, \dots, \sigma_\ell)$

Proposition 1. *Construction 1 is steganographically secret and robust for $\mathcal{I}, \mathcal{T}$.*

The proof of Proposition 1 is similar in flavor to those of Hopper, Langford and von Ahn [7] and relies on the fact that when A returns the correct solution on received image i'_j , the recovered bit σ_j is equal to the intended bit σ with probability approximately $\frac{1}{2} + \frac{1}{4}(1 - c)$ and otherwise $\sigma_j = \sigma$ with probability $1/2$; therefore the probability that the majority of the σ_j are incorrect is negligible in ℓ . For details, see the Appendix.

Remarks

1. Better solutions to $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$ imply more efficient stegosystems: if δ is larger, then ℓ can be smaller and less images need to be transmitted to send a bit secretly and robustly.
2. Since we assume that $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$ (or, as it might be the case, $\mathcal{P}1_{\mathcal{I}, \mathcal{T}}$) is easy for humans, our protocol *could* be implemented as a cooperative effort between the human recipient and the decoding procedure (without the need for a solution to $\mathcal{P}1_{\mathcal{I}, \mathcal{T}}$ or $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$). However, decoding each bit of the secret message will require classifying many images, so that a human would likely fail to complete the decoding well before any sizeable hidden message could be extracted (this is especially true in case we are dealing with $\mathcal{P}1_{\mathcal{I}, \mathcal{T}}$ and a large set $[\mathcal{I}]$: a human would have to search the entire set $[\mathcal{I}]$ as many as ℓ times for each transmitted bit). Thus to be *practical*, a (δ, τ) -solution (for small τ) to $\mathcal{P}1_{\mathcal{I}, \mathcal{T}}$ or $\mathcal{P}2_{\mathcal{I}, \mathcal{T}, \lambda}$ will be required.

6 Discussion and Closing Remarks**Interaction with the AI Community**

A primary goal of the CAPTCHA project is to serve as a challenge to the Artificial Intelligence community. We believe that having a well-specified set of goals will contribute greatly to the advancement of the field. A good example of this process is the recent progress in reading distorted text images driven by the CAPTCHA in use at Yahoo!. In response to the challenge provided by this test, Malik and Mori

[9] have developed a program which can pass the test with probability roughly 0.8. Despite the fact that this CAPTCHA has no formal proof that a program which can pass it can read under other distributions of image transformations, Malik and Mori claim that their algorithm represents significant progress in the general area of text recognition; it is encouraging to see such progress. For this reason, it is important that even Automated Turing Tests without formal reductions attempt to test ability in general problem domains; and even though these tests may have specific weaknesses it is also important that AI researchers attempting to pass them strive for solutions that generalize.

Other AI Problem Domains

The problems defined in this paper are both of a similar character, and deal with the advantage of humans in sensory processing. It is an open question whether CAPTCHAs in other areas can be constructed. The construction of a CAPTCHA based on a text domain such as text understanding or generation is an important goal for the project (as CAPTCHAs based on sensory abilities can't be used on sensory-impaired human beings). As mentioned earlier, the main obstacle to designing these tests seems to be the similar levels of program ability in text generation and understanding.

Logic problems have also been suggested as a basis for CAPTCHAs and these present similar difficulties, as generation seems to be difficult. One possible source of logic problems are those proposed by Bongard [4] in the 70s; indeed [1] presents a test based on this problem set. However, recent progress in AI has also yielded programs which solve these problems with very high success probability, exceeding that of humans.

Conclusion

We believe that the fields of cryptography and artificial intelligence have much to contribute to one another. CAPTCHAs represent a small example of this possible symbiosis. Reductions, as they are used in cryptography, can be extremely useful for the progress of algorithmic development. We encourage security researchers to create CAPTCHAs based on different AI problems.

Acknowledgments. We are grateful to Udi Manber for his suggestions. We also thank Lenore Blum, Roni Rosenfeld, Brighten Godfrey, Moni Naor, Henry Baird and the anonymous Eurocrypt reviewers for helpful discussions and comments. This work was partially supported by the National Science Foundation (NSF) grants CCR-0122581 and CCR-0085982 (The Aladdin Center). Nick Hopper is also partially supported by an NSF graduate research fellowship.

References

1. Luis von Ahn, Manuel Blum, Nicholas J. Hopper and John Langford. The CAPTCHA Web Page: <http://www.captcha.net>. 2000.

2. Luis von Ahn, Manuel Blum and John Langford. Telling Humans and Computers Apart (Automatically) or How Lazy Cryptographers do AI. To appear in *Communications of the ACM*.
3. Mihir Bellare, Russell Impagliazzo and Moni Naor. Does Parallel Repetition Lower the Error in Computationally Sound Protocols? In *38th IEEE Symposium on Foundations of Computer Science (FOCS' 97)*, pages 374–383. IEEE Computer Society, 1997.
4. Mikhail M. Bongard. *Pattern Recognition*. Spartan Books, Rochelle Park NJ, 1970.
5. A. L. Coates, H. S. Baird, and R. J. Fateman. Pessimal Print: A Reverse Turing Test. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR' 01)*, pages 1154–1159. Seattle WA, 2001.
6. Scott Craver. On Public-key Steganography in the Presence of an Active Warden. In *Proceedings of the Second International Information Hiding Workshop*, pages 355–368. Springer, 1998.
7. Nicholas J. Hopper, John Langford and Luis von Ahn. Provably Secure Steganography. In *Advances in Cryptology, CRYPTO' 02*, volume 2442 of *Lecture Notes in Computer Science*, pages 77–92. Santa Barbara, CA, 2002.
8. M. D. Lillibridge, M. Adabi, K. Bharat, and A. Broder. Method for selectively restricting access to computer systems. Technical report, US Patent 6,195,698. Applied April 1998 and Approved February 2001.
9. Greg Mori and Jitendra Malik. Breaking a Visual CAPTCHA. Unpublished Manuscript, 2002. Available electronically:
<http://www.cs.berkeley.edu/~mori/gimpy/gimpy.pdf>.
10. Moni Naor. Verification of a human in the loop or Identification via the Turing Test. Unpublished Manuscript, 1997. Available electronically:
<http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps>.
11. Benny Pinkas and Tomas Sander. Securing Passwords Against Dictionary Attacks. In *Proceedings of the ACM Computer and Security Conference (CCS' 02)*, pages 161–170. ACM Press, November 2002.
12. S. Rice, G. Nagy, and T. Nartker. *Optical Character Recognition: An Illustrated Guide to the Frontier*. Kluwer Academic Publishers, Boston, 1999.
13. Adi Shamir and Eran Tromer. Factoring Large Numbers with the TWIRL Device. Unpublished Manuscript, 2003. Available electronically:
<http://www.cryptome.org/twirl.ps.gz>.
14. J. Xu, R. Lipton and I. Essa. Hello, are you human. Technical Report GIT-CC-00-28, Georgia Institute of Technology, November 2000.

A Proof of Proposition 1

Lemma 1. *Construction 1 is steganographically secret.*

Proof. Consider for any $1 \leq j \leq \ell$ and $x \in [\mathcal{I}]$ the probability ρ_x^j that $i_j = x$, i.e. $\rho_x^j = \Pr[i_j = x]$. The image x is returned in the j th step only under one of the following conditions:

1. D_0 : $d_0 = x$ and $F_K(j, \lambda(d_0)) = \sigma$; or
2. D_1 : $d_1 = x$ and $F_K(j, \lambda(d_0)) = 1 - \sigma$

Note that these events are mutually exclusive, so that $\rho_x^j = \Pr[D_0] + \Pr[D_1]$. Suppose that we replace F_K by a random function $f : \{1, \dots, \ell\} \times L \rightarrow \{0, 1\}$. Then we have that $\Pr_{f, d_0}[D_0] = \frac{1}{2} \Pr_{\mathcal{I}}[x]$ by independence of f and d_0 , and $\Pr_{f, d_1}[D_1] = \frac{1}{2} \Pr_{\mathcal{I}}[x]$, by the same reasoning. Thus $\rho_x^j = \Pr_{\mathcal{I}}[x]$ when F_K is replaced by a random function. Further, for a random function the ρ_x^j are all independent. Thus for any $\sigma \in \{0, 1\}$ we see that $SE(K, \sigma)$ and $\mathcal{I}^\ell$ are computationally indistinguishable, by the pseudorandomness of F .

Lemma 2. *Construction 1 is steganographically robust for $\mathcal{T}$.*

Proof. Suppose we prove a constant bound $\rho > \frac{1}{2}$ such that for all j , $\Pr[\sigma_j = \sigma] > \rho$. Then by a Chernoff bound we will have that $\Pr[\text{majority}(\sigma_1, \dots, \sigma_\ell) \neq \sigma]$ is negligible, proving the lemma.

Consider replacing F_K for a random K with a randomly chosen function $f : \{1, \dots, \ell\} \times L \rightarrow \{0, 1\}$ in SE, SD . We would like to assess the probability $\rho_j = \Pr[\sigma_j = \sigma]$. Let A_j be the event $A(i'_j) = \lambda(i_j)$ and $\overline{A_j}$ be the event $A(i'_j) \neq \lambda(i_j)$, and write $\lambda(d_b^j)$ as l_b^j . We have the following:

$$\Pr_{f, i_j, t}[\sigma_j = \sigma] = \Pr[\sigma_j = \sigma | A_j] \Pr[A_j] + \Pr[\sigma_j = \sigma | \overline{A_j}] \Pr[\overline{A_j}] \quad (6)$$

$$= \delta \Pr[\sigma_j = \sigma | A_j] + \frac{1}{2}(1 - \delta) \quad (7)$$

$$= \delta(\Pr[f(j, l_0^j) = \sigma \text{ or } (f(j, l_0^j) = 1 - \sigma \text{ and } f(j, l_1^j) = \sigma)] + \frac{1 - \delta}{2} \quad (8)$$

$$= \delta(\frac{1}{2} + \Pr[f(j, l_0^j) = 1 - \sigma \text{ and } f(j, l_1^j) = \sigma \text{ and } l_0^j \neq l_1^j]) + \frac{1 - \delta}{2} \quad (9)$$

$$= \delta(\frac{1}{2} + \frac{1}{4}(1 - c)) + \frac{1 - \delta}{2} \quad (10)$$

$$= \frac{1}{2} + \frac{\delta}{4}(1 - c) \quad (11)$$

Here (7) follows because if $A(i'_j) = l \neq \lambda(i_j)$ then $\Pr_f[f(j, l) = \sigma] = \frac{1}{2}$, and (8) follows because if $A(i'_j) = \lambda(i_j)$, then $f(j, A(i'_j)) = \sigma$ iff $f(j, \lambda(i_j)) = 0$; the expression results from expanding the event $f(j, \lambda(i_j)) = 0$ with the definition of i_j in the encoding routine.

For any constant $\delta > 0$, a Chernoff bound implies that the probability of decoding failure is negligible in ℓ when F_K is a random function. Thus the pseudorandomness of F_K implies that the probability of decoding failure for Construction 1 is also negligible, proving the lemma.

ATTACHMENT 5

Shilling Recommender Systems for Fun and Profit

Shyong (Tony) K. Lam

John Riedl

GroupLens Research
Computer Science and Engineering
University of Minnesota
Minneapolis, MN 55455
{lam,riedl}@cs.umn.edu

ABSTRACT

Recommender systems have emerged in the past several years as an effective way to help people cope with the problem of information overload. One application in which they have become particularly common is in e-commerce, where recommendation of items can often help a customer find what she is interested in and, therefore can help drive sales. Unscrupulous producers in the never-ending quest for market penetration may find it profitable to *shill* recommender systems by lying to the systems in order to have their products recommended more often than those of their competitors. This paper explores four open questions that may affect the effectiveness of such shilling attacks: which recommender algorithm is being used, whether the application is producing recommendations or predictions, how detectable the attacks are by the operator of the system, and what the properties are of the items being attacked. The questions are explored experimentally on a large data set of movie ratings. Taken together, the results of the paper suggest that new ways must be used to evaluate and detect shilling attacks on recommender systems.

Categories and Subject Descriptors

H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval; H.3.4 [Information Storage and Retrieval]: Systems and Software; H.3.5 [Information Storage and Retrieval]: Online Information Services

General Terms

Experimentation, Algorithms

Keywords

collaborative filtering, recommender systems, shilling

1. INTRODUCTION

People face a bewildering number of choices when looking for items that they are interested in. A seemingly never-ending flood of content is available, but certainly not enough time exists to evaluate all possible choices. This, in a nutshell, is the problem of information overload. In recent years, recommender systems have emerged as one tool that can help people overcome this problem and quickly locate items to consume. These systems use opinions about items in some information domain in order to make

recommendations to a user regarding which items she may find interesting. One instance of a recommender system is *MovieLens* (<http://www.movielens.org>). GroupLens, our research group, operates this recommender system, which makes personalized recommendations suggesting movies that a user might like based on movies that she has seen and has expressed an opinion about.

While recommender systems are clearly beneficial to users, they can also be a valuable asset to retail companies in helping their customers find things that they might want to buy and, in effect, increasing not only sales, but perhaps also cross-sales and customer retention. This is particularly true in the realm of e-commerce. For example, *Amazon.com* has made many recommender systems available to their customers. These range from manually operated recommenders where users can recommend items to other users by writing reviews or creating lists, to automated systems where the site generates a list of recommended items based on what the user has looked at recently or has purchased in the past.

Producers of items (manufacturers, authors, etc.) would like their products to sell well in the marketplace. With recommender systems, there is a natural motivation to want one's own products to be recommended more often than those of a competitor. Of course, one way to accomplish this is to produce quality goods that people like and regard highly. However, unscrupulous producers may opt to take a more deceitful route; they may try to influence recommender systems in such a way that their items are recommended to users more often, whether or not they are of high quality. An instance of a company generating false "recommendations" to consumers arose in June 2001 when Sony Pictures admitted that it had used fake quotes from non-existent movie critics to promote a number of newly released films.¹ The online retailer *Amazon.com* has found that their recommenders are prone to some level of abuse on at least two different occasions.^{2,3} Also, *eBay*, which uses a recommender system as a reputation mechanism, has found itself continually dealing with users who subvert the system in various ways, including purchasing good ratings (feedback) from other members in order to bolster their own reputations.⁴

One way to influence a recommender system is to arrange to have a group of users (human or agent) enter the system and vouch for the items in question. These users become *shills*, whose false opinions are intended to mislead other users. Shills pose a serious threat to users and operators of recommender systems. They may cost users time and money by recommending bad items. They may

¹<http://news.bbc.co.uk/1/hi/entertainment/film/1368666.stm>

²<http://www.wired.com/news/ebiz/0,1272,53634,00.html>

³<http://news.com.com/2100-1023-976435.html>

⁴<http://www.auctionbytes.com/cab/abn/y03/m09/i17/s01>

Copyright is held by the author/owner(s).

WWW2004, May 17–22, 2004, New York, New York USA.

ACM 1-58113-844-X/04/0005.

cost operators by degrading the user's level of trust in the recommender system and the retailer behind it.

This paper focuses on recommender systems that use automated collaborative filtering (ACF) to generate recommendations. ACF is a class of algorithms commonly used in the implementation of recommender systems. These algorithms operate on the basis that similar users have similar tastes; thus, if people similar to you can be located, then the items they enjoy are likely to be ones you will also enjoy. These algorithms normally have two modes of operation: prediction and recommendation. In the *prediction* mode, the algorithm simply predicts how much a user will like some item or set of items. The items may have been selected by the user through browsing or searching. In the *recommendation* mode, the algorithm produces an ordered list of items that it believes the user is most likely to enjoy. This distinction will become important as we explore the practical effects of attacks on recommender systems.

The rest of the introduction comprises related work and the statement of hypotheses, setting the stage for the experimental work in the rest of the paper.

1.1 Related Work

1.1.1 Recommender Systems

One of the earliest instances of a collaborative filtering based recommender system is *Tapestry* [6]. In 1994, Resnick et al. [16] automated the collaborative filtering process and introduced an ACF algorithm based on k-Nearest-Neighbor. Since then, a number of improvements to kNN have been proposed [9, 8]. The *user-user* algorithm we study in this paper is a tuned version of the original kNN algorithm. Another related algorithm that we study is a tuned version of the item-based k-Nearest-Neighbor [18]. We call this algorithm the *item-item* algorithm in this paper. Many other recommendation algorithms have been developed as well, including ones based on singular value decomposition [17], Bayesian networks [2], and factor analysis [3].

Besides *MovieLens*, a myriad of other recommender systems exist, particularly on e-commerce sites. Schafer [19] examines and categorizes a large set of these commercialized recommender systems. In addition, numerous recommenders in a variety of domains have been developed for research purposes, including *GroupLens* (Usenet news) [11], *Ringo* (music) [20], and *Jester* (jokes) [7].

1.1.2 Attacks on Recommenders

While a large body of work exists on recommender system algorithms, less attention has been devoted to exploring and improving their resistance to attacks. Dellarocas [5] outlines several attacks on reputation systems used in online trading communities such as *eBay* and proposes a predictive algorithm similar to existing collaborative filtering algorithms which helps minimize the effect of the attacks. Canny [3] presents a system in which user preferences (ratings) are kept private both from the recommender system's administrators and from other users, which he believes can mitigate attacks where the system administrators are involved (e.g. a retailer being paid to place a company's items highly on recommendation lists).

More recently, O'Mahony et al. [14] have performed empirical studies of the resistance of the kNN user-user algorithm to attacks based on injecting a number of shill users into the system. The attacks were shown to be successful both at *push-ing* items by raising predicted ratings, and at *nuke-ing* items by lowering predicted ratings. Furthermore, they present a theoretical analysis of the effect of noise – perhaps injected by shills – on the performance of ACF algorithms and perform several experiments with real-world data

sets to evaluate the generated models. Our paper builds on their work by including more recommender algorithms and by evaluating the attacks on recommendations as well as on predictions.

1.1.3 Impact of Successful Attacks

Psychologists have shown experimentally that people tend to conform with the opinions of others, even when those opinions may be incorrect. This was the case in the classic conformity study by Asch [1] where test subjects were asked to perform a simple task in a small group. The other members of the group were working for the experimenter and deliberately gave incorrect responses. Even though the correct choice was easy to see, the test subjects agreed with the group and made the wrong choice in one-third of all trials.

A similar effect is likely to occur in interactions between computers and people. Nass and Moon [13] find that people tend to treat computers as they would treat people – in one of their experiments, they show that people are less likely to criticize a computer's performance if that computer asks for the evaluation than if another computer asks.

Cosley et al. [4] find that users are indeed affected by manipulated predictions provided by a recommender. As suggested by the conformity studies mentioned above, users tend to rate items toward the predicted rating provided by the recommender regardless of prediction accuracy. Whether this represents a genuine change in opinion is unknown – it might just be that users conform in what they say, not in what they believe – but the result indicates that it may very well be possible that some users can be fooled into accepting a bad recommendation. Further, the manipulated predictions experiment suggests that shilling may be doubly dangerous, since affected users may rate towards the manipulated predictions, influencing other users who trust them.

1.2 Hypotheses

The previous research on shilling has demonstrated that shilling should be of concern in recommender systems, but leaves several important questions unanswered. The first of these questions is about how effective shilling is on finely tuned versions of the user-user [9] and item-item [18] algorithms. Though the algorithms are based on fundamentally similar ideas about relationships between users and items, their implementations are sufficiently different that we suspect they may exhibit different behavior under shilling attack. Formally:

HYPOTHESIS 1. *Different ACF algorithms respond differently to shilling attacks.*

This hypothesis has importance for designers of ACF algorithms, who may be able to design algorithms that are more resistant to attack, for operators of recommender systems, who may be able to select algorithms that are resistant to likely attacks, and for evaluators of shilling attacks, who may need to be aware their results are algorithm dependent. We suspect there will be differences in shilling resistance among other less similar recommender algorithms as well; if we find differences between these two algorithms we will recommend future work to understand in detail the shilling resistance of the entire suite of known recommender algorithms.

Recommender systems in e-commerce, where shilling is most often feared [5, 14], are used more often to produce recommendations [19] than predictions. Attacks should be judged based on how they affect the recommender in its most common mode of operation – after all, if a user only looks at the top 10 items on a list, does it matter that the shilling attack has changed the prediction she would have seen for the 500th item on her list?

HYPOTHESIS 2. *Shilling attacks affect recommender algorithms differently from prediction algorithms.*

If this hypothesis is supported, it will mean that researchers who evaluate attacks must base their measure of the effectiveness of the attack on the specific ways the targeted recommender system is being used. Metrics used in the past for evaluating recommender algorithms have all been focused on predictions; perhaps for recommendation tasks recommendation metrics should be used instead.

The fact that past research has shown that shilling is effective in some cases, raises the question of whether the operators of recommender systems can detect that their systems are under effective shilling attack. Our next hypothesis is that they cannot do so with existing tools. Note that this hypothesis is distinct from hypothesis 2 even though both are about metrics. Hypothesis 2 is about measures that evaluate shilling attacks based on knowing exactly which items are being attacked, which the operator of the recommender system will not know.

HYPOTHESIS 3. *Shilling attacks are not detectable using traditional measures of algorithm performance.*

ACF algorithm designers often utilize metrics such as Mean Absolute Error (MAE) to evaluate the overall predictive accuracy of their algorithms and to compare it with other algorithms. Thus, it may be tempting to use such metrics to detect attacks by looking for changes in algorithm quality caused by attacks. We believe this will generally not be possible; that is, attacks can be subtle and focused enough that their overall effect on the system is minimal.

This hypothesis is unnerving to those who – like us – run a recommender system, since it means that our systems may already be under successful attack, and that despite all of the measurement tools at our disposal we may be unaware of all but the crudest attacks. We seek to spur understanding of which evaluation techniques are best for detecting shilling attacks.

One possible place to look for detecting shilling attacks is to understand which target items are most vulnerable to them. We hypothesize that the number of ratings of an item, and the spread of those ratings over possible ratings values influence the extent to which an attack can succeed.

HYPOTHESIS 4. *Ratings distribution of the target item influences attack effectiveness.*

We will study popularity (number of ratings), likability (average rating), and entropy as the variables that describe the ratings distribution. We believe that the following statements will be true:

- Likability – the more well-liked an item is (that is, the higher its average rating is), the easier it is to cause that item to be recommended more often
- Popularity – the less popular an item is (that is, the fewer the number of ratings it has), the easier it is to manipulate the predictions and recommendations for that item
- Entropy – the higher the entropy of an item's ratings, the easier it is to manipulate the predictions and recommendations for that item

1.3 Contributions

This paper builds on the work presented in [14] and further explores the feasibility and effectiveness of influencing recommender systems based on ACF algorithms through shill attacks.

First, we propose a set of dimensions that describe and categorize a wide variety of shill attacks. These dimensions set the stage for

the types of evaluation we carry out, though our evaluation so far is only of the dimensions we expect to be most important in practice.

Second, we build on past work to develop two basic attack types, and perform a series of experiments to study their effectiveness in influencing both the predictions and the recommendations made by two different ACF algorithms. In our experimental work we simultaneously evaluate all three aspects of the shilling attack: the attack itself, the ACF algorithm under attack, and the different metrics used to evaluate both the algorithms and the attacks.

Finally, we present the results of the experiments and examine how they support or refute our hypotheses, and conclude with a look at some open questions and possible future work.

2. DIMENSIONS OF ATTACKS

In this paper we only consider *shill attacks* where the attacker's available action is to introduce a new set of users and a set of ratings made by those new users to the recommender system. In the case of online recommender systems, more traditional attacks such as denial-of-service, password cracking, system hacking, and bribery are possible, but are beyond the scope of this paper.

Each shill attack has a number of intrinsic properties that can be useful in describing and comparing different attacks. A list of these properties, or dimensions, follows.

2.1 Attack Intent

Different shill attacks may have very different intents. While the direct result of a shill attack is generally that the predictions made to users are manipulated in some way, the eventual goal of a shiller can be one of several alternatives. Two straightforward intents are to “push” one or more items in the system in order to have them recommended to more users and, conversely, to “nuke” a set of items to cause them to be recommended to fewer users.

Another possible intent is to simply damage the recommender system as a whole; that is, to reduce prediction and recommendation quality across the board with the goal of causing users to stop trusting the system and eventually to stop using it. A successful attack of this nature might benefit competing recommender systems.

2.2 Targets

Shill attacks can be directed at a particular subset of users and a subset of items in a recommender system. It is in an attacker's best interest to restrict the effect of an attack to a small target set of items in order to be more subtle and try to avoid detection by the system operators. Additionally, it might be beneficial to also restrict the effect to some desirable set of target users. For instance, it might induce suspicion to cause a heavy metal album to be recommended to a connoisseur of classical music who would have absolutely no interest in such an album.

2.3 Required Knowledge

Attacks may require some level of knowledge about the items, users, ratings, and algorithms in the recommender system being attacked. An informed attack will generally be more effective than an uninformed attack. Further knowledge about the system such as ratings sparsity, ratings distribution, and algorithm parameters can help in choosing which attack to employ and in tuning attack parameters to maximize effectiveness and minimize detectability.

2.4 Cost

A shill attack has an associated cost value that depends on the level of effort and information needed to successfully execute the attack. With a cost dimension and a suitable means of evaluating attack effectiveness, one might be able to evaluate attacks on a

cost/benefit basis to determine if it is economically worthwhile to execute an attack. The following factors contribute to the cost of a given attack:

- Size of attack: the number of new users and ratings.
- Difficulty of interacting with the recommender system. For instance, an attack on a system that employs anti-automation techniques such as CAPTCHAs may have a far higher cost than an attack on a system that does not [21].
- Obtaining required knowledge about the algorithm, users, items, and ratings in the recommender system.
- Any other resources required for attack planning or execution, such as additional logistical, computational, or technical requirements.

2.5 Algorithm Dependence

Some shill attacks may be specifically designed to exploit a particular weakness in a specific algorithm or class of algorithms, while others might be more general and can be effective against a variety of algorithms. More specific attacks will intuitively require fewer resources for the same effectiveness, but require detailed knowledge of the algorithm being used and its parameter settings.

2.6 Detectability

Inherent properties of attacks may make them more or less easily detectable, both to users and operators of the recommender system. In general, an attacker would like to be less detectable to operators in order to be able to sustain the attack for as long as possible before being discovered and stopped. The importance of detectability to users depends on the attack intent. This dimension is one that can evolve very rapidly as shill detection methods are developed or improved, similar to the current arms race seen in spam (unsolicited commercial email) detection and detection evasion.

3. EXPERIMENTAL DESIGN

3.1 Our Shill Attack Design

This paper focuses on algorithmic attacks that attempt to push or nuke an item (that is, raise or lower the recommender's predictions for the item) by introducing shills into the system. It is assumed that the attacker does not have access to the ratings matrix but can obtain broad statistical measures of the ratings data and other publicly available data. We do not concern ourselves with reducing the cost of the attacks or with designing attacks that are difficult to detect. If these brute force attacks prove difficult for operators to detect, recommender systems operators should be concerned indeed!

We begin with the type of attacks used in [5]. These attacks inject a collection of new users into the system, each of which has rated a set of items to try to be similar to existing users, and has rated the particular item being attacked very high in order to push it. Practical implementations of both user-user and item-item algorithms scale correlations according to the number of ratings in common [9]. It has been observed that the attack from [5] does not work well for this variant of the kNN algorithms if too few items are rated by the shill users because the similarities are scaled down too far to be considered by the algorithm [14]. We therefore modify the attack to rate *all* movies in the system to maximize the number of items in common between shill users and real users. Attacks like these are related to *filterbots*, which also rate all items [8]. However, filterbots are used to improve recommendation coverage and

quality, while these shills ("shillerbots," perhaps) will be used to decrease quality for a small subset of items.

One way in which the ACF algorithms we test are different from those used in the past [5, 14] is that we do not use negative correlations between items. We made this decision because our past experience has been that negative correlations often lead to recommendations that are inconsistent with user preferences. Shills might exploit negative correlations to produce very strong attacks – but system operators will likely disable the use of negative correlations to improve quality first.

Our attacks target the entire population of users and a small target set of items while requiring relatively limited knowledge about the ratings matrix. The number of new shill users introduced to the recommender system is varied between 25 and 100. The intent of these attacks is to either push or nuke the target set of movies. The two attack methods developed are:

3.1.1 RandomBot

RandomBots are a naive attack in which each introduced user rates items not in the target set randomly on a normal distribution with mean 3.6 and standard deviation 1.1. These values are chosen because they represent the ratings distribution in the data set. Even if these values are not known to the attacker, they can be estimated relatively easily, perhaps by observing people using the recommender system and obtaining a sample of their ratings. A normal distribution is used to approximate the observed user rating behavior in *MovieLens*. To accomplish its objective, the filterbot rates items in the target set with value equal to either the minimum or maximum allowed rating, depending on its intentions (nuke or push, respectively).

3.1.2 AverageBot

AverageBots are a somewhat more sophisticated attack than RandomBots and require knowledge of the average rating of each item in the system. A number of recommender systems, including *MovieLens*, will readily provide this information. Furthermore, such aggregate information about users' preferences may be found from other sources. In the case of movies, the *Internet Movie Database* (<http://www.imdb.com>) publicly displays the average user ratings of listed movies.

Each introduced user rates items not in the target set randomly on a normal distribution with mean equal to the average rating of the item being rated and standard deviation 1.1. The rationale behind this attack is that providing ratings centered around the average rating will help the filterbot be more similar to existing users, and thus, have a larger effect on the recommendations. As with the RandomBot, items in the target set are rated with a minimal or maximal value depending on the intent.

3.2 Data Set

A data set derived from *MovieLens* consisting of 999,799 ratings on 3,404 movies by 7,463 users is used in all experiments in this paper. All ratings are integral values between 1 and 5, inclusive, where 1 represents a poor movie and 5 represents an excellent one.

3.3 ACF Algorithms

We experimented with two commonly-used ACF algorithms – kNN user-user and kNN item-item.

3.3.1 kNN User-User

The classic kNN user-user algorithm introduced by Resnick in [16] is chosen because it is considered to be a good baseline algorithm and is widely used in both academia and industry.

The user-user algorithm uses the following formula to compute a predicted rating p for a user u on an item i .

$$p_{u,i} = \bar{r}_u + \frac{\sum_{v \in U_{u,i}} [w_{u,v}(r_{v,i} - \bar{r}_v)]}{\sum_{v \in U_{u,i}} |w_{u,v}|}$$

Here, $\bar{r}_u$ is user u 's average rating over all rated items, $w_{u,v}$ is the mean-adjusted Pearson correlation ("similarity") between users u and v , and $U_{u,i}$ is user u 's neighborhood with respect to item i and consists of the k users who have rated i and have the greatest Pearson correlation with u . k is a tunable parameter and represents the number of neighbors.

Several optimizations and suggested parameters from [9] are used. In particular, we set k to 20 and use the $n/50$ significance weighting and deviation from mean optimizations. A similarity threshold of 0.1 is also used. The only difference between the user-user variant used in this paper and the published algorithm is that only positive similarities are considered here; negative similarities and ones under the threshold are ignored. This variant was also used for several years in the *MovieLens* recommender system and does not appear to be detrimental to performance.

3.3.2 kNN Item-Item

The kNN item-item algorithm was introduced by Sarwar et al. [18] and is similar to user-user in both definition and predictive accuracy. However, it computes and uses similarities between items rather than users. We choose to experiment with this algorithm in order to explore how well an attack that is effective against user-user operates with a somewhat-different algorithm. The formula used to compute a prediction in item-item is:

$$p_{u,i} = \frac{\sum_{j \in \text{all similar items}} [s_{i,j} * r_{u,j}]}{\sum_{j \in \text{all similar items}} |s_{i,j}|}$$

Here, $s_{i,j}$ is the similarity between items i and j . The algorithm implementation used in this paper is the *MultiLens* recommender engine developed by Brad Miller [12], which uses the adjusted cosine method of computing similarity, and considers the 20 rated items with highest similarity to be the set of "all similar items" (again, ignoring negative similarities). A tuned version of this algorithm is currently utilized in *MovieLens*.

3.4 Methods

A total of twenty-four experiments were performed in a 2x2x2x3 design. The algorithm (user-user or item-item), attack type (AverageBot or RandomBot), attack intent (nuke or push), and number of new users/bots (25, 50, or 100) were varied in each experiment.

The target set for the experiments consists of 22 items. This set was selected to include a variety of different movie types including future releases, new releases, obscure films, popular films, controversial films, and long-standing favorites. In terms of ratings properties, this selection of items represents a wide range of popularity (number of ratings), entropy (a measure of the variance of ratings), and likability (mean rating). Table 1 displays the properties of items in the target set.

3.5 Metrics

There are two things that we would like to be able to measure:

1. How much the overall accuracy of the ACF algorithm is affected by an attack.
2. How effective an attack is in accomplishing its goal.

Table 1: Properties of movies in chosen target set. Ratings is total number of ratings (popularity), mean is average rating (likability), and entropy is the standard information-theoretic entropy of the ratings distribution. Recall that ratings are provided on a 5-point scale.

Item	Ratings	Mean	Entropy
1	0	N/A	N/A
2	7	2.57	0.99
3	8	2.88	1.56
4	17	3.00	2.04
5	17	3.24	1.61
6	18	2.11	1.99
7	27	3.15	2.21
8	34	4.00	1.53
9	46	1.78	1.67
10	48	2.60	2.18
11	92	3.17	2.15
12	105	1.72	1.65
13	106	2.71	2.12
14	116	4.12	1.59
15	234	2.44	2.17
16	263	1.89	1.82
17	354	4.28	1.60
18	422	3.20	2.26
19	1298	3.78	1.68
20	1828	4.49	1.39
21	2316	3.35	2.18
22	2654	3.74	2.08

We consider many metrics here. The first time we introduce a metric, the name will be in **bold face**.

To address the first requirement, we turn to a metric commonly used to evaluate ACF algorithms, **Mean Absolute Error**, or MAE. This is simply the average absolute difference between the predicted rating and actual rating over all users and items in a test set. The algorithm performance is evaluated before and after each attempted attack to determine each attack's effect on the system as a whole.

While MAE is generally considered to be a standard in evaluation of ACF algorithms, we find that there is a mismatch between what it measures and what really matters to users in a recommender system. Recommender systems are a decision support tool, and in general, the exact predicted value is of far less importance to a user than the fact that an item is recommended.

In practice many recommender systems are used by e-commerce sites to help their customers find products to purchase. One might expect that many of these systems produce lists of recommendations for their users, rather than interfaces that require the user to explicitly seek out the products that she desires. In fact, Schafer et al. found that by far the most common output of e-commerce recommender engines was suggestions of items for the customers to purchase [19]. Therefore, the quality of the *recommendations* should be emphasized rather than the quality of the predictions.

Furthermore, since users do not tend dig very "deeply" when shown a list of things to examine, only the items near the top of a recommendation list should be considered when evaluating algorithm quality. We analyzed *MovieLens* user behavior from the search logs, and discovered that the median recommendation search ends within the first 40 items displayed. Figures 1 and 2 show the browse depth of 137,991 *MovieLens* recommendation (top- N)

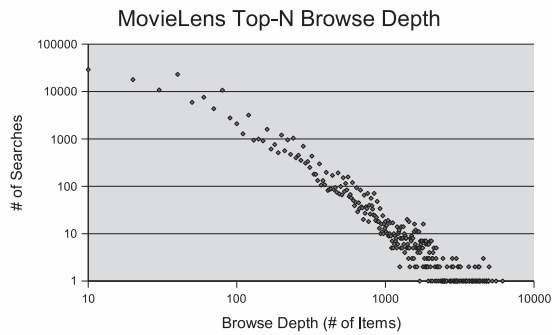


Figure 1: Log-log graph of top- N search browse depths in *MovieLens*. The median search depth is 40, and 79% of recommendation searches end at or before 80 items.

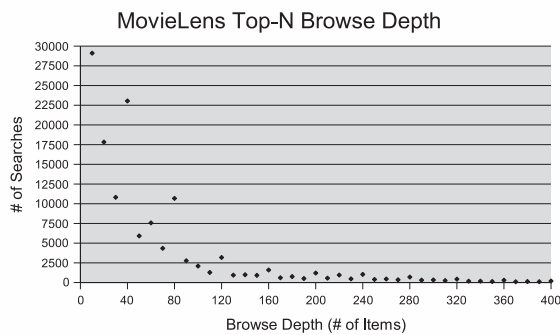


Figure 2: Partial graph of top- N search browse depths in *MovieLens*. The long tail to the right of 400 items is not shown.

searches. Figure 2 is truncated at 400 items so that the low median is easier to visualize. Figure 1 shows the entire set of data on a log-log scale so that the long tail is visible. The phenomenon that users do not dig deeply through a list of results may be even more acute in other domains. For instance, one study found that 54 percent of users view only a single page of search results in each session [10]. Developing a top- N recommendation accuracy metric for entire recommender systems is beyond the scope of this paper; however, we will develop an attack effectiveness metric that is centered around recommendations rather than predictions.

Before we do so, we first examine the attack effectiveness metrics used in previous work. One metric that is introduced in [14] is the **Stability of Prediction** metric, which measures the relative number of predictions for target items that are *not* manipulated beyond some given threshold. This metric does not directly address the notion of manipulating top- N recommendation lists. It treats all prediction manipulations of equal amount as being of equal importance. However, this is intuitively not the case – for instance, on a 5-point rating scale, causing a prediction to change from 2 to 3 is far less meaningful than moving it from 4 to 5. An item with a 3-point prediction is usually far less likely to appear on a recommendation list than an item with a 5-point prediction.

Another metric also introduced in [14] is the **Power of Attack** metric, which is defined as the average change in prediction toward some target value (usually the minimum allowed rating r_{min} or the maximum allowed rating r_{max}) over all target users and items. We believe that this has the same drawbacks as the Stability of Prediction

metric. Since Power of Attack is defined *differently* in earlier work [15] by the same authors, we will refer to this metric as **Prediction Shift** when presenting our results.

We will use the version of the metric defined in [15] when referring to **Power of Attack**. This is the percentage of predictions for target items *not* manipulated to some target value (again, r_{min} or r_{max}). For attacks with a push intent, we find that this metric is somewhat useful for gauging impact on recommendation lists for attacks with a push intent, even though it is a prediction-based metric. An item that has its predicted rating manipulated to r_{max} is indeed likely to appear in a top- N list.

However, this metric falls short in a number of ways. First, the metric is less suitable for measuring the effectiveness of attacks with a nuke intent. In the vast majority of cases, it is unnecessary to manipulate a prediction to r_{min} to cause it to not appear in a top- N . Secondly, some algorithms are extremely conservative in making predictions of r_{max} . This metric is less able to accurately gauge attack performance in this case, as the values the metric takes will be universally high (remember: high values mean most items *not* successfully manipulated). Conversely, some algorithms may be fairly liberal in producing predictions of r_{max} . This metric does not take into account the possibility that more than N items can have a predicted rating of r_{max} – in such cases, the top- N recommendation list is not well-defined and the method used to break such “ties” to determine which N items to display is implementation-dependent.

So, we are left with a need to define a metric that directly measures the effect of an attack on top- N recommendation lists. Furthermore, the metric should take into account the possibility of items tied for inclusion in these lists. Based on these requirements, we propose a metric called **Expected Top- N Occupancy** (ExpTop N). The metric is defined as the expected number of occurrences of *target items* in a top- N recommendation list, measured over all users, assuming that the displayed ordering of items tied at any particular rank is random. We believe that in the absence of any information about the algorithm implementation, the metric should treat all possible tiebreaker functions equally, which has the beneficial side effect of leading to a metric that gives a lower value for attacks that result in many ties involving target items.

For example, let the target items be items E and F , and suppose that the candidates for inclusion in a top-5 recommendation list for some user are as shown below.

Rank	Item	Pred
1	B	5.0
2	E	4.9
3	D	4.7
4	A	4.5
4	C	4.5
4	F	4.5

In this case, there are three items tied for the fourth and fifth entry in the top-5 list. One of these three items is in the target set. Assuming all orderings of these three items are equally likely, the Expected Top-5 Occupancy for this user is 1.666. The target item E in the top-5 contributes 1, and item F being in two out of the three permutations for the fourth and fifth items displayed on the list contributes 0.666.

By examining the change in this metric caused by some attack, one can determine the attack’s effect as perceived by users of the recommender system. To reflect actual *MovieLens* usage, we will use $N = 40$ in accordance with our system usage analysis. To compare this metric with the prediction-centric ones, we will report MAE as well as the value of both Power of Attack metrics. The

Table 2: Effect of attacks as measured by the Prediction Shift metric (PredShift) and the change in MAE (Δ MAE). An increase in MAE indicates lower overall predictive accuracy.

Algorithm	Intent	Attack	Bots	PredShift	Δ MAE
User-user	Push	Random	25	0.499	0.002
			50	0.671	0.004
			100	0.830	0.009
		Average	25	1.032	0.006
			50	1.189	0.011
			100	1.300	0.019
	Nuke	Random	25	0.422	0.002
			50	0.589	0.004
			100	0.759	0.010
		Average	25	0.656	0.007
			50	0.815	0.014
			100	0.956	0.023
Item-item	Push	Random	25	0.030	0.002
			50	0.053	0.002
			100	0.069	0.004
		Average	25	0.363	0.002
			50	0.426	0.004
			100	0.471	0.010
	Nuke	Random	25	-0.046	0.002
			50	-0.069	0.002
			100	-0.092	0.004
		Average	25	0.332	0.003
			50	0.354	0.006
			100	0.361	0.014

earlier definition from [15] will be referred to as Power of Attack (POA) and the later definition from [14] will be referred to as Prediction Shift (PredShift).

4. RESULTS AND DISCUSSION

The discussion of results is organized according to the hypotheses they relate to.

HYPOTHESIS 1. *Different ACF algorithms respond differently to shilling attacks.*

Table 2 shows how each of the attacks affected the predictions made by the ACF algorithms. This table reports the two prediction-centric metrics, PredShift and change in MAE. These metrics are most appropriate for applications that form predictions for items selected by the user. First, we examine the attack effect as measured by the prediction shift metric. We delay discussion of MAE until the presentation of the hypothesis about detecting shilling, because we found MAE not very useful for the other hypotheses. In general the MAE changes were very small, and seem unlikely to represent noticeable change to users.

The user-user algorithm responds very strongly to all attacks; that is, the attacks are successful to some extent in manipulating the predictions for items in the target set. For the push attacks, the PredShift metric indicates that AverageBot-based attacks are able to raise the predictions by over one point, and RandomBot-based attacks are somewhat less effective. A similar pattern is seen for the nuke attack.

On the other hand, the item-item algorithm responds far less strongly. According to PredShift, the *most* effective push attack on item-item (100 AverageBots) has an effect on predictions that is

Table 3: Effect of attacks as measured by the Power of Attack metric (POA) and percent change in Expected Top-40 Occupancy (ExpTop40). For POA, a lower value means the attack was more effective. The value of Expected Top-40 Occupancy pre-attack is 0.57 for the user-user algorithm and 0.24 for item-item.

Algorithm	Intent	Attack	Bots	POA	ExpTop40
User-user	Push	Random	25	0.900	711%
			50	0.865	1190%
			100	0.816	1649%
		Average	25	0.715	1286%
			50	0.609	1674%
			100	0.519	1918%
	Nuke	Random	25	0.943	-39%
			50	0.928	-33%
			100	0.908	-32%
		Average	25	0.963	-67%
			50	0.952	-70%
			100	0.943	-75%
Item-item	Push	Random	25	1.000	150%
			50	1.000	171%
			100	1.000	229%
		Average	25	0.999	158%
			50	0.999	154%
			100	0.999	117%
	Nuke	Random	25	0.954	146%
			50	0.954	204%
			100	0.954	333%
		Average	25	0.955	-33%
			50	0.955	-54%
			100	0.954	-71%

comparable to the *least* effective push attack on user-user (25 RandomBots). RandomBots have an even weaker effect on item-item predictions – no average prediction shift is greater than one-tenth of a point.⁵

Next, we turn to Table 3, which shows the results of the attacks in terms of metrics that measure their ability to affect recommendations. These metrics are more appropriate for applications that generate lists of suggested items for their users.

According to the POA metric, push attacks on the user-user algorithm are more successful than nuke attacks, with AverageBot being superior to RandomBot. In the MovieLens domain push attacks may be easier because the average rating is higher than the midpoint of the scale. With the item-item algorithm, push attacks seem ineffective – the values of 1 and 0.999 indicate that essentially no predictions for items in the target set were successfully manipulated to the target predicted value of r_{max} , or 5. Nuke attacks on item-item are roughly as effective as they are on user-user, according to POA.

The ExpTop40 metric should be sensitive to prediction changes that influence the top 40 items, whether or not those changes move the predictions to the top of the scale. With this metric, we see a striking difference in response between the two algorithms. The

⁵Note that using RandomBots in a nuke attack actually *increases* predictions for the item slightly. We are unsure why this happens, though we have been able to show in small-scale examples that attacks on one user in item-item sometimes result in a reverse effect for other users. That is, if an item is pushed for some users, that item is nuked for other users. Perhaps the increase is due to a reversal on a subset of users.

user-user algorithm is profoundly affected by push attacks, with AverageBot again being more successful than RandomBot. Note that a 100-AverageBot attack causes a nineteen-fold increase in how often the target items are recommended in the top-40!

Clearly, the push attacks have a far more subdued effect on the item-item algorithm according to ExpTop40. In the strongest case, ExpTop40 increases by 229% with a 100-RandomBot push attack. Unexpectedly, we see that a 100-RandomBot *nuke* attack has an even greater effect, causing target items to be recommended over three times as often! So, while there is a large effect on item-item recommendations in a relative sense, it may not be considered substantive enough to be worthwhile for an attacker. Even after the strongest attack, less than one occurrence of a target item appears in each user's top-40 on average.

Looking at the nuke attacks, we see that the AverageBot-based attacks are successful on both ACF algorithms in manipulating the top-40. With 100 AverageBots, the number of items from the target set appearing in a user's top-40 is reduced by about 75% on the user-user algorithm and 71% on the item-item algorithm.

So, we find that the user-user and item-item algorithms do respond differently, particularly in how their recommendations are affected under push attacks. We judge that the evidence supports the hypothesis that different algorithms respond differently to shilling attacks. Algorithms that are less similar than user-user and item-item may react even more differently.

HYPOTHESIS 2. *Shilling attacks affect recommender algorithms differently from prediction algorithms.*

To address this hypothesis, we compare the values of the PredShift, POA, and ExpTop40 metrics. PredShift and POA are prediction metrics, while ExpTop40 is a recommendation metric. For many of the tests both classes of metrics move in the same direction. For instance, under a AverageBot-based push attack on user-user, PredShift is at least a half point, POA shows that at least thirty percent of predictions were successfully manipulated to r_{max} , and the change in ExpTop40 shows that many target items are pushed into the top 40.

However, there are notable differences, too. For instance, The PredShift metric indicates that RandomBot-based push attacks have a nearly negligible effect on item-item's predictions for target items. The ExpTop40 metric, on the other hand, says that up to twice as many target items are in the top 40 after this attack.

Conversely, with Average-bot based push attacks on item-item, PredShift shows that there is an measurable effect on the predictions, but the effect on recommendations as measured by ExpTop40 are minimal. Thus, this attack does appear to have different effects on the prediction and recommendation modes of item-item.

Note that in a few cases, the POA metric shows diametrically opposed results in comparison with the other metrics. In the RandomBot nuke attacks on item-item, PredShift and ExpTop40 show that the predictions and recommendation frequencies for target items *increased*, while POA indicates some success in reducing predicted values for target items to r_{min} , or 1. In fact, POA indicates just as much success for this attack as it does for the AverageBot nuke attack on item-item, which PredShift and ExpTop40 agree actually does nuke the target items. POA is unable to distinguish between AverageBot and RandomBot attacks on item-item even though each attack has different effects on the predictions and recommendations according to the other metrics.

The high values POA takes for attacks on the item-item algorithm is a result of one of the weaknesses we mentioned earlier. We have observed in previous experiments that item-item tends to be more conservative than user-user in giving very high or very low

predictions. Thus, it is difficult to manipulate a prediction to either extreme of the scale, and as a result, the POA metric gives deceptively high values for item-item. Because of this deficiency, we recommend against using POA as the sole metric for evaluating shilling attack performance.

Overall, the evidence for this hypothesis is mixed. Usually, all of the metrics we studied move in the same direction, and it is hard to directly compare them. However, in a few cases there are clear differences between an attack's impact on predictions and on recommendations. We believe these cases are sufficient evidence to argue that selection of an appropriate metric is important. If recommendation is more important than prediction for the recommender system, then a recommendation-based metric such as Expected Top- N Occupancy should be used. On the other hand, if prediction is more important, a prediction-based metric such as Prediction Shift should be used.

HYPOTHESIS 3. *Shilling attacks are not detectable using traditional measures of algorithm performance.*

We return to Table 2 where the last column shows the change in MAE caused by each attack. The MAEs are obtained by performing five-fold cross-validation on a 80%/20% test/train split of the data set before and after each attack. Note that no attack increases the mean absolute error by more than 0.023, and that RandomBot attacks on item-item induce extremely small reductions in accuracy. These changes seem small, but some of them are comparable to the improvements in MAE that are often considered "significant" differences between algorithms [9].

As an aside, it is unclear whether *people* are able to perceive such differences in system accuracy. Cosley et al. [4] show that users are able to detect intentionally-produced shifts of one point on many items in a small set of predictions, but the average change in error here is over 40 times smaller and is distributed across a large set of items. Furthermore, the system's user interface may not make small changes in prediction evident – for instance, *MovieLens* only displays predictions in half-point increments. Of course, whether users can detect changes in MAE is not directly related to whether MAE can be used by system operators to detect shills.

Overall, we find that insufficient evidence exists to support or reject this hypothesis. Some less naive attacks may fall "under the radar," but others may be detectable by watching standard metrics such as MAE. Still, we believe that this ambiguity suggests that other ACF algorithm performance metrics are necessary to reliably detect attacks.

HYPOTHESIS 4. *Ratings distribution of the target item influences attack effectiveness.*

We hypothesized that the properties of an item's ratings distribution has an effect on how much impact an attack has on that item. Recall that the properties mentioned earlier were popularity, likability, and entropy. Intuitively, if an item has few ratings (low popularity) and/or has a high spread of ratings (entropy), it should be easier to manipulate the predictions and recommendations for that item because it is more "volatile" in some sense. Likewise, items that are already well-liked should be easy to push, while items that are disliked by many should be easy to nuke.

We examined the effects of these variables on PredShift and ExpTop40. We only considered push attacks for ExpTop40, since so few of our target items were in top 40 lists prior to the attacks, so nuke attacks would have few items to "nuke."

Figures 3, 4, and 5 show several examples of these relationships (or lack thereof). The likability of an item correlates with PredShift

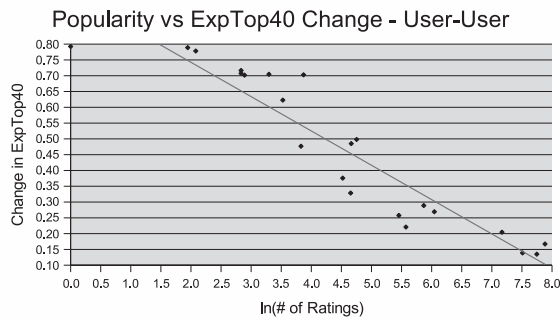


Figure 3: Relationship between popularity of an item and the effect of a 100-AverageBot push attack on user-user recommendations ($r^2 = 0.874$)

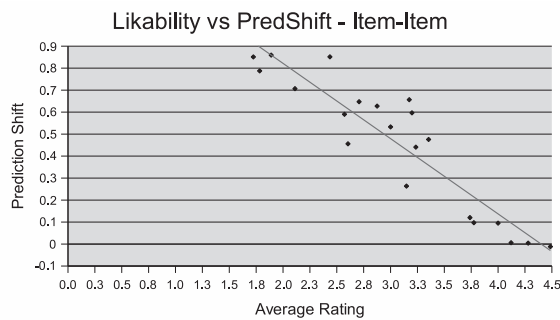


Figure 4: Relationship between likability of an item and the effect of a 100-AverageBot push attack on item-item predictions ($r^2 = 0.853$)

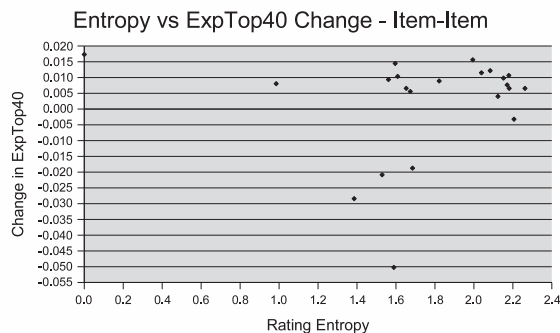


Figure 5: (Lack of) Relationship between entropy of an item and the effect of a 100-AverageBot push attack on item-item recommendations ($r^2 = 0.004$)

on both the user-user and item-item algorithms, but not with the change in ExpTop40 in either algorithm. The popularity of an item correlates highly with the change in ExpTop40 on user-user, but not on item-item. Furthermore, popularity does not correlate well with PredShift. Finally, the entropy of an item's ratings distribution correlates with neither metric and neither algorithm.

So, while a portion of this hypothesis is rejected, the supported part of the hypothesis does carry an important implication for sys-

tems that use the user-user algorithm for *recommendations*. Items that have low popularity can be an easy push target for attackers. This result is significant because new items initially simultaneously have low popularity, and are highly desirable to push, since new items often sell at a premium.⁶

Note that these results also provide further support for our first hypothesis that different ACF algorithms respond differently to attacks.

5. CONCLUSIONS AND FUTURE WORK

Overall, our results are mixed. However, there are some conclusions that can be drawn for operators of recommender systems who would like to reduce the threat of shilling to their systems. We first identify those lessons, then discuss what all of the shilling studies to date together say about the dimensions of shilling, and close by suggesting some rich areas for future work.

Prefer Item-Item. The item-item algorithm was much less affected by the attacks in our study than was the user-user algorithm. In many domains, item-item provides recommendations that are the same or better quality as user-user, and the same or better performance. Our study suggests one more reason for operators to prefer item-item: it appears to be more resistant to shilling.

One internal reviewer of our paper suggested that the reason user-user is easier to attack is that the version we used has a neighborhood size of only 20. The argument is that the small neighborhood makes it easy for shills to displace “good” neighbors. To test this hypothesis we repeated some of our experiments with a neighborhood size of 500. We found only very small differences – and the larger neighborhood proved overall easier to shill!

Use Recommendation Metrics. Most recommender systems in practice are used as a source of recommendations, rather than predictions [19]. Our results show that metrics that are sensitive to changes in prediction accuracy may be less sensitive to changes in recommendation accuracy. We recommend that operators who are producing mostly recommendations use a recommendations-centric metric, such as Expected Top- N Occupancy, to evaluate the effectiveness of shilling attacks on their systems. We also recommend that researchers consider focusing on recommendation accuracy, rather than prediction accuracy, in future shilling studies.

Watch Metrics, but Worry Anyway. Operators wish to know whether their recommendations systems are under attack. Watching for sharp changes in the value of traditional algorithm performance metrics such as MAE may be useful for detecting some attacks. However, our results suggest that many effective attacks will not be visible through simple aggregate metrics like MAE, so work needs to be done to develop more reliable tools to detect attackers. Note that the attacks we used in this paper could be easily detected by watching for individual users with unreasonably large numbers of ratings. Real-world attacks are likely to be more subtle.

Protect New Items. Our experiments show that new or obscure items, particularly in the user-user algorithm, are especially susceptible to attack. Recommender system operators should consider obtaining ratings for such items from a trusted source in order to make them less vulnerable. For instance, the ratings for a new item could be seeded with ratings from professional critics, with ratings from trusted volunteers, or with filterbots [8].

The research that has been done on shilling in recommender systems, including the present paper, considers attacks that are in a

⁶In fact, part of the motivation for this study is a long-term MovieLens user who is convinced that new movies are frequently shilled. Inspired by his suspicions, we looked hard for ratings behavior that looks like shilling. We have been unable to find evidence of shilling of new items in MovieLens to date.

fairly narrow region of the space of attacks spanned by our dimensions from Section 2. Here we consider which parts of that space has been covered, and suggest how future studies might cover other areas. We consider each of the dimensions for Section 2 in turn.

Along the **intent** dimension, only basic push and nuke attacks have been examined. One can imagine a world in the not-too-distant future where competing retailers, all operating recommender systems, may deliberately try to sabotage each other's systems in an attempt to frustrate and lure away customers. Attacks that accomplish this goal will likely be quite different from the attacks that have been proposed and tested thus far.

Along the **targets** dimension, the known attacks tend to target broad groups of users (in our case, all users). Our work suggests that this may be difficult under some algorithms such as item-item. We hypothesize that targeted attacks will be more effective, easier to formulate, and harder to detect.

Most attacks described to date assume a limited amount of **knowledge** about the recommender system. They are primarily targeted against **algorithms** like kNN user-user or its kin. They have little **cost/benefit analysis** beyond basic attack size, and have not been analyzed for **detectability**.

Besides formulation of additional classes of attacks, many other open questions remain in this area. Are there highly-robust ACF algorithms that are resistant to most attacks? Why are some attacks better against certain algorithms? Are there reliable ways of detecting attacks? What can system operators do if their system is attacked? Can users of recommender systems do anything to prevent or detect shilling attacks? These questions are becoming increasingly important as recommender systems become more and more commonplace in commercial applications.

6. ACKNOWLEDGMENTS

We gratefully thank the members of the GroupLens Research Project for many interesting and inspiring discussions while we carried out this work. We especially acknowledge Brad Miller for the MultiLens system, which we used for many of the experiments. We also appreciate the feedback from Dan Frankowski, Sean McNee, and Al Mamunur Rashid on early drafts of this document. This work was supported by grants from the NSF (DGE 95-54517, IIS 96-13960, IIS 97-34442, IIS 99-78717, and IIS 01-02229)

7. REFERENCES

- [1] S. E. Asch. Effects of group pressure upon the modification and distortion of judgements. *Groups, Leadership, and Men*, pages 177–190, 1951.
- [2] J. S. Breese, D. Heckerman, and C. Kadie. Empirical analysis of predictive algorithms for collaborative filtering. In *Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (UAI-98)*, pages 43–52, July 1998.
- [3] J. Canny. Collaborative filtering with privacy via factor analysis. In *IEEE Conference on Security and Privacy*, May 2002.
- [4] D. Cosley, S. K. Lam, I. Albert, J. Konstan, and J. Riedl. Is seeing believing? How recommender system interfaces affect users' opinions. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, 2003. CHI Letters 5(1).
- [5] C. Dellarocas. Immunizing online reputation reporting systems against unfair ratings and discriminatory behavior. In *ACM Conference on Electronic Commerce*, pages 150–157, 2000.
- [6] D. Goldberg, D. Nichols, B. Oki, and D. Terry. Using collaborative filtering to weave an information tapestry. *Communications of the ACM*, 35(12):61–70, 1992.
- [7] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins. Eigentaste: A constant time collaborative filtering algorithm. *Information Retrieval*, 4(2):133–151, 2001.
- [8] N. Good, B. Schafer, J. Konstan, A. Borchers, B. Sarwar, J. Herlocker, and J. Riedl. Combining collaborative filtering with personal agents for better recommendations. In *Proceedings of the 1999 Conference of the American Association of Artificial Intelligence (AAAI-99)*, July 1999.
- [9] J. Herlocker, J. Konstan, A. Borchers, and J. Riedl. An algorithmic framework for performing collaborative filtering. In *Proceedings of the 1999 Conference on Research and Development in Information Retrieval (SIGIR-99)*, Aug. 1999.
- [10] B. J. Jansen and A. Spink. An analysis of web documents retrieved and viewed. In *Internet Computing Conference*, Las Vegas, 2003.
- [11] B. Miller, J. Riedl, and J. Konstan. GroupLens for Usenet: Experiences in applying collaborative filtering to a social information system. In C. Leug and D. Fisher, editors, *From Usenet to CoWebs: Interacting with Social Information Spaces*. Springer-Verlag, 2002.
- [12] B. N. Miller. *Toward a Personal Recommender System*. PhD thesis, University of Minnesota, 2002.
- [13] C. Nass and Y. Moon. Machines and mindlessness: Social responses to computers. *Journal of Social Issues*, pages 81–103, 2000.
- [14] M. P. O'Mahony, N. Hurley, N. Kushmerick, and G. Silvestre. Collaborative recommendation: A robustness analysis. *ACM Transactions on Internet Technology*, 2003. Special Issue on Machine Learning for the Internet.
- [15] M. P. O'Mahony, N. J. Hurley, and G. C. Silvestre. Promoting recommendations: An attack on collaborative filtering. In *Proceedings of the 13th International Conference on Database and Expert Systems Applications*, pages 494–503. Springer Verlag, 2002.
- [16] P. Resnick, N. Iacovou, M. Sushak, P. Bergstrom, and J. Riedl. GroupLens: An open architecture for collaborative filtering of netnews. In *Proceedings of CSCW 1994*. ACM SIG Computer Supported Cooperative Work, 1994.
- [17] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Application of dimensionality reduction in recommender system – a case study. In *ACM WebKDD 2000 Web Mining for E-Commerce Workshop*, 2000.
- [18] B. M. Sarwar, G. Karypis, J. A. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International World Wide Web Conference (WWW10)*, Hong Kong, May 2001.
- [19] J. Schafer, J. Konstan, and J. Riedl. Electronic commerce recommender applications. *Data Mining and Knowledge Discovery*, Jan. 2001.
- [20] U. Shardanand and P. Maes. Social information filtering: Algorithms for automating 'word of mouth'. In *Proceedings of the ACM SIGCHI Conference on Human Factors in Computing Systems*, pages 210–217, 1995.
- [21] L. von Ahn, M. Blum, N. Hopper, and J. Langford. CAPTCHA: Using hard AI problems for security. In *Proceedings of Eurocrypt*, 2003, 2003.

ATTACHMENT 6

WorldCat: The thirteenth International World Wide Web Conference

OCLC		71006498		No holdings in LDL - 13 other holdings		No LDL holdings in GLIMIR cluster; 13 other holdings in GLIMIR cluster	
Books	Rec Stat	c	Entered	20050601	Replaced	20190130024329.2	
Type	a	ELvl	7	Src	Audn	Ctrl	Lang
BLvl	m	Form		Conf	Biog	MRec	Ctry
		Cont	b	GPub	LitF	Indx	
Desc	a	Ills	a	Fest	DtSt	Dates	

010 2005276576

040 DLC \$b eng \$c DLC \$d WAU \$d C\$A \$d OCLCA \$d OCLCF \$d OCLCO \$d OCLCO \$d OCL \$d OCLCO \$d AU@ \$d OCLCA \$d OCLCO

020 158113844X \$q (pbk.)

020 9781581138443 \$q (pbk.)

050 14 TK5105.888 \$b .I58 2004

090

049 LDLL

1112_ International WWW Conference \$n (13th : \$d 2004 : \$c New York, N.Y.)

245 14 The thirteenth International World Wide Web Conference : \$b proceedings.

246 1_ \$i Also known as: \$a WWW '04

260 New York : \$b Association for Computing Machinery, \$c 2004.

300 xx, 737 pages : \$b illustrations ; \$c 28 cm

336 text \$b txt \$2 rdacontent

337 unmediated \$b n \$2 rdamedia

338 volume \$b nc \$2 rdacarrier

530 Also issued online with additional title: Proceedings of the 13th International Conference on World Wide Web.

504 Includes bibliographical references and indexes.

650_0 World Wide Web \$v Congresses.

650_0 Computer networks \$v Congresses.

650_0 Hypertext systems \$v Congresses.

650_0 Internet \$v Congresses.

650_7 Computer networks. \$2 fast \$0 (OCoLC)fst00872297

650_7 Hypertext systems. \$2 fast \$0 (OCoLC)fst00965876

650_7 Internet. \$2 fast \$0 (OCoLC)fst00977184

650_7 World Wide Web. \$2 fast \$0 (OCoLC)fst01181326

655_7 Conference papers and proceedings. \$2 fast \$0 (OCoLC)fst01423772

7102_ Association for Computing Machinery.

7400_ Proceedings of the 13th International Conference on World Wide Web.

85641 \$u [http://portal.acm.org/toc.cfm?](http://portal.acm.org/toc.cfm?id=988672&coll=portal&dl=ACM&type=proceeding&idx=SERIES968&part=series&WantType=series&title=International%20World%20Wide%20Web%20Conference)

id=988672&coll=portal&dl=ACM&type=proceeding&idx=SERIES968&part=series&WantType=series&title=International%20World%20Wide%20Web%20Conference

029 1_ AU@ \$b 000040668230

Action Status	Delete Holdings _	Export _	Label _	Produce _	Update Holdings _	Validate _
---------------	-------------------	----------	---------	-----------	-------------------	------------

ATTACHMENT 7



CAPTCHA: Using Hard AI Problems for Security

International Conference on the Theory and Applications of Cryptographic Techniques

EUROCRYPT 2003: Advances in Cryptology — EUROCRYPT 2003 pp 294-311 | Cite as

- Luis von Ahn (1)
- Manuel Blum (1)
- Nicholas J. Hopper (1)
- John Langford (2)

1. Computer Science Dept., Carnegie Mellon University, , Pittsburgh, USA

2. IBM T.J. Watson Research Center, , Yorktown Heights, USA

Conference paper

First Online: 13 May 2003

- [336 Citations](#)
- [14 Mentions](#)
- 11k Downloads

Part of the [Lecture Notes in Computer Science](#) book series (LNCS, volume 2656)

Abstract

We introduce captcha, an automated test that humans can pass, but current computer programs can't pass: any program that has high success over a captcha can be used to solve an unsolved Artificial Intelligence (AI) problem. We provide several novel constructions of captchas. Since captchas have many applications in practical security, our approach introduces a new class of hard problems that can be exploited for security purposes. Much like research in cryptography has had a positive impact on algorithms for factoring and discrete log, we hope that the use of hard AI problems for security purposes allows us to advance the field of Artificial Intelligence. We introduce two families of AI problems that can be used to construct captchas and we show that solutions to such problems can be used for steganographic communication. CAPTCHAs based on these AI problem families, then, imply a win-win situation: either the problems remain unsolved and there is a way to differentiate humans from computers, or the problems are solved and there is a way to communicate covertly on some channels.

Keywords

Optical Character Recognition Distorted Text Current Computer Program

Cryptographic Protocol Image Transformation

These keywords were added by machine and not by the authors. This process is experimental and the keywords may be updated as the learning algorithm improves.

[Download](#) to read the full conference paper text

References

1. Luis von Ahn, Manuel Blum, Nicholas J. Hopper and John Langford. The CAPTCHA Web Page: <http://www.captcha.net> (<http://www.captcha.net>). 2000.
2. Luis von Ahn, Manuel Blum and John Langford. Telling Humans and Computers part (Automatically) or How Lazy Cryptographers do AI. To appear in *Communications of the ACM*.
 [Google Scholar \(\[\\)\]\(https://scholar.google.com/scholar?q=Luis%20von%20Ahn%2C%20Manuel%20Blum%20and%20John%20Langford.%20Telling%20Humans%20and%20Computers%20part%20%28Automatically%29%20or%20How%20Lazy%20Cryptographers%20do%20AI.%20To%20appear%20in%20Communications%20of%20the%20ACM.\)](https://scholar.google.com/scholar?q=Luis%20von%20Ahn%2C%20Manuel%20Blum%20and%20John%20Langford.%20Telling%20Humans%20and%20Computers%20part%20%28Automatically%29%20or%20How%20Lazy%20Cryptographers%20do%20AI.%20To%20appear%20in%20Communications%20of%20the%20ACM.)
3. Mihir Bellare, Russell Impagliazzo and Moni Naor. Does Parallel Repetition Lower the Error in Computationally Sound Protocols? In *38th IEEE Symposium on Foundations of Computer Science (FOCS' 97)*, pages 374–383. IEEE Computer Society, 1997.
 [Google Scholar \(\[\\)\]\(https://scholar.google.com/scholar?q=Mihir%20Bellare%2C%20Russell%20Impagliazzo%20and%20Moni%20Naor.%20Does%20Parallel%20Repetition%20Lower%20the%20Error%20in%20Computationally%20Sound%20Protocols%3F%20In%2038th%20IEEE%20Symposium%20on%20Foundations%20of%20Computer%20Science%20%28FOCS%E2%80%9999%2097%29%2C%20pages%20374%E2%80%9993383.%20IEEE%20Computer%20Society%2C%201997.\)](https://scholar.google.com/scholar?q=Mihir%20Bellare%2C%20Russell%20Impagliazzo%20and%20Moni%20Naor.%20Does%20Parallel%20Repetition%20Lower%20the%20Error%20in%20Computationally%20Sound%20Protocols%3F%20In%2038th%20IEEE%20Symposium%20on%20Foundations%20of%20Computer%20Science%20%28FOCS%E2%80%9999%2097%29%2C%20pages%20374%E2%80%9993383.%20IEEE%20Computer%20Society%2C%201997.)
4. Mikhail M. Bongard. *Pattern Recognition*. Spartan Books, Rochelle Park NJ, 1970.
[zbMATH](http://www.emis.de/MATH-item?0205.21201) (<http://www.emis.de/MATH-item?0205.21201>)
 [Google Scholar \(\[\\)\]\(http://scholar.google.com/scholar_lookup?title=Pattern%20Recognition&author=M.%20M..%20Bongard&publication_year=1970\)](http://scholar.google.com/scholar_lookup?title=Pattern%20Recognition&author=M.%20M..%20Bongard&publication_year=1970)
5. A. L. Coates, H. S. Baird, and R. J. Fateman. Pessimistic Print: A Reverse Turing Test. In *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR' 01)*, pages 1154–1159. Seattle WA, 2001.
 [Google Scholar \(\[\\)\]\(https://scholar.google.com/scholar?q=A.%20L.%20Coates%2C%20H.%20S.%20Baird%2C%20and%20R.%20J.%20Fateman.%20Pessimistic%20Print%3A%20A%20Reverse%20Turing%20Test.%20In%20Proceedings%20of%20the%20International%20Conference%20on%20Document%20Analysis%20and%20Recognition%20%28ICDAR%E2%80%9999%2001%29%2C%20pages%201154%E2%80%99931159.%20Seattle%20WA%2C%202001.\)](https://scholar.google.com/scholar?q=A.%20L.%20Coates%2C%20H.%20S.%20Baird%2C%20and%20R.%20J.%20Fateman.%20Pessimistic%20Print%3A%20A%20Reverse%20Turing%20Test.%20In%20Proceedings%20of%20the%20International%20Conference%20on%20Document%20Analysis%20and%20Recognition%20%28ICDAR%E2%80%9999%2001%29%2C%20pages%201154%E2%80%99931159.%20Seattle%20WA%2C%202001.)
6. Scott Craver. On Public-key Steganography in the Presence of an Active Warden. In *Proceedings of the Second International Information Hiding Workshop*, pages 355–368. Springer, 1998.

- Google Scholar ([\)](https://scholar.google.com/scholar?q=Scott%20Craver.%20On%20Public-key%20Steganography%20in%20the%20Presence%20of%20an%20Active%20Warden.%20In%20Proceedings%20of%20the%20Second%20International%20Information%20Hiding%20Workshop%2C%20pages%20355%E2%80%93368.%20Springer%2C%201998.)
7. Nicholas J. Hopper, John Langford and Luis von Ahn. Provably Secure Steganography. In *Advances in Cryptology, CRYPTO' 02*, volume 2442 of *Lecture Notes in Computer Science*, pages 77–92. Santa Barbara, CA, 2002.
Google Scholar ([\)](http://scholar.google.com/scholar_lookup?title=Provably%20Secure%20Steganography&author=N.%20J..%20Hopper&author=J..%20Langford&author=L.%20Ahn&pages=77-92&publication_year=2002)
 8. M. D. Lillibridge, M. Adabi, K. Bharat, and A. Broder. Method for selectively restricting access to computer systems. Technical report, US Patent 6,195,698. Applied April 1998 and Approved February 2001.
Google Scholar ([\)](https://scholar.google.com/scholar?q=M.%20D.%20Lillibridge%2C%20M.%20Adabi%2C%20K.%20Bharat%2C%20and%20A.%20Broder.%20Method%20for%20selectively%20restricting%20access%20to%20computer%20systems.%20Technical%20report%2C%20US%20Patent%206%2C195%2C698.%20Applied%20April%201998%20and%20Approved%20February%202001.)
 9. Greg Mori and Jitendra Malik. Breaking a Visual CAPTCHA. Unpublished Manuscript, 2002. Available electronically:
<http://www.cs.berkeley.edu/~mori/gimpy/gimpy.pdf>
(<http://www.cs.berkeley.edu/~mori/gimpy/gimpy.pdf>).
 10. Moni Naor. Verification of a human in the loop or Identification via the Turing Test. Unpublished Manuscript, 1997. Available electronically:
<http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps>
(<http://www.wisdom.weizmann.ac.il/~naor/PAPERS/human.ps>).
 11. Benny Pinkas and Tomas Sander. Securing Passwords Against Dictionary Attacks. In *Proceedings of the ACM Computer and Security Conference (CCS' 02)*, pages 161–170. ACM Press, November 2002.
Google Scholar ([\)](https://scholar.google.com/scholar?q=Benny%20Pinkas%20and%20Tomas%20Sander.%20Securing%20Passwords%20Against%20Dictionary%20Attacks.%20In%20Proceedings%20of%20the%20ACM%20Computer%20and%20Security%20Conference%20%28CCS%E2%80%9999%2002%29%2C%20pages%20161%E2%80%93170.%20ACM%20Press%2C%20November%202002.)
 12. S. Rice, G. Nagy, and T. Nartker. *Optical Character Recognition: An Illustrated Guide to the Frontier*. Kluwer Academic Publishers, Boston, 1999.
Google Scholar ([\)](http://scholar.google.com/scholar_lookup?title=Optical%20Character%20Recognition%3A%20An%20Illustrated%20Guide%20to%20the%20Frontier&author=S..%20Rice&author=G..%20Nagy&author=T..%20Nartker&publication_year=1999)
 13. Adi Shamir and Eran Tromer. Factoring Large Numbers with the TWIRL Device. Unpublished Manuscript, 2003. Available electronically:
<http://www.cryptome.org/twirl.ps.gz> (<http://www.cryptome.org/twirl.ps.gz>).

14. J. Xu, R. Lipton and I. Essa. Hello, are you human. Technical Report GIT-CC-00-28, Georgia Institute of Technology, November 2000.
[Google Scholar](https://scholar.google.com/scholar?q=J.%20Xu%2C%20R.%20Lipton%20and%20I.%20Essa.%20Hello%2C%20are%20you%20human.%20Technical%20Report%20GIT-CC-00-28%2C%20Georgia%20Institute%20of%20Technology%2C%20November%202000.) (https://scholar.google.com/scholar?q=J.%20Xu%2C%20R.%20Lipton%20and%20I.%20Essa.%20Hello%2C%20are%20you%20human.%20Technical%20Report%20GIT-CC-00-28%2C%20Georgia%20Institute%20of%20Technology%2C%20November%202000.)

Copyright information

© International Association for Cryptologic Research 2003

About this paper

Cite this paper as:

von Ahn L., Blum M., Hopper N.J., Langford J. (2003) CAPTCHA: Using Hard AI Problems for Security. In: Biham E. (eds) Advances in Cryptology — EUROCRYPT 2003. EUROCRYPT 2003. Lecture Notes in Computer Science, vol 2656. Springer, Berlin, Heidelberg. https://doi.org/10.1007/3-540-39200-9_18

- First Online 13 May 2003
- DOI https://doi.org/10.1007/3-540-39200-9_18
- Publisher Name Springer, Berlin, Heidelberg
- Print ISBN 978-3-540-14039-9
- Online ISBN 978-3-540-39200-2
- eBook Packages [Springer Book Archive](#)
- [Buy this book on publisher's site](#)
- [Reprints and Permissions](#)

Personalised recommendations

SPRINGER NATURE

© 2020 Springer Nature Switzerland AG. Part of [Springer Nature](#).

Not logged in Not affiliated 76.84.212.16

ATTACHMENT 8

US PATENT & TRADEMARK OFFICE

PATENT APPLICATION FULL TEXT AND IMAGE DATABASE



(14 of 16)

United States Patent Application**20070192849****Kind Code****A1****Golle; Philippe ; et al.****August 16, 2007**

PHYSICAL TOKEN FOR SUPPORTING VERIFICATION OF HUMAN PRESENCE IN AN ONLINE ENVIRONMENT

Abstract

A physical token operates together with an online server to test whether a user operating a user computer to access a service available at the online server is human. The physical token has two states: inactive and active. When in the inactive state the physical token displays interaction instructions for performing a physical action on the token. When the interaction instructions are satisfied, the physical token transitions to the active state and displays a response that may be used for satisfying a challenge issued by the online server to the user seeking to access the service.

Inventors: **Golle; Philippe;** (*San Francisco, CA*) ; **Ducheneaut; Nicolas B.;** (*Mountain View, CA*)

Correspondence **PATENT DOCUMENTATION CENTER**

Address: **XEROX CORPORATION**

100 CLINTON AVENUE SOUTH, XEROX SQ. 20 TH FLOOR

ROCHESTER

NY

14644

US

Assignee: **PALO ALTO RESEARCH CENTER INCORPORATED**

3333 Coyote Hill Road

Palo Alto

CA

Family ID: **38370295**

Appl. No.: **11/276055**

Filed: **February 10, 2006**

Current U.S. Class:**726/16****Current CPC Class:**

G06F 21/34 20130101; G06F 21/36 20130101; G06F 2221/2151
20130101; G06F 2221/2103 20130101

Class at Publication:**726/016**

12. An apparatus for verifying human presence in an online environment, the apparatus comprising: a first computational unit providing a service and having a key and a synchronization measure; a second computational unit communicating with the first computational unit; a third computational unit having a synchronization measure and a key shared with the first computational unit; wherein each computational unit further comprises (i) a memory for storing processing instructions and (ii) a processor coupled to the memory for executing the processing instructions; wherein the processing instructions of the first computational unit further comprises processing instructions for (i) issuing a challenge from a first computational unit to a second computational unit after the second computational unit communicates a request to access a service provided by the first computational unit and (ii) enabling, at the first computational unit, access to the service to the second computational unit after the challenge is satisfied when a response received from the second computational unit is authenticated using a function that takes as input the response and the key shared with the third computational unit; wherein the processing instructions of the third computational unit further comprises a user interface and processing instructions for operating in (i) a first state for computing a response using a function that takes as input a synchronization measure and a key shared with the first computational unit; the computed response being computed for satisfying the challenge issued by the first computational unit to the second computational unit, and (ii) a second state for displaying the computed response on a user interface of the third computational unit only if interaction instructions involving a physical challenge output on the user interface are satisfied; wherein the processing instructions of the second computational unit further comprises a user interface and processing instructions for (i) receiving input at the second computational unit representing the response displayed on a user interface of the third computational unit, and (ii) transmitting the input received representing the response from the second computational unit to the first computational unit for satisfying the challenge issued by the first computational unit.

14. The apparatus according to claim 13, wherein the key shared between the first computational unit and the third computational unit is a symmetric key.

16. The apparatus according to claim 15, wherein the ciphertext received from the second computational unit is deciphered using a symmetric cipher that takes as input the ciphertext and the symmetric key shared with the third computational unit.

18. The apparatus to claim 12, wherein the response is displayed on a display touchpad of the third computational unit.

20. The apparatus according to claim 18, wherein interaction instructions issued on the display touchpad specify tracing over a shape displayed on the display touchpad.

21. The apparatus according to claim 20, wherein the service provided at the first computational agent is a game.

22. The apparatus according to claim 12, wherein the third computational unit has no input connectivity with the first computational unit or the second computational unit.

Description

CROSS REFERENCE TO RELATED APPLICATION

[0001] Cross-reference is made to U.S. Patent Application filed concurrently herewith and entitled "XOR Encoded Document For Secure Message Exchange" (Docket No. 20050701-US-NP), which is assigned to the same assignee as the present invention and incorporated herein by reference.

BACKGROUND AND SUMMARY

[0002] The following relates generally to methods, unit and articles of manufacture therefor, for verifying human presence in an online environment.

[0003] There exist a variety of automated tests for telling humans and computers apart in an online environment. Such tests are collectively known as CAPTCHA (Completely Automated Public Turing Test to Tell Computers and Humans Apart) tests. A CAPTCHA test is a program that can generate and grade tests that most humans can pass but that current computer programs cannot. CAPTCHA tests are of necessity heuristic; there is no way to prove that a computer program cannot pass a test that a human can pass. CAPTCHA tests are adapted to exploit gaps between the abilities of humans and computers.

[0004] CAPTCHA tests may be used at the start of or during an online application, such as a multi-player game, to verify human presence as opposed to an automated operator (otherwise known as a "bot" or automated player agent). The CAPTCHA test may or may not be directly integrated with the online application. CAPTCHA tests that are not integrated with online applications are referred to herein as out-of-band CAPTCHA tests. Such out-of-band CAPTCHA tests are flexible in that they may be used with a varied number of online applications. CAPTCHA tests that are tightly integrated into an application (such as a game) are less likely (depending on their method) to be disruptive to the user.

[0005] A wide range of CAPTCHA testing strategies is possible. In multi-player game applications for example, a single CAPTCHA challenge may be presented when a player joins a game; alternatively, repeated CAPTCHA challenges may be presented to the player at random intervals over the course of a game. The game may on occasion allow a player not to reply or to give an incorrect answer. A player that fails one or several challenges may be suspended from the game, temporarily for a limited period of time, or permanently (in which case all progress made in the game is lost). CAPTCHA tests deter bots most effectively in long stateful games where players have a lot to lose if they fail a test; games played in short, stateless rounds may require more frequent tests.

[0006] A drawback with known CAPTCHA tests is that the tests may be delegated by an automated user to a user that is a human to perform the test. For example, with multi-player online games, know CAPTCHA test may be solved by a human, but that human need not be the user playing the game. That is, known CAPTCHA tests are susceptible to relay attacks in which automated player agents direct digital CAPTCHA tests that require human evaluation to a user that is human to solve the test. The cost of setting up a relay attack may be prohibitive to solve a single digital CAPTCHA test (e.g., when opening a new email account), but it becomes negligible if it can be amortized over many tests, as in online games.

[0007] Additional background on the use of CAPTCHA tests and their limitations is disclosed in the following publications which are incorporated herein by reference: Luis von Ahn, Manuel Blum, Nicholas J. Hopper, and John Langford, "CAPTCHA: Using Hard AI Problems For Security", Advances in Cryptology--

EUROCRYPT'03, Vol. 2656, pp. 294-311, 2003; and Philippe Golle and Nicolas Ducheneaut, "Preventing Bots from Playing Online Games", in the ACM Computers in Entertainment, Vol. 3, July, 2005, which is incorporated herein by reference.

[0008] Accordingly, there continues to be a need for improved CAPTCHA test for distinguishing bots from human players in various online applications. Such improved CAPTCHA tests would advantageously be computationally resistant (i.e., are not readily carried out by a bot without undue expense), not readily outsourced (i.e., not readily carried out by a third-party human), and non-disruptive (i.e., do not deleteriously disrupt a human's interaction with the online application).

[0009] In accordance with the various embodiments disclosed herein, there is provided a computer implemented method, apparatus and article of manufacture therefor, for verifying human presence in an online environment using a CAPTCHA (or physical) token. In operation, a challenge is issued from an application server (e.g., first computational unit) to a user computer (e.g., second computational unit) after the user computer communicates a request to access a service provided by the application server. A response is computed at the CAPTCHA token (e.g., third computational unit) using a function that takes as input a synchronization measure and a key shared with the application server. The computed response is computed for satisfying the challenge issued by the application server to the user computer and displayed on a user interface of the CAPTCHA token only if interaction instructions involving a physical challenge output on the user interface are satisfied. Input is received at the user computer representing the response displayed on a user interface of the CAPTCHA token. The input received representing response from the user computer is transmitted to the application server for satisfying the challenge issued by the application server. Access to the service to the user computer is enabled by the application server after the challenge is satisfied. The challenge is satisfied at the application server when the response received from the user computer is authenticated using a function that takes as input the response and the key shared with the CAPTCHA token.

[0010] In accordance with one of the various embodiments disclosed herein: the each synchronization measure shared between the application server and the user computer is one of a measure of time output by a clock, a counter, and a nonce output by a computational unit; the key shared between the application server and the user computer is a key for a symmetric cipher; the function is computed at the CAPTCHA token with a symmetric cipher that takes as input a synchronization measure and the symmetric key shared with the application server; and the response received from the user computer is deciphered by the application server using a symmetric cipher that takes as input the ciphertext response and the symmetric key shared with the CAPTCHA token.

BRIEF DESCRIPTION OF THE DRAWINGS

[0011] These and other aspects of the disclosure will become apparent from the following description read in conjunction with the accompanying drawings wherein the same reference numerals have been applied to like parts and in which:

[0012] FIG. 1 illustrates an example environment in which a CAPTCHA token is used;

[0013] FIG. 2 sets forth a series of operations for allowing an application server to verify the presence of human user operating a user computer using a CAPTCHA token;

[0014] FIGS. 3 and 4 illustrate a first embodiment of a user interface of a CAPTCHA token; and

[0015] FIGS. 5 and 6 illustrate a second embodiment of a user interface of a CAPTCHA token.

DETAILED DESCRIPTION

[0016] A. Definition of Terms

[0017] The terms defined below have the indicated meanings throughout this application, including the claims

and the figures:

[0018] "Physical CAPTCHA" is used herein to mean an apparatus having one or more of a combination of user input devices (e.g., a keyboard, mouse, joystick, touchpad, keypad, speaker, camera, and touch screen) that accepts physical input (e.g., motion, pressure, sound, etc.) and produces digital output (or response). A user of the physical CAPTCHA may only learn of the digital output at some given time after the user has physically produced input using a user input device. To prove the digital output arrived at a given time, the output must be capable of being authenticated.

[0019] A physical CAPTCHA is "tamper-proof" if it loses its ability to produce digital output that is capable of being authenticated when it is tampered with. For example, any attempt to bypass an interface (e.g., by opening a device and rewiring its microcontroller), such as its user input device, may cause a physical CAPTCHA to lose its authentication key.

[0020] "CAPTCHA token" and "CAPTCHA input device" are used herein to mean different embodiments of a physical CAPTCHA.

[0021] B. Example Operating Environment of the CAPTCHA Token

[0022] FIG. 1 illustrates an example environment for using a CAPTCHA token 102 (also referred to as a third computational unit). The CAPTCHA token 102 is a physical CAPTCHA for supporting the verification of human presence in an online environment, such as, between a human user 104 operating user computer 106 (also referred to as second computational unit) and an application server 108 (also referred to as a first computational unit). The user computer 106 (with user input devices 107) and the application server 108 have an established communication channel 110, whereas the CAPTCHA token has, in one embodiment, no input connectivity with the user computer 106 or the application server 108. In operation, the CAPTCHA token 102 is used by the user 104 to satisfy interaction instructions $i.sub.t$ that involve a physical challenge before receiving a response for satisfying a digital challenge $c.sub.t$ issued by the application server 108.

[0023] By way of overview, FIG. 2 sets forth a series of operations for allowing the application server 108 to verify the presence of human user 104 operating the user computer 106 using the CAPTCHA token 102. At 202, when user computer 106 requests access to the application 112, it is issued a challenge ct by the application server 108 and is prompted to enter a response $s.sub.t$ (e.g., ciphertext or cryptographic digest st that is displayed at the user interface 120 of the CAPTCHA token 102 when in an active state). At 204, the CAPTCHA token 102 specifies interaction instructions $i.sub.t$ (when in an inactive state) to be carried out (by the user 104) for access to the response $s.sub.t$. If the interaction instructions $i.sub.t$ are not followed correctly at 206, the CAPTCHA token 102 (remains in an inactive state and) reissues or issues new interaction instructions (e.g., $i.sub.t$ or $i.sub.t+1$) to be satisfied for access to the response $s.sub.t$ at 204; otherwise, when the interaction instructions $i.sub.t$ are followed correctly (at 206), the CAPTCHA token 102 (transitions to an active state and) displays the response $s.sub.t$ requested to satisfy the challenge $c.sub.t$ issued by the application server 108 for a given time interval (at 208). At 210, the user supplies (either manually or automatically) the displayed response at the user computer 106 to satisfy the challenge $c.sub.t$ issued by the application server 108 for access to the application 112.

[0024] More specifically (at 202), the application server 108 offers to human user 104 through user computer 106 access to an application 112. In one embodiment, the application 112 is a multi-player online game, where the human user 104 seeks to be (or continue to be) a participant in the game through a local presence running on user computer 106 for exchanging data with the application 112 operating on the application server 108. The function of the CAPTCHA token 102 is to enable the application 112 to test using a physical action component whether the user 104 operating the local presence at the user computer 106 is human. The CAPTCHA token 102 does not serve as input to the application 112, and thus may be used in performing a similar function with a number of different applications. That is, the CAPTCHA token 102 provides input to the application 112 that is response-specific (i.e., for the purpose of satisfying a response to a challenge) but not application-specific and thus may be used to perform a similar function with different applications.

[0025] The CAPTCHA token 102 may take on a number of forms for performing the function of testing using a physical action component. In the different forms, the CAPTCHA token 102 includes a synchronization measure (e.g., clock 114, a counter, or a nonce supplied by a computational unit), memory for storing a shared key (e.g., symmetric key memory 116), a cryptographic processing or computational unit (e.g., symmetric cipher computational unit 118, or a hash function computational unit), and a user interface 120.

[0026] FIGS. 3-4 illustrate one embodiment and FIGS. 4-6 illustrate another embodiment of different forms that the user interface 120 may take to enable the physical action component of the CAPTCHA test. The user interface 120 may be equipped with one or more of a combination of a keypad, touchpad, screen, microphone, accelerometer, or speaker to request and measure performance of a physical action. In one embodiment, the CAPTCHA token is tamper-proof as data may only be entered (or input) to the CAPTCHA token through a user interface with keys that must be depressed on a keypad or with identified areas on a touchpad that must be depressed.

[0027] More specifically (at 204), FIGS. 3 and 4 illustrate a first embodiment and FIGS. 5 and 6 illustrate a second embodiment in which the CAPTCHA token 102 is in a first state in FIGS. 3 and 5 and a second state in FIGS. 4 and 6. In FIG. 3, the CAPTCHA token 102 includes a user interface 120 in which the user 104 is issued interaction instructions in first area 302 that request a user to perform the physical action of typing in a series of numbers into touchpad in second area 304 to activate the CAPTCHA token 102. Once the interaction instructions $i_{sub.t}$ are satisfied, the CAPTCHA token 102 is activated and a current response $s_{sub.t}$ is displayed in the first area 302 of the user interface 120 as shown in FIG. 4. In FIG. 5, the CAPTCHA token 102 includes a user interface 120, in which the user 104 is issued interaction instructions in first area 502 that request a user to perform the physical action of tracing a curve on a touchpad in second area 504 using, for example, a stylus to activate the CAPTCHA token 102. Once the interaction instructions $i_{sub.t}$ are satisfied, the CAPTCHA token 102 is activated and a current response $s_{sub.t}$ is displayed in the first area 502 of the user interface 120 as shown in FIG. 6.

[0028] In alternate embodiments, the user interface 120 of the CAPTCHA token 102 includes one or a plurality of displays, accelerometers, keypads, screens, microphones, and/or touchpads that operate as an integral unit or as independent units to issue a series of interaction instructions that need to be satisfied with a physical action before a current response $s_{sub.t}$ may be displayed. Each time interaction instructions are issued on the user interface 120, they may appear static (e.g., a keypad that does not change the location of its digits each time new interaction instructions are issued) or dynamic (e.g., a touchpad that changes the location of its digits each time new interaction instructions are issued).

[0029] When in operation, the CAPTCHA token 102 may be in one of two states: active or inactive. When in an active state, the CAPTCHA token displays on its user interface 120 the value $s_{sub.t}$ computed by a function E 118 (shared with the application server 108) that takes as input a current synchronization measure t produced, for example, by clock 114, a counter (not shown), or a value (a nonce) output by computational unit, and key shared with the application server 108. In one embodiment, the value $s_{sub.t}$ is updated each interval the time t is incremented. When inactive, the CAPTCHA token displays a random instruction $i_{sub.t}$, which is referred to herein as interaction instructions $i_{sub.t}$. The CAPTCHA token 102 transitions from its inactive state to its active state for a set time interval at switch 122, when the user 104 enters input at the user interface 120 that satisfies the current interaction instruction $i_{sub.t}$ displayed at the user interface 120.

[0030] In one embodiment (at 208), the CAPTCHA token 102 remains in the active state once the interaction instructions it are satisfied for a time interval on the order of one to two minutes. In other embodiments, the time interval varies depending, for example, on one or more factors such as the application. Once the time interval expires, the CAPTCHA token 102 transitions to the inactive state, and re-transitions back to the active state only once the currently issued interaction instructions $i_{sub.t}$ are re-satisfied. In satisfying or re-satisfying the interaction instructions $i_{sub.t}$, the user is not required to recall a secret code; the purpose of the interaction instructions $i_{sub.t}$ are to test (or challenge) on the basis of a specified physical action whether the user of the CAPTCHA token 102 is human as opposed to an automated computational agent (or bot).

[0031] Once the interaction instructions $i.sub.t$ are satisfied and the response $s.sub.t$ displayed on the user interface 120 of the CAPTCHA token 102 (at 208), the response $s.sub.t$ may be used (at 210) by the user 104 to authenticate a human presence by delivering the response $s.sub.t$ to the application server 108 to satisfy its issued challenge $c.sub.t$. In one embodiment, once the CAPTCHA token 102 is placed in an active state by the user 104 after satisfying the current interaction instructions $i.sub.t$, the user 104 enters the response $s.sub.t$ displayed at the user interface 120 manually using a user input device 107 of the user computer 106 to satisfy a challenge $c.sub.t$ issued by the application server 108. In an alternate embodiment in which there exists some form of output connectivity between the CAPTCHA token 102 and the user computer 106, the response $s.sub.t$ is automatically transmitted to the application server once interaction instructions $i.sub.t$ are satisfied. Upon receipt of the response $s.sub.t$, the application server 108 tests the response $s.sub.t$ received from the user computer 106 by performing cryptographic computations using computation unit 119.

[0032] C. Response Authentication

[0033] In preparing the response $s.sub.t$ for display (at 206), the CAPTCHA token 102 prepares one of a sequence of numbers ($s.sub.1, s.sub.2, s.sub.3, \dots$) that may each be authenticated by the application server 108. In one embodiment, each number s in the sequence of numbers is ciphertext that has a limited number of digits (e.g., 5). In an alternate embodiment, each number s is cryptographic hash (or one-way hash). In either embodiment, the knowledge of values ($s.sub.1, s.sub.2, s.sub.3, \dots s.sub.t$) generally does not aid in learning value $s.sub.t+1$.

[0034] In one embodiment (at 208), the response $s.sub.t$ is produced from sequence of ciphertext ($s.sub.1, s.sub.2, s.sub.3, \dots$) computed using a symmetric cipher E 118 defined as $s.sub.t = E.sub.k(t)$ (e.g., that follows the Advanced Encryption Standard (AES)), where k is a symmetric key stored in symmetric key memory 116 and t is a current time (e.g., measured in minutes) produced by clock 114 (that is initially synchronized with clock 115 of application server 108). During authentication (at 210), the ciphertext $s.sub.t$ is deciphered to plaintext $t.sub.s$ using a symmetric cipher E (that is shared with the symmetric cipher E of the CAPTCHA token 102) defined as $t.sub.s = E.sub.k(s.sub.t)$, where k is a symmetric key stored in symmetric key memory 117 (that is the same as the symmetric key 116 of the CAPTCHA token 102). The challenge ct issued by the application server 108 is satisfied when the deciphered plaintext $t.sub.s$ is found by comparator 124 to be within a given range of (i.e., or approximates) the current time t output by clock 115 (that is synchronized with the clock 114 of the CAPTCHA token 102). Once the challenge $c.sub.t$ is satisfied the application server 108 allows the user computer 106 to access the application 112.

[0035] In an alternate embodiment (at 208), the response $s.sub.t$ is produced from a sequence of one-way hash ($s.sub.1, s.sub.2, s.sub.3, \dots$) computed using a cryptographic hash function. One example of a cryptographic hash function, which has the properties of one-wayness (i.e., irreversibility) and collision-resistance, is the Secure Hash Algorithm (SHA-1) that is specified in the Secure Hash Standard (SHS), which may be used to generate as output the response $s.sub.t$ from a secret key and a synchronization measure (such as a counter) that is shared by the CAPTCHA token 102 and the application server 108. In this alternate embodiment, both the CAPTCHA token 102 and the application server 108 each produce a similar one-way hash from a shared secret key and a shared synchronization measure using a shared cryptographic hash function. The challenge is satisfied when the two are similar.

[0036] D. CAPTCHA Input Devices

[0037] Another embodiment of a physical CAPTCHA is a physical CAPTCHA input device (e.g., a joystick) that may be used on its own or in combination with a CAPTCHA token for verifying human presence in an online environment. The physical CAPTCHA input device combines two functionalities: (1) it records an input stream produced by physical action that is relayed to an online application (e.g., application 112); and (2) it ensures that the input stream is authenticated and comes from a tamper-proof input device. Effectively, such a physical CAPTCHA input device is used to mediate physical action applied to the input device between the application and a human user. Advantageously, a human user is adapted to exert such physical action to such a

physical CAPTCHA input device faster and more reliably than a machine. Thus, unlike the CAPTCHA token which servers to ensure human presence, CAPTCHA input devices serve also to ensure human play (i.e., that all interactions come from a human user). For example, joysticks and other small game input devices may be modified to be tamper-proof and include an authentication mechanism. Applications (e.g., an online multi-player game) that use such a modified joystick (or other input device) would ensure that there is no other way for a human user (e.g., player) to send inputs to the application than by physical action with the joystick (e.g., pressing, pushing, etc.).

[0038] E. Miscellaneous

[0039] To recapitulate, there has been disclosed hardware-based "physical" CAPTCHA tests for allowing an online application to distinguish a human user from an automated agent based on the human's ability to interact with the physical token or input device. While human players may readily perform physical actions such as pressing a button, moving a joystick, tapping or tracing on a touch-sensitive area, automation of such physical actions is difficult and costly depending on the action. Advantageously over digital CAPTCHA tests, CAPTCHA tests with a physical action component may be resistant to relay attacks, may be less intrusive, and may require all interaction with an online application to originate from a human user.

[0040] More specifically, the CAPTCHA token is resistant to relay attacks because they may not be readily delegated by an automated player agent as may a digital CAPTCHA test. In addition, the CAPTCHA token is also resistant to delegation from one user who is human to another user who is human because a CAPTCHA token may not be forwarded to a plurality of locations at one time, as may a digital CAPTCHA test through an electronic medium using email or the like. In addition, the CAPTCHA token is also resistant to delegation because of the logistics, complexity, time and trust involved in mailing a physical object from one location to another that may, for example, be required to be used over a period of time to satisfy one or more challenges and that may be required to be returned to or exchanged with the application provider either once the application is no longer used by the user or on a period basis or randomly on demand (which each may be used as an additional challenge to the user).

[0041] In one embodiment disclosed herein, the physical CAPTCHA is a token that computes an authenticated sequence of time-dependent responses. To learn the current response of the token, a user operating in an online environment must physically interact with the token in a prescribed manner defined by randomly issued interaction instructions. The physical interaction prescribed by the interaction instructions varies depending on the form factor of the CAPTCHA token. For example, the form factor may include any one or more of a combination of: depressing keys on a keypad, tapping or tracing on a sensitive pad or screen, etc. Upon satisfying the interaction instructions issued by the token, the token makes available the current time-dependent response for a defined period of time, which the user submits as proof of being human when responding to a challenge issued by an online application.

[0042] Additional information concerning the disclosed hardware-based "physical" CAPTCHA token is described in the following publications incorporated herein in their entirety by reference: Philippe Golle and Nicolas Ducheneaut, "Preventing Bots from Playing Online Games", in the ACM Computers in Entertainment, Vol. 3, July, 2005; and Philippe Golle and Nicolas Ducheneaut, "Keeping bots out of online games", ACM SIGCHI International Conference on Advances in Computer Entertainment Technology, Valencia Spane, Jun. 15-17, 2005.

[0043] Those skilled in the art will recognize that a general purpose computer may be used as an apparatus for implementing the computational units of the operating environment shown in FIG. 1 and described herein. Such a general purpose computer would include hardware and software. The hardware would comprise, for example, memory (ROM, RAM, etc.) (e.g., for storing processing instructions of the CAPTCHA token, the user computer, and the application server), a processor (i.e., CPU) (e.g., coupled to the memory for executing the processing instructions of the CAPTCHA token, the user computer, and the application server), persistent storage (e.g., CD-ROM, hard drive, floppy drive, tape drive, etc.), user I/O, and network I/O. The user I/O (or user input devices) may include a camera, a microphone, speakers, a keyboard, a pointing device (e.g., pointing stick, mouse, etc.),

and a display. The network I/O (such as connection 110) may for example be coupled to a network such as the Internet. The software of the general purpose computer would include an operating system and application software providing the functions performed in the example operating environment shown and described in FIG. 1.

[0044] Further, those skilled in the art will recognize that the forgoing embodiments may be implemented as a machine (or system), process (or method), or article of manufacture by using standard programming and/or engineering techniques to produce programming software, firmware, hardware, or any combination thereof. It will be appreciated by those skilled in the art that the flow diagram described in the specification is meant to provide an understanding of different possible embodiments. As such, alternative ordering of the steps, performing one or more steps in parallel, and/or performing additional or fewer steps may be done in alternative embodiments.

[0045] Any resulting program(s), having computer-readable program code, may be embodied within one or more computer-usable media such as memory devices or transmitting devices, thereby making a computer program product or article of manufacture according to the embodiment described herein. As such, the terms "article of manufacture" and "computer program product" as used herein are intended to encompass a computer program existent (permanently, temporarily, or transitorily) on any computer-usable medium such as on any memory device or in any transmitting device.

[0046] Executing program code directly from one medium, storing program code onto a medium, copying the code from one medium to another medium, transmitting the code using a transmitting device, or other equivalent acts may involve the use of a memory or transmitting device which only embodies program code transitorily as a preliminary or final step in making, using, or selling the embodiments as set forth in the claims.

[0047] Memory devices include, but are not limited to, fixed (hard) disk drives, floppy disks (or diskettes), optical disks, magnetic tape, semiconductor memories such as RAM, ROM, Proms, etc. Transmitting devices include, but are not limited to, the Internet, intranets, electronic bulletin board and message/note exchanges, telephone/modem based network communication, hard-wired/cabled communication network, cellular communication, radio wave communication, satellite communication, and other stationary or mobile network systems/communication links.

[0048] A machine embodying the embodiments may involve one or more processing systems including, but not limited to, CPU, memory/storage devices, communication links, communication/transmitting devices, servers, I/O devices, or any subcomponents or individual parts of one or more processing systems, including software, firmware, hardware, or any combination or subcombination thereof, which embody the disclosure as set forth in the claims.

[0049] While particular embodiments have been described, alternatives, modifications, variations, improvements, and substantial equivalents that are or may be presently unforeseen may arise to applicants or others skilled in the art. Accordingly, the appended claims as filed and as they may be amended are intended to embrace all such alternatives, modifications variations, improvements, and substantial equivalents.

* * * * *



ATTACHMENT 9

(19)



(11)

EP 1 845 468 A1

(12)

EUROPEAN PATENT APPLICATION

(43) Date of publication:

17.10.2007 Bulletin 2007/42

(51) Int Cl.:

G06F 21/00 (2006.01)**H04L 29/06 (2006.01)**(21) Application number: **06007693.2**(22) Date of filing: **12.04.2006**

(84) Designated Contracting States:

AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HU IE IS IT LI LT LU LV MC NL PL PT RO SE SI SK TR

Designated Extension States:

AL BA HR MK YU

(71) Applicants:

- **Deutsche Telekom AG**
53113 Bonn (DE)
- **Technische Universität Berlin**
10623 Berlin (DE)

(72) Inventors:

- **Karrer, Roger, Dr.**
10555 Berlin (DE)
- **Kühn, Ulrich, Dr.**
10110 Berlin (DE)

(74) Representative: **Blumbach - Zinngrebe**

Patentanwälte
Alexandrastrasse 5
65187 Wiesbaden (DE)

(54) Method, computer network system and gate for identifying and controlling unsolicited traffic

(57) It is a major aspect of the present invention to identify and control unsolicited traffic, in particular by limiting or entirely blocking unsolicited traffic in an access network (107) or in a core network (104), e. g. the Internet.

This aim is achieved by at least one gate (109), wherein each gate is located between an access network (105, 107) and the core network (109). The gate (109) operates to identify and control unsolicited traffic received from malicious sources (102-1 to 102-N) and to forward permissions or capabilities to sources (101-1 to

101-N; 102-1 to 102-N) wishing to send data by using an authentication authority (113). These functions performed by the gate (109) and the authentication authority (113) are also called edge-based capability (EC) functions. Each source, e. g. a user, a web client or a host computer wanting to send data to a destination (103-1) must receive a permission to send prior to sending. This permission is used to compute a mark for every packet so that the gate (109) is able to distinguishing authenticated (111) from unsolicited (112) traffic and thus to allow only authenticated traffic to pass the gate (109).

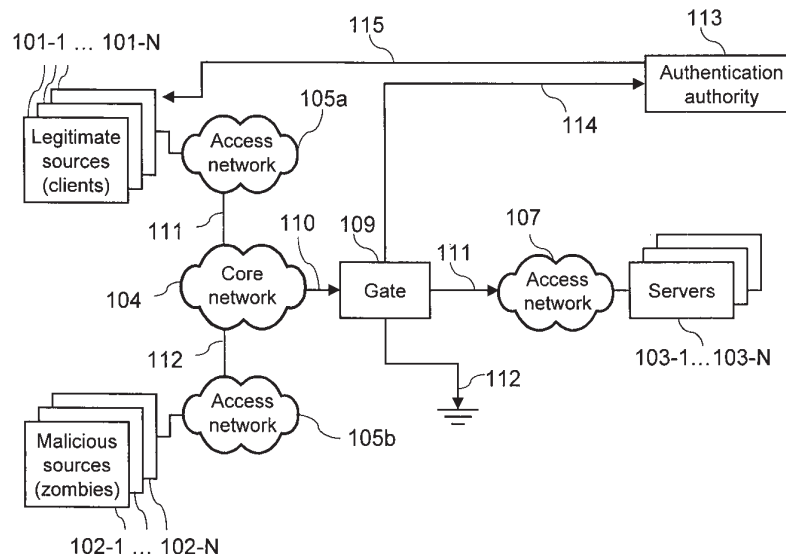


Fig. 2

EP 1 845 468 A1

DescriptionTechnical Field

5 **[0001]** The present invention relates to a method for identifying and controlling unsolicited traffic in a computer network system which comprises access networks for connecting sources and destinations, respectively, to a core network, like the Internet. The present invention further relates to a computer network system and a gate adapted to perform a method for identifying and controlling unsolicited traffic.

10 Background of the Invention

[0002] The open design of the Internet has been a key factor for its growth and success. However, the ability to send anything to anybody at any time has shown to have severe drawbacks. The lack of authentication and security is increasingly exploited to inject unsolicited traffic into the network, leading to uncontrolled virus and worm spread and devastating Distributed Denial-of-Service (DDoS) attacks.

15 **[0003]** Viruses and worms have been shown to clog entire parts of the Internet by causing extensive traffic. Moreover, they can carry code that modifies or destroys infected end systems up to a complete loss of data or steal private data.

[0004] DDoS attacks aim at overloading a resource such that legitimate users are denied access to the resource. Well-known DDoS targets include, but are not limited to, Web servers and Internet Data Centers.

20 **[0005]** The damage caused by unsolicited traffic has rapidly increased in the past years and its damage exceeds today billions of dollars.

[0006] There already exist approaches which aim to protect end systems or end devices against unsolicited traffic such as Denial of Service Attacks, worms or virus.

25 **[0007]** The present invention aims at protecting network resources. In particular, access networks that have a significantly lower capacity than the core must be protected against unsolicited traffic. Protection includes the prevention of flooding attacks with unsolicited traffic (DDoS), such that legitimate traffic is no longer able to reach its destination due to exceeded network capacity, as well as the early prevention of unsolicited (malicious or infectious) traffic spreading into the core to infect other systems.

30 **[0008]** One known approach is based on statistical methods as, for example, disclosed by Y. Kim et al. in "Packetscore: statistics-based overload control against distributed denial-of-service attacks", Proceedings of IEEE INFOCOM'04, Hong Kong, March 2004.

[0009] Statistical methods require no user interaction that might annoy the user. However, the increasing sophistication of attacks, such as mimicking attacks as flash crowds or attack with specific traffic patterns, makes it significantly more difficult to address unsolicited traffic by statistical methods.

35 **[0010]** Further approaches are disclosed by S. Savage et al. in "Network support for IP-traceback", IEEE/ACM Transactions on Networking, 9(3):226-237, June 2001 and A. Yaar et al. "A stateless internet flow filter to mitigate DDoS flooding attacks; Proceedings of the IEEE Symposium on Security and Privacy, Oakland, CA, May 2004. In these approaches all or at least a large number of routers along a communication path within a network participate in the defending end systems against DDoS attacks. In practice, such solutions are slow to be deployed as they require changes

40 by multiple providers or even support from routers in the network.

[0011] A still further solution is disclosed by X. Yang et al. in "A DoS-limiting network architecture", ACM SIGCOMM'05, Philadelphia, PA, August 2005. This solution requires a collaboration of multiple entities at the edge of an access network and a core network. Again, a deployment is slow and difficult to achieve in practice as it is difficult to imagine that multiple Internet Service providers (ISPs) exchange information, in particular regarding security-related issues.

45 **[0012]** A further approach to protect end devices against Denial-of-Service attacks is known wherein a source wanting to send data to a destination must first receive a permission to communicate prior to sending data. This permission to communicate is also termed capability. See for example the publication of T. Anderson et al. "Preventing Internet Denial-of-Service with capabilities", Proceedings of Hot Topics in Networking (HotNetsII), Boston, October 2003. The capabilities are used to authenticate sources on an end-to-end basis. This means, that the capabilities are issued by a destination after checking the legitimacy of a source, and thus traffic is checked for capabilities by the destination to protect end system resources. However, such an end-to-end scheme does not protect the access networks and the core network from unsolicited traffic. Therefore, DDoS attacks on an access network or malware spread into the core network are not addressed.

50 **[0013]** Furthermore, EP 1 284 558 B1 discloses a method and apparatus for limiting the effect upon certain clients that access an electronic merchant's server from a denial of service attack against the electronic merchant's server. Therefore, the e-merchant designates some clients as VIP clients, wherein a packet addressed to the e-merchant's server is marked for forwarding in a privileged class of services, if the source of the packet is a VIP client.

Summary of the Invention

[0014] It is an object of the present invention to provide a method, a computer network system and a gate which are capable of preventing unsolicited traffic from spreading into the core network or preventing unsolicited traffic from over-

[0015] It is a major aspect of the invention to identify and control unsolicited traffic, in particular by limiting or entirely blocking unsolicited traffic in an access network or the core network, e. g. the Internet. In particular, the present invention relates to a computer network system and a method using authentication or permissions, also termed capabilities, to

- a) thwart Distributed Denial of Service (DDoS) attacks against a destination's access network at the edge of a core network and thus by implication on servers, and
- b) stop worm and virus spread prior to reaching the core network.

[0016] This aim is achieved by at least one gate, wherein each gate is located between an access network and the core network. The gate operates to identify and control unsolicited traffic received from malicious sources and to forward permissions or capabilities to sources wishing to send data by using an authentication authority. These functions performed by the gate and the authentication authority are also called edge-based capability (EC) functions due to the fact that the gate is located at an edge of the core network. Every source, e. g. a user, a web client or a host computer wanting to send data to a destination must receive a permission to send (a capability) prior to sending. This permission is used to generate and include a tag in every packet so that the EC gate is able to distinguish authenticated from unsolicited traffic and thus to allow only authenticated traffic to pass the gate.

[0017] Therefore, the present invention based on edge-based capabilities protects the core network and/or a destinations' access network, whereas the above mentioned known schemes protect end systems or destination resources.

[0018] According to one aspect of the invention, the authentication authority uses state-of-the-art authentication techniques, such as CAPTCHAS as described by L. v. Ahn et al. in "CAPTCHA: Using Hard AI Problems for Security"; Advances in Cryptology-EUROCRYPT 2003, volume 2656 of Lecture Notes in Computer Science, pages 294-311, Springer Verlag and/or IP Puzzles as described by D. Dean et al. in "Using client puzzles to protect TLS", Proceedings of the 10th USENIX Security Symposium, Washington, D.C., August 2001 to transmit permission information to the sources. The sources, i.e. users and/or computers try to obtain the valid tagging key from the permission information, used for tagging packets to be sent to a destination.

[0019] It is to be noted, that in a preferred embodiment the authentication information is shared among three distributed components, i.e. sources, the gates and the authentication authority in a secure, efficient and scalable way.

[0020] It is a further aspect of the invention that the computer network system can be implemented readily, e.g. by a single Internet Service Provider (ISP). Owners of end systems connected to the ISP's network have the incentive to install the respective components of the present invention to profit from the protection against unsolicited traffic provided by the gate.

[0021] The above mentioned technical problem is solved by a a method for identifying and controlling unsolicited traffic in a computer network system. The computer network system comprises at least a first access network to which at least one source is connected, at least a second access network to which at least one destination is connected and a core network connecting the access networks. The method comprises the following steps:

- directing traffic units addressed to a destination from the at least one source to a gate which is interconnected between the core network and the second access network;
- verifying each traffic unit at the gate to determine whether or not the traffic unit has been marked validly;
- identifying each traffic unit as unsolicited if it has been marked invalidly or if it has not been marked at all;
- controlling the flow of each traffic unit having been identified as unsolicited in response to at least one predetermined system parameter.

[0022] This method helps to thwart Distributed Denial of Service (DDoS) attacks against the second access network at the edge of a core network and thus by implication on servers.

[0023] The technical problem is further solved by a method which comprises the steps:

- directing traffic units addressed to a destination from the at least one source to a gate which is interconnected between the core network and the first access network;
- verifying each traffic unit at the gate to determine whether or not the traffic unit has been marked validly;
- identifying each traffic unit as unsolicited if it has been marked invalidly or if it has not been marked at all;
- controlling flow of each traffic unit having been identified as unsolicited in response to at least one predetermined system parameter.

[0024] This method aims to stop worm and virus spread prior to reaching the core network.

[0025] Preferred embodiments are set forth in the dependent claims.

[0026] If the Internet is used as the core network traffic units are IP packets.

5 **[0027]** The step of controlling the flow of each traffic unit having been identified as unsolicited may include blocking, delaying and/or limiting the flow of each traffic unit having been identified as unsolicited to a destination for example in dependence of the congestion condition of the core network and/or the second access network.

[0028] In order to improve the network based flow control of traffic units having been identified as unsolicited a permission message is generated and transmitted to the at least one source. Traffic units to be sent are then marked at the source using information from the permission message. It is to be noted, that the permission message comprises
10 as an information a tagging key, in the description also called session keys, to mark traffic units. To allow the gate to verify traffic units information relating to the permission message is stored at the gate. This information might be the tagging key or information to derive a tagging key.

[0029] In a preferred embodiment, the permission message is transmitted in form of a challenge to be solved by the at least one source to obtain the tagging key. Using challenges only computers working properly are able to obtain a
15 valid tagging key from the permission message.

[0030] The challenge is to be designed such that it can distinguish legitimate users or end systems from other sources. In a preferred embodiment that allows a distinction between human users and remotely controlled machines (zombies, bot-nets) a challenge is constructed such that it can be solved only with an interaction of a human user.

[0031] A CAPTCHA algorithm and/or an IP puzzle algorithm as mentioned above can be used to generate a challenge to transmit the permission message to the at least one source. In this case cryptographic keys are used for tagging
20 traffic units.

[0032] In a preferred embodiment the tagging key is used in a message authentication code (MAC) to compute a tag to mark traffic units. The tagging key or information to derive the tagging key is stored at the gate to verify traffic units received from sources.

25 **[0033]** In a preferred embodiment, the tagging key is a cryptographic key that is used by the source to generate tags for marking traffic units and that is used by the gate to verify tags included in traffic units.

[0034] Preferably, the permission message is generated by the gate and forwarded to an authentication means which in turn transmits the permission message to the at least one source.

[0035] Alternatively, the permission message is generated by an authentication means which transmits the permission message to the at least one source and the gate.
30

[0036] The above mentioned problem is also solved by a computer network system for identifying and controlling unsolicited traffic. The system comprises a core network, at least a first access network, at least one source connected to said first access network, at least a second access network, at least one destination connected to said second access network, and at least one gate located between the first access network and the core network and/or between the second
35 access network and the core network. Furthermore, means for generating and transmitting a permission message to the at least one source are provided, wherein the at least one source comprises means for obtaining a tagging key from that permission message used to mark traffic units addressed to a destination. The gate comprises storage means for storing information related to a permission message, means for verifying received traffic units on the basis of the information stored to determine whether or not the traffic units are marked validly, means for identifying each traffic unit being
40 marked invalidly as unsolicited, and means for controlling the flow of each traffic unit being marked invalidly in response to at least one predetermined system parameter.

[0037] It is to be noted that the feature "traffic unit being marked invalidly" also covers traffic units which are not marked at all.

45 **[0038]** In a preferred embodiment, each source comprises a client computer, each destination comprises a server and the core network is the Internet.

[0039] In order to facilitate authentication of sources, the means for generating and transmitting a permission message to the at least one source is implemented in an authentication unit which can be located in a separate device connected to the gate or in at least one destination.

50 **[0040]** In a preferred embodiment the authentication unit is adapted to generate a permission message in form of a challenge to be solved by the at least one source, wherein only sources having solved the challenge are able to mark traffic units validly.

[0041] The above mentioned technical problem is further solved by a gate used in a computer network system according to any one of the claims 14 to 17. The gate comprises storage means for storing information related to a permission message, means for verifying traffic units on the basis of the information stored to determine whether or not the traffic
55 units are marked validly, means for identifying each traffic unit being marked invalidly as unsolicited, and means for controlling the flow of each traffic unit being marked invalidly in response to at least one predetermined system parameter. It is to be noted that the feature "traffic unit being marked invalidly" also covers traffic units which are not marked at all.

Brief Description of the Drawing:**[0042]**

- 5 Fig. 1 is a principle block diagram of a communication system showing the basic concept of the present invention;
 Fig. 2 is a block diagram of a communication system adapted to thwart DoS attacks against a destination's access network;
 Fig. 3 is a block diagram of a communication system adapted to stop worm and virus spread prior to reach the core network;
 10 Fig. 4 is a block diagram showing the function modules associated to the respective components of the communication systems as depicted in Fig. 2 and 3;
 Fig. 5 is a flowchart showing the steps performed at the gate.

Detailed Description

15

[0043] Fig. 1 shows a block diagram of a preferred communication system 10 based on a commonly used distributed networked system including an aspect of the present invention. The communication system 10 comprises end systems or end devices and a network. End systems are separated into sources 101 and destinations 103, wherein only one source and one destination are shown. The network is divided into a core network 104, e. g. the Internet, and access networks. In order to facilitate the illustration only a single source-side access network 105 and a single destination-side network 107 are shown. The conjunction of an access network and the core network is called edge. Thus, we distinguish between a source-side edge 106 and a destination-side edge 108. Each edge usually comprises one or multiple hardware components, such as routers. A common characteristic of the edges is that the network capacity is significantly different between an access network and the core network, i.e., the network capacity is one or several orders of magnitude smaller in the access networks 105, 108 than in the core network 104. Therefore, an access network is significantly more vulnerable to traffic flooding than the core network 104. Therefore, unsolicited traffic must be identified and controlled, e.g. limited or completely blocked, at an edge. For this purpose, a specific network element termed gate 109 is added to the network infrastructure at least at one edge, for example at the source side and/or the destination side that ensures that the network resources are available for legitimate traffic.

30 **[0044]** Fig. 2 shows a first preferred embodiment of a communication system as generally depicted in Fig. 1. This communication system comprises a gate 109 at a destination side edge of a core network 104, i. e. between an destination-side access network 107 and the core network 104. The destination-side access network 107 serves to connect destinations, e. g. Web servers 103-1 to 103-N to the core network 104. It is to be understood, that basically a plurality of destination-side access networks are connected to the core network 104. The gate 109 operates to protect the destination-side access network 107 against DDoS flooding attacks.

35 **[0045]** In this embodiment, legitimate sources 101-1 to 101-N, such as Web clients are connected to the core network 104 via a source-side access network 105a, whereas malicious sources (zombies) 102-1 to 102-N are connected to the core network 104 via a separate access network 105b. In Fig. 2, wanted traffic generated by the legitimate sources 101-1 to 101-N is designated by reference number 111, whereas unsolicited traffic generated by the malicious sources 102-1 to 102-N is designated by reference numbers 112. It will be described later how clients are identified and distinguished as legitimate sources and malicious sources, respectively.

[0046] The attack or unsolicited traffic 112 from the malicious sources 102 to 102-N sent to one or multiple destinations 103-1 to 103-N converges at the edge joining the core network 104 with the destination-side access network 107. Without gate 109, the access network 107 with its lower capacity can be flooded with unsolicited traffic 112.

45 **[0047]** In the communication system, whenever a source wants to communicate with a destination, it must first receive a permission to communicate. This permission to communicate is also termed capability in literature. The permission 115 is generated and forwarded to one or more sources by a specific software module called authentication authority 113. The authentication authority 113 might be implemented in one or more destinations 103 to 103-N or in a separate device associated to the gate 109. By generating and forwarding permissions to sources wishing to communicate with destinations the sources become implicitly authenticated, i. e. legitimate sources, only if they are able to obtain a valid tagging key from the received permission. This procedure will be explained below in detail. With other words, capabilities or permissions are used to authenticate sources implicitly. A legitimate source uses the tagging key included in the permission to tag packets to be sent to a destination.

50 **[0048]** With gate 109, the attack traffic 112 is identified and can be filtered out such that only a controllable number of legitimate packets 111 are forwarded from the core network 104 to the access network 107. The unsolicited traffic 112 filtered out by the gate 109 is symbolically depicted by a ground symbol.

[0049] Additionally or alternatively, the amount of attack traffic 112 flowing through the access network 107 can be controlled by the gate 109 in dependence of the condition of the access network 107. The gate 109 identifies traffic as

unsolicited traffic if the received packets do not contain a valid tag generated on the basis of the permission.

[0050] Furthermore, as will be described later, the gate 109 uses a key generation algorithm to generate tagging keys. The tagging keys are transported to the authentication authority 113 which forwards the permission and the tagging key or portion of the key to the sources. Only sources which work correctly are able to calculate the tagging key from the permission which is then used to tag or mark packets to be sent to respective destinations.

[0051] Fig. 3 shows a second embodiment of a communication system. This communication system differs from that shown in Fig. 2 in that a gate 109 is arranged at a source side edge of a core network 104, i.e. between a source-side access network 105 and the core network 104. Assume that malware, e. g. a virus or a worm has infected a set of sources 102-1 to 102-N, for example computers and replicates itself with horrendous speed to infect other source computers. These source computers 102-1 to 102-N will fail to obtain a correct permission generated by and received from an authentication authority 115. Assume further, that the malware 112 spreads via email. Gate 109 is adapted to automatically determine whether an email 111 has been sent by a legitimate source 101-1 to 101-N or by a malicious computer 102-1 to 102-N. Only emails 111 sent by a legitimate source 101 to 101-N have been marked correctly and will pass the gate 109. Therefore, the malware which are not marked or carrying an invalid tag can be identified as unsolicited traffic 112 and prevented from spreading past the gate 109 into the Internet 104.

[0052] The internal operation of the gate 109 as shown in Fig. 2 and 3 is depicted in Fig. 5. The gate 109 has the ability to identify and control, e.g. filter out or limit unsolicited traffic. When a packet 110 arrives from sources 101 or 102 at the gate 109, the gate 109 first checks in step 123 whether the packet contains a tag. If the packet contains a tag, then the gate 109 retrieves in step 120 a tagging key previously generated for a given session. If a suitable tagging key is available, the gate 109 uses the tagging key to determine in step 124 whether or not the packet includes a valid tag. If the tag is valid, the packet 111 can be forwarded immediately to an output queue of the gate 109.

[0053] If, however, a tag is not included in a packet 118 or not valid 122 or a tagging key is not available 121 in the gate 109, the packet is considered as an unsolicited or a malicious packet 112 and thus will be filtered or controlled 125 by the gate 109. The term "filtered" or "controlled" 125 can be interpreted and implemented in various ways. Examples include, but are not limited, to simply drop unsolicited packets 112 or to forward them to a low-priority queue 125. An efficient solution is required such that the gate 109 does not slow down the traffic in the destination's access network 107 or in the core network 104. Furthermore, the solution needs to scale to a large number of flows.

[0054] As a brief overview, the capability exchange for legitimating sources to send wanted traffic 111 involves three independent components: sources which must acquire a capability, the authentication authority 113 to issue and forward capabilities and the gate 109 to check capabilities in order to identify and control unsolicited traffic. The general concept is shown in Fig. 4.

[0055] For this purpose, the gate 109 uses a tagging scheme for verifying packets and a keying scheme for generating tagging keys. The keying scheme can be implemented in key generation means 114 and the tagging scheme can be located in a tag verification means 117, both located within the gate 109.

[0056] Furthermore, a key transport scheme is used by the authentication authority 113. The key transport scheme can be implemented in a key transport means 118. In a preferred embodiment both the authentication authority 113 as well as the key transport scheme are implemented in one or more of the destinations 103-1 to 103-N. In addition, a tagging scheme is used by the sources to generate tags and embed them into packets to be sent. Therefore, tag generating means 116 can be implemented in each source to perform the tagging scheme. A detailed description of these schemes will be given below.

[0057] In a first step, a tagging key is generated by the key generation means 114 of the gate 109. Derivations of this key are passed to the authentication authority 113. The authentication authority 113 is a logical unit or software module that may be mapped onto an independent device, or may be collocated with other devices, such as the server 103-1. The authentication authority 113 is adapted to implicitly authenticate clients 101. This is done by generating a permission or a capability, for example in form of a challenge which includes a tagging key or a portion of a tagging key that is valid for a single session. The challenge is forwarded to the sources 101 and 102. Each source tries to solve the received challenge for obtaining a valid tag which is used to mark packets to be sent to destinations. In the gate 109 packets marked with a valid tag are considered as a legitimate traffic 111 whereas traffic without a tag or with an invalid tag are identified as unsolicited traffic 112 and thus filtered respectively, as shown in Fig. 2, 3 and 5.

[0058] An illustrative operation of the computer network system as shown in Fig. 2 will be described in more detail.

[0059] This computer network system is adapted to prevent the access network 107 from DDoS attacks, but as mentioned above, it can be used for other scenarios as well.

[0060] It is assumed, that the destinations 103-1 to 103-N are servers, that the sources 101-1 to 102-N are client computers and that the authentication authority 113 is located in at least one of the servers 103.

[0061] In a preferred embodiment, the Keying Scheme built into the key generation means 114 is responsible for generating a key at the gate 109 and communicating it to one or more of the servers 103-1 to 103-N or, in general, the authentication authority 113. It is important to realize that the gate 109 defines and manages keys and exchanges them with the servers, but not vice versa. The key sent to the server 103-1 is server-specific. The server 103-1 stores its key

locally. Once the server 103-1 receives a request from a client, e. g. client 101-1, it derives a session-specific key as a tagging key to be sent to the sources.

[0062] The Key Transport Scheme associated to the authentication authority and implemented for example in the server 103-1 is responsible for communicating the session or tagging key to clients, e.g. in form of a challenge generated by the authentication authority 113. When the user of the client 101-1 and/or the client itself solves this challenge, the client 101-1, the user uses, is considered as authenticated.

[0063] In the present embodiment, the clients 101-1 to 101-N are assumed to be able to solve the received challenge correctly and thus are legitimated or authenticated. It is assumed that the clients 102-1 to 102-N are not able to solve the received challenge correctly and thus are considered as zombies.

[0064] The Tagging Scheme implemented in the clients allow the legitimate clients to embed a tag in each packet it send, after being authenticated and after obtaining a tagging key by solving the challenge correctly.

[0065] At gate 109, the Tagging Scheme defines how the gate 109 verifies the tags included in the packets with information stored at the gate 109.

[0066] In the following, we describe the three schemes in detail. For the description we introduce the following notation: *Notation 1:* For an IP packet P we use $IP_s(P)$, $IP_d(P)$, $Ident(P)$, $Flow(P)$, $Len(P)$ to denote the respective header fields source address, destination address, identification (in IPv4), flow label (in IPv6), and payload length of the packet. Likewise, we use $Port_s(P)$, $Port_d(P)$, $Seq(P)$, $Ack(P)$ to denote the source port, destination port, sequence and acknowledgement number in case that P is a TCP packet.

Notation 2: For a bitstring X let $[X]_r$ denote the leftmost r bits of X ; if X has less than r bits, it is filled up by zero bits. Let $[X]_{r,s}$ with $r \leq s$ denote the sub-bitstring consisting of the bits $r, \dots, s$, filled up by zeroes if necessary.

1. Keying Scheme

[0068] In the following two schemes are described for tagging key generation and management for the gate, servers, and clients. The first scheme, which is called *stateless keying*, derives all tagging keys from a central master key held at the gate 109. In the second scheme the gate 109 needs to keep some state for each flow routed through it.

a) Stateless Keying

[0069] In this keying scheme, the gate 109 has a central master or gate key k_G . As all other keys are derived from this key it is kept secret at all times. The scheme further uses two key derivation functions KDF_1 , KDF_2 to derive server keys and session or client keys. Session keys are also termed tagging keys. Given a server S with address $IP(S)$, its key is

$$k_S \leftarrow KDF_1(k_G, IP(S))$$

[0070] Furthermore, for a client C with address $IP(C)$, the tagging key used to tag any packet sent to server S is

$$k_{C,S} \leftarrow KDF_2(k_S, IP(C), IP(S))$$

[0071] The interface to the tag verification means 117 in the gate 109 can be easily implemented using these functions. Note that the client does only receive $k_{C,S}$, so it does not need to know about KDF_1 , KDF_2 .

[0072] Adding a cache (not shown) in the gate 109 for the results of key derivation does not affect the property that the scheme is stateless. However, it does improve computational efficiency of the scheme. We use two LRU (least recently used) tables in the gate 109 holding entries $(IP(S), k_S)$ and $(IP(S), IP(C), k_{C,S})$. That way we can amortize the computational complexity of the tagging key derivation over a significant number of tag verifications while the tables are easy to maintain. The achievable improvement in computation time depends on available memory at the gate 109 and the number of different flows passing the gate at any single time.

b) Keying with State at the Gate

[0073] Rather than deriving all the keys from a single master key kept at the gate 109, the gate 109 could issue a new tagging key for each single connection it encounters. In the following we describe how such an alternative keying scheme could work.

[0074] In this scheme the gate 109 generates a session or tagging key k whenever it sees a connection initiation

involving a client C and a server S pass by, i.e. in form of a SYN packet. It then stores k along with (C, S) and other information identifying the flow, e.g. port numbers, for future use.

[0075] Then the gate 109 passes on the tagging key to the respective server in which the authentication authority 113 is implemented such that the server can transport it to the client. This requires that the session or tagging key k is encrypted with some key k_s shared between the gate 109 and the server 103-1. The communication between the gate 109 and the server 103-1 can be done in-band, i.e. by adding the encrypted key to the original SYN packet of the client 101-1 before passing it on to the server 103-1. To expire session keys the gate 109 has to watch the connection and know about protocol timeouts. This requires a similar state-keeping as in a stateful-inspection firewall.

c) Selection of Algorithms and Parameters

[0076] For the stateless keying scheme the key sizes as well as the key derivation functions need to be specified. Given that we select CBC-MAC with AES and a key size of 128 bits for the tagging scheme, it is reasonable also use these parameters for the keying scheme. Furthermore, the key derivation functions KDF_1 , KDF_2 can be implemented using CBC-MAC, given that it is a good pseudorandom function. This function is described, for example, by M. Bellare et al. "The security of the cipher block chaining message authentication code.", Journal of Computer and Systems Sciences, 61(3):362-399, December 2000. This way no additional code needs to be added. Concretely,

$$KDF_1(k_G, IP(S) := CBC - MAC[AES]_{k_G}(c_1 || IP(S)),$$

$$KDF_2(k_s, IP(C), IP(S) := CBC - MAC[AES]_{k_s}(c_2 || IP(C) || IP(S))$$

using two different constants c_1 , c_2 for separating the input to the encryption function. We use constants of 12 or 8 octets each. The performance of these functions is then as follows: For IPv4 the IP address(es) and constants result in inputs of 16 octets for KDF_1 resp. KDF_2 , i.e. a single block cipher operation. For IPv6 the IP address(es) and constants result in inputs of 28 or 40 octets, resulting in KDF_1 resp. KDF_2 using 2 or 3 block cipher operations. Furthermore, a key scheduling operation is needed when putting the resulting key to use.

2. Key Transport Scheme

[0077] In this section we propose methods to transport the cryptographic key generated by the gate 109 from the server 103-1 or authentication authority 113 to the client 101-1 at the start of the respective connection. The client uses this key to tag each packet to be sent to a server.

[0078] We employ the general idea of a CAPTCHA test (Completely Automated Public Turing Test to tell Computers and Humans Apart) to distinguish between a human user and a machine or client. The CAPTCHA test is for example described by L.v.Ahn et al. in "CHAPTCHA: Using Hard AI Problems for Security", Advances in Cryptology-EUROCRYPT 2003, vol. 2656 of Lecture Notes in Computer Science, pages 294-311, Springer Verlag.

[0079] This way our scheme ensures that human users can still access a server even under attack. On the other hand, a malicious client, e.g. 102-1, that is potentially part of a DDoS attack network, and that can therefore not rely on help of a human user, is not able to get a valid tagging key.

[0080] The CAPTCHAs were originally designed as tests. In this embodiment, we use a subclass of those problems that allow the user to input some information derived from a given artificial intelligence (AI) problem. In particular, such an AI problem is called a challenge. The challenge is generated by the authentication authority 113 using a tagging key received by the gate 109. The challenge including the tagging key is then transmitted to the clients 101 and 102.

[0081] Then we use this input for a computation that will give a cryptographic key to the client computer.

[0082] One problem we face here is that probably only a minor fraction of potential user is willing to input the equivalent of a 128-bit tagging key, which is a tedious and error-prone task, e.g. reading the key from a distorted image. Therefore we must restrict ourselves to using only a small number of bits, resulting in a low-entropy input.

[0083] In the following we propose two solutions that combine the CAPTCHA with a variant of IP puzzles in a way that either the user helps the client machine in solving the puzzle faster, or allows the client machine to use the solution at all. IP puzzles are known and described for example by D. Dean et al. in "Using client puzzles to protect TLS", Proceedings of the 10th USENIX Security Symposium, Washington, D.C., August 2001.

[0084] In fact, in the first scheme the puzzle-solving is helped by the human user, while the second scheme requires a correct solution of the CAPTHCA by the user.

[0085] In both schemes the client machine tries to obtain the key that is used by the server 103-1 or the authentication authority 113 to encrypt the tagging key.

[0086] Our proposals are scalable in the sense that a busy server 103-1 can select which effort, e. g. in computing cycles or waiting time a client computer has to invest in order to access the server.

5 **[0087]** The schemes make use of a preimage-resistant cryptographic hash function h (such as SHA-256) with

$$h: \{0,1\}^* \rightarrow \{0,1\}^n.$$

10

[0088] The hash function SHA-256 is described, for example, in National Institute of Standards and Technology (NIST), Computer Systems Laboratory. Secure Hash Standard. Federal Information Processing Standards Publication (FIPS PUB 180-2), February 2004.

[0089] We assume that h is either shared somehow between the client and the server or is otherwise public.

15 **[0090]** Furthermore it should be noted that due to the way the key transport schemes are integrated into the EC communication system, the schemes need to be realized as a single round, i.e. a single back-and-forth exchange of messages. The idea is to piggyback a client-to-server message, if existing, into an initial SYN packet, while the server-to-client message is integrated into the SYN-ACK packet. Preferably, we are using in-band communication.

20 **[0091]** With the two key transport schemes the system's designer or operator has a choice whether client machines shall be able to connect even though no human user stands essentially behind it. This choice might also be dependent on the current system and network load.

a) Scheme Helped by CAPTCHA Solution

25 **[0092]** In the first scheme, we are effectively realizing a differential work load for a client computer, i.e. a client computer and a human user working together have a vastly faster access to the tagging key than the machine alone.

[0093] Our IP puzzle works as follows. The server 103-1 or authentication authority 113 selects an $m < n$, i. e. the work factor, and a random string r^* of length n . The value r^* itself will not be transmitted to the client computer. Additionally the server computes

30

$$k_e \leftarrow h(c_p \| IP(S) \| IP(C) \| r^*)$$

35 and uses k_e to encrypt the tagging key. Let $n_p = n - m$ denote the number of cleartext bits, m_u the number of bits to be entered by the user, and m_c the number of bits determined by the machine. We require $m_u + m_c = m$. Let $r^* = [r^*]_{n_p}$, $r_c = [r^*]_{n-m+1, n-m_u}$, and $r_u = [r^*]_{n-m_u+1, n}$. Then $r^* = r^* \| r_c \| r_u$.

[0094] The client is given the tuple

40

$$(r^*, s^*, C)$$

with

45

$$s^* = h(r^* \| IP(C) \| IP(S)),$$

50

$$C = Enc_{k_t}(k_t),$$

where k_t is the tagging key to be transmitted.

[0095] In order to obtain the cleartext k_b the client needs to find r_c and r_u in order to retrieve r^* . The client can check that it found a solution r by testing if

55

$$h(r\|IP(C)\|IP(S)) = s^*$$

5

[0096] Given that h is a cryptographic hash function that is providing some partial-input preimage resistance, this needs to be done by brute force. This allows us to roughly predict the average amount of computing time the client needs to invest. After finding r^* the client can compute k_e and decrypt k_t .

10

[0097] The CAPTCHA now plays into this by giving an additional hint on r^* by providing r_u in the form of a CAPTCHA, so that the user can type in r_u into the client computer and thus effectively help the client computer to solve the puzzle and retrieve the tagging key.

b) Scheme Requiring a CAPTCHA Solution

15

[0098] A slight variation of the above scheme allows making sure that the client computer can make use of the solution to the client puzzle only if a human user provides the correct solution to the CAPTCHA. The client puzzle is very similar to the one given above.

[0099] Using the notation r^*, r^A, r_c, r_u established above, the server 103-1 selects, instead of r^* , a random r^A, r_c, r_u , computes $r_u = [h(c_u \| r_u)]_{mu}$. Then we use $r^* = r^A \| r_c \| r_u$ and

20

$$k_e \leftarrow h(c_p \| IP(S) \| IP(C) \| r^*) \oplus h(c_p \| IP(S) \| IP(C) \| h(c_u \| r_u))$$

25

as key to en- or decrypt k_t . Here c_u and c'_u are again two publicly known constants (differing also from the other constants used above). Like in the first key transport scheme, the user will be given r_u in the form of a CAPTCHA. The crucial observation here is that the client is not provided a means to integrity check either k_e or k_t and therefore has no means to directly verify that the correct key has been found. This means that essentially the correct key is obtained only if both the correct solution to the IP puzzle and the CAPTCHA are known to the client computer.

30

c) Selection of Algorithms and Parameters

35

[0100] We propose a possible instantiation of our first scheme for key transport, the one realizing a differential work factor. We instantiate the 2nd preimage resistant hash function h with SHA-256. For the encryption function $Enc_k(.)$ we employ the block cipher AES with a 128-bit key; we obtain the key k_e from the output of h by truncating the output to the required size.

40

[0101] The selection of m_c and m_u (and thus m) is critical for the security as well as the usability of the system. We assume that a graphical CAPTCHA consisting of characters and digits (and resolving some ambiguities such as '0' vs. 'O', '1' vs. 'l') is acceptable. We use only the lowercase characters and convert uppercase letters if necessary. This will give us about 5 bit per character typed in by the user. In our opinion 4 characters or 20 bits in total should be acceptable. Furthermore, on a Pentium M (1.5GHz) system the implementation of SHA-256 runs at about 2^{18} operations per second.

45

[0102] This choice of $m_u=20$ results in a differential work factor of about 2^{20} , so that a choice of $m_c=18$ and thus $m=38$ should be sufficient at the current time. Solving the puzzle with human help should need about 0.5 seconds on average on the machine given as reference above. The value of m_c would need to be gradually increased in the future when vastly faster machines are commonplace.

3. Tagging Scheme

50

[0103] First, we recall the requirements that the tagging scheme performed by a client must fulfill in order to efficiently and securely work. Then we describe the concrete scheme, and provide some performance figures.

55

[0104] To explain the tagging scheme we assume that a client computer C wishing to communicate with a server S behind the gate 109 has a cryptographic key $k_{c,s}$ that is specific to the client-server pair and possibly time-dependent or even session-dependent. Furthermore, we assume that the gate 109 itself does also know $k_{c,s}$, possibly by deriving it via the keying scheme from other information known to the gate 109.

a) Requirements

[0105] The purpose of the tagging scheme is to provide proof to the gate 109 about the authenticity of a packet sent from the client computer *C* to the server *S*, as the gate sets -in the event of an attack or overload - the priority with which the packet is forwarded via the access network 107 to the server.

* Security. The tagging mechanism shall keep an adversary from learning the cryptographic key $k_{c,s}$ from the packets of a flow. Furthermore, an adversary shall be unable to reuse the tag from one packet to forge a valid tag for another packet, even if the adversary spoofs the client address as origin.

* Efficiency of verification. The gate 109 has to check every packet that is tagged, so the tagging scheme shall not impose too much computational overhead in order to keep latencies small.

[0106] It should be noted that the server *S* or the authentication authority does not have to cope with generation or verification of tags.

b) The Concrete Scheme

[0107] For tagging a packet the tag generation means 116 of a client uses a special tag field in the IP header of a packet to be sent. The gate 109 will then look for available tags in received packets, check the tags, and for example base a decision about the packet's priority on this.

[0108] Given the requirements discussed above we employ a secure Message Authentication Code (MAC) scheme, i.e. a pair of possibly randomized algorithms (*Mac*, *Ver*) for tagging packets and verification, making use of parameters *k* and *t* for tagging key and tag lengths measured in bits:

25

$$Mac : \{0,1\}^k \times \{0,1\}^* \rightarrow \{0,1\}^t$$

30

$$Ver : \{0,1\}^k \times \{0,1\}^* \times \{0,1\}^t \rightarrow \{0,1\}$$

with $Ver(k, M, Mac(k, M)) \Rightarrow 1$.

[0109] In the following we describe the tagging and verification scheme based on these functions.

[0110] First we describe which data from a packet *P* is included in the tag computation and verification. Here we distinguish between IPv4 and IPv6: For a given packet *P*, we use the tuples

40

$$M_{hdr}(P) \leftarrow \begin{cases} IP_s(P), IP_d(P), Len(P), Ident(P) & IPv4 \\ IP_s(P), IP_d(P), Len(P), Flow(P) & IPv6 \end{cases}$$

and for a TCP packet

45

$$M_{tcp}(P) \leftarrow (Port_s(P), Port_d(P), Seq(P), Ack(P))$$

50

(or $M_{tcp}(P) \leftarrow ()$ for non-TCP packets) to set up the input to the tagging and verification operations as

55

$$M(P) \leftarrow (M_{hdr}(P), M_{tcp}(P))$$

[0111] For IPv4 the maximal total size of $M(P)$ is 12+12 octets or 96+96=192 bits, for IPv6 these numbers are 37+12=49

octets resp. $296+96=392$ bits (where we count $Flow(P)$ as 3 octets or 24 bits).

[0112] Now the procedures for computing a tag at the client 101-1 and for checking a tag at gate 109 are as follows:

* The client tries to retrieve from its internal key storage the tagging key for each packet P it sends to the server:

5

$$k \leftarrow \text{client_get_session_key}(\text{IP}_d(P)).$$

10 If no tagging key is available, the packet is sent untagged. Otherwise the client sets up $M(P)$, computes

$$\tau(P) \leftarrow \text{Mac}(k, M(P)),$$

15

and includes $\tau(P)$ in P using the tag option.

* The gate 109 maintains a label (cfg. Fig.5)

20

$$\lambda(P) \in \{\text{Untagged}, \text{Invalid}, \text{Valid}\}$$

for each packet for further queuing decisions, determined as follows. If a packet P is not tagged

25

$$\lambda(P) \leftarrow \text{Untagged}.$$

If a packet P carries a tag option, the gate obtains

30

$$k \leftarrow \text{gate_get_session_key}(\text{IP}_s(P), \text{IP}_d(P)).$$

If no tagging key is available $\lambda(P) \leftarrow \text{Invalid}$. Otherwise the gate extracts the alleged tag $\tau'(P)$, obtains $M(P)$ from the other packet information as described above, and computes

35

$$b \leftarrow \text{Ver}(k, M(P), \tau'(P)),$$

40

Then

$$\lambda(P) \leftarrow \begin{cases} \text{Valid} & b = 1 \\ \text{Invalid} & b = 0 \end{cases}$$

45

Based on the label the gate 109 decides what further to do with the packet, e.g. forward it with high or low priority, or drop it entirely. Note, however, that dropping of SYN packets is not a good idea, as it would essentially prevent the server to be reachable.

50

d) Selection of Algorithms and Parameters

[0113] For implementing the MAC scheme several straight-forward choices are available, such as CBC-MAC with various block ciphers or the HMAC construction, e.g. HMAC-SHA1. This choice directly affects the natural key size and maximal tag length of our scheme. For IPv4 $M(P)$ fits into two AES blocks, so using CBC-MAC with AES would require just two encryption operations, while for IPv6 four encryption operations would be needed. In both cases HMAC-SHA1 would need, depending on implementation, 2 or 4 compression function operations, so we chose CBC-MAC with AES

55

and a 128 bit key.

[0114] We shorten the output of 128 bits to the first 80 bits of tag data, so that the option for IPv4, including option type and length, has a total length of 96 bits or 12 octets. This still offers a level of security against an adversary that is appropriate for our purpose.

5

Claims

1. A method for identifying and controlling unsolicited traffic in a computer network system (10), which comprises at least a first access network (105, 105a, 105b) connecting at least one source (101-1 to 101-N; 102-1 to 102-N) to a core network (104) and at least a second access network (107) connecting at least one destination (103-1 to 103-N) to said core network (104), wherein said method comprises the steps:

- directing traffic units addressed to at least one destination (103-1) from the at least one source (101-1) to the gate (109) which is interconnected between the core network (104) and the second access network (107);
 - verifying each traffic unit at the gate (109) to determine whether or not the traffic unit has been marked validly;
 - identifying each traffic unit (112) as unsolicited if it has been marked invalidly or if it has not been marked at all;
 - controlling the flow of each traffic unit (112) having been identified as unsolicited in response to at least one predetermined system parameter.

20

2. A method for identifying and controlling unsolicited traffic in a computer network system, which comprises at least a first access network (105) connecting at least one source ((101-1 to 101-N; 102-1 to 102-N) to a core network (104) and at least a second access network (107) connecting at least one destination (103-1 to 103-N) to said core network (104), wherein said method comprises the steps:

25

- directing traffic units addressed to at least one destination (103-1) from the at least one source (103-1) to the gate (109) which is interconnected between the core network (104) and the first access network (105);
 - verifying each traffic unit at the gate (109) to determine whether or not the traffic unit has been marked validly;
 - identifying each traffic unit (112) as unsolicited if it has been marked invalidly or if it has not been marked at all;
 - controlling flow of each traffic unit having been identified as unsolicited in response to at least one predetermined system parameter.

30

3. The method according to claim 1 or 2, wherein the traffic units are IP packets or TCP packets and wherein the core network (104) is the Internet.

35

4. The method according to any one of the claims 1 to 3, wherein the step of controlling comprises the following steps:

blocking, delaying and/or limiting flow of each traffic unit having been identified as unsolicited to a destination (103-1 to 103-N) in dependence of the congestion condition of the core network (104) and/or second access network (107).

40

5. The method according to any one of the preceding claims, comprising the following steps:

- Generating and transmitting a permission message to the at least one source (101-1);
 - Marking traffic units to be sent at the source using information from the permission message; and
 - Storing information relating to the permission message at the gate (104) to allow the gate to verify traffic units.

45

6. The method according to claim 5, wherein the permission message includes a tagging key used to mark traffic units to be sent at the source.

50

7. The method according to claim 5 or 6, wherein the permission message is transmitted in form of a challenge to be solved by the at least one source to obtain the tagging key.

8. The method of claim 7, wherein the challenge is solved at the source only with an interaction of a human user.

55

9. The method of claim 7 or 8, wherein a CAPTCHA algorithm and/or an IP puzzle algorithm are used to generate a challenge to transmit the permission message to the at least one source.

10. The method of claim 6, wherein the tagging key is used in a message authentication code (MAC) to compute a tag to mark traffic units, wherein the tagging key or information to derive the tagging key is stored at the gate.
- 5 11. The method of claim 10, wherein the tagging key is a cryptographic key that is used by the source to generate tags for marking traffic units and that is used by the gate to verify tags included in traffic units.
12. The method according to any one of the claims 5 to 11, wherein the permission message is generated by the gate (104) and forwarded to an authentication means (113) which in turn transmits the permission message to the at least one source.
- 10 13. The method according to any one of the claims 5 to 11, wherein the permission message is generated by an authentication means (113) which transmits the permission message to the at least one source and the gate.
14. A computer network system for identifying and controlling unsolicited traffic, comprising
- 15 a core network (104),
 at least a first access network (105; 105a, 105b),
 at least one source (101-1 to 101-N; 102-1 to 102-N) connected to said first access network (105; 105a, 105b),
 at least a second access network (107),
 at least one destination (103-1 to 103-N) connected to said second access network (107),
 20 at least one gate (109) located between the first access network (105; 105a, 105b) and the core network (104) and/or between the second access network (107) and the core network (104),
 means (113; 114) for generating and transmitting a permission message to the at least one source (101-1 to 101-N; 102-1 to 102-N), wherein
 the at least one source (101-1 to 101-N; 102-1 to 102-N) comprises means (116) for obtaining a tagging key from
 25 that permission message used to mark traffic units addressed to a destination,
 the gate (109) comprises:
- storage means for storing information related to a permission message,
 - means (117) for verifying received traffic units on the basis of the information stored to determine whether or
 - 30 not the traffic units are marked validly,
 - means (117) for identifying each traffic unit being marked invalidly as unsolicited, and
 - means (117) for controlling the flow of each traffic unit being marked invalidly in response to at least one predetermined system parameter.
- 35 15. The computer network system according to claim 14, wherein each source (101-1 to 101-N; 102-1 to 102-N) comprises a client computer, each destination (103-1 to 103-N) comprises a server and wherein the core network (104) is the Internet.
- 40 16. The computer network system according to claim 14, wherein the means for generating and transmitting a permission message to the at least one source is implemented in an authentication unit (113) which is located in a device connected to the gate (104) or in at least one destination.
17. The computer network system according to claim 16, wherein the authentication unit (113) is adapted to generate a permission message in form of a challenge to be solved by the at least one source, wherein only sources (101-1 to 101-N) having solved the challenge are able to mark traffic units validly.
- 45 18. A gate (109) used in a computer network system according to any one of the claims 14 to 17, comprising
- storage means for storing information related to a permission message,
 - 50 - means (117) for verifying traffic units on the basis of the information stored to determine whether or not the traffic units are marked validly,
 - means (117) for identifying each traffic unit being marked invalidly as unsolicited, and
 - means (117) for controlling the flow of each traffic unit being marked invalidly in response to at least one predetermined system parameter.
- 55

10

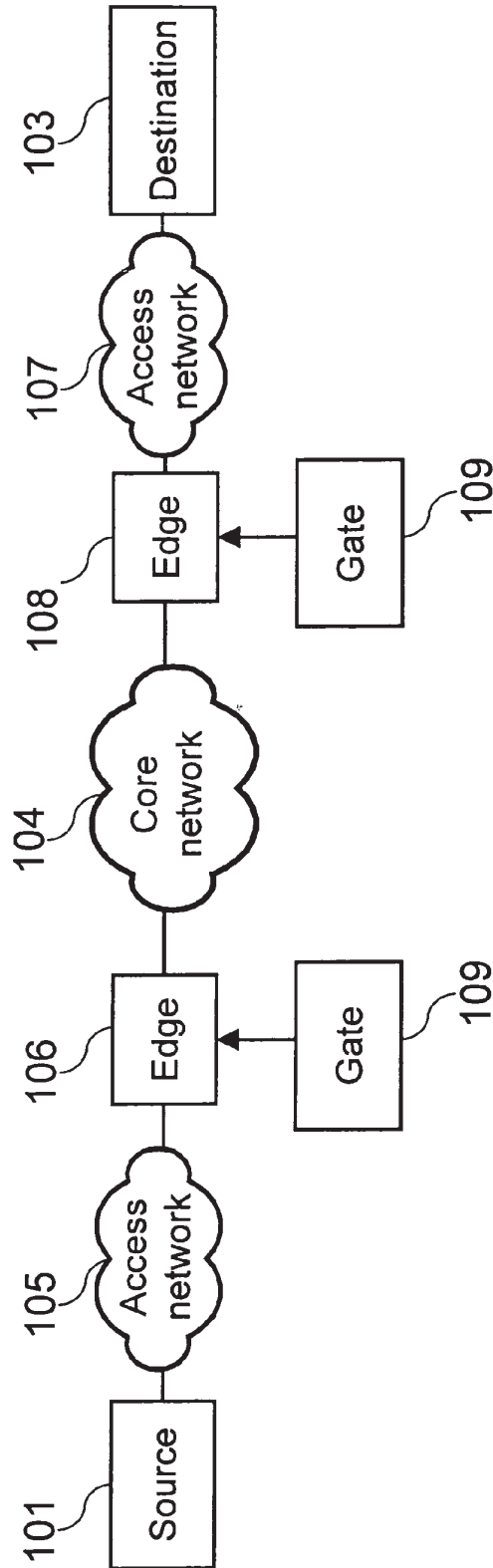


Fig. 1

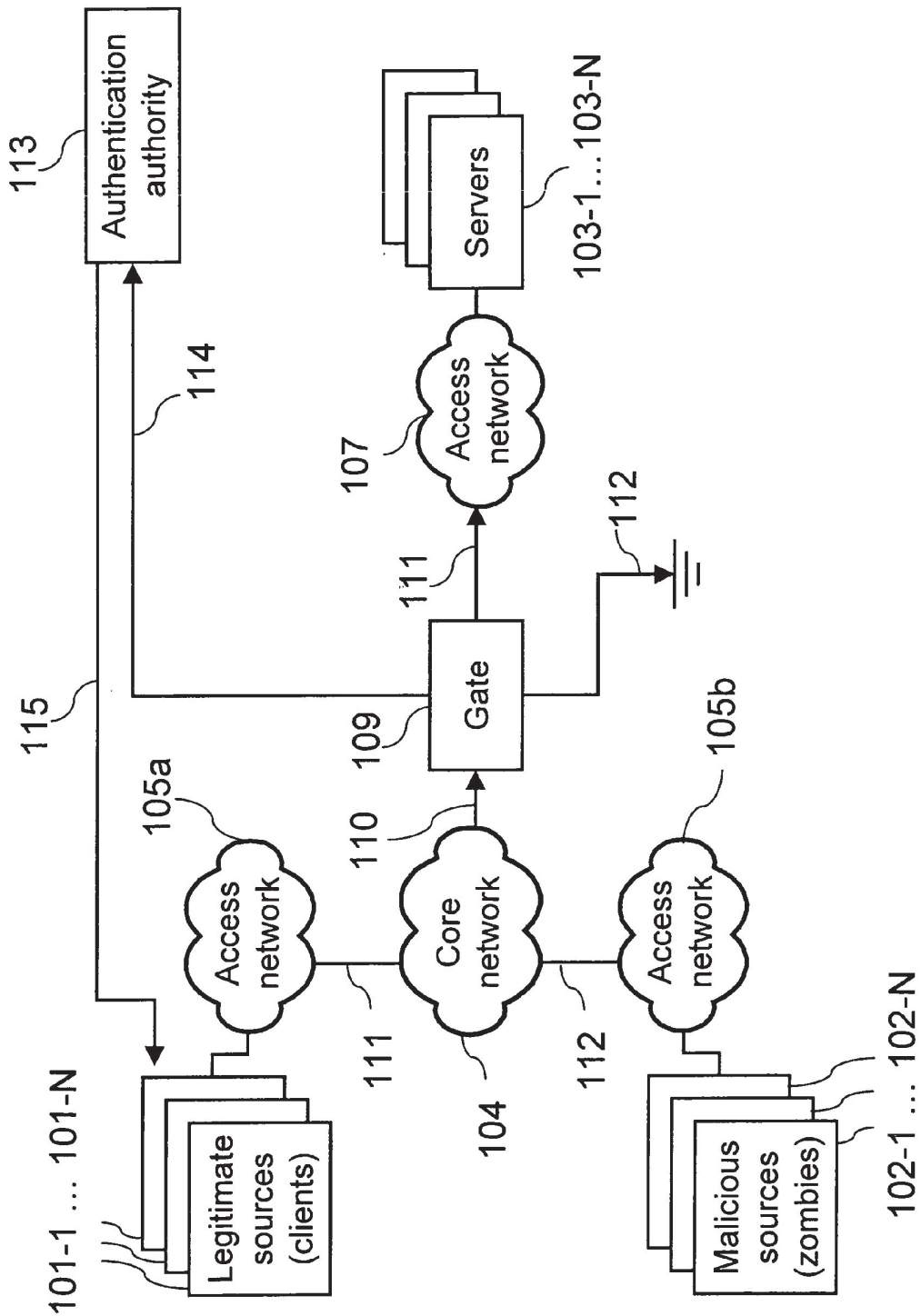


Fig. 2

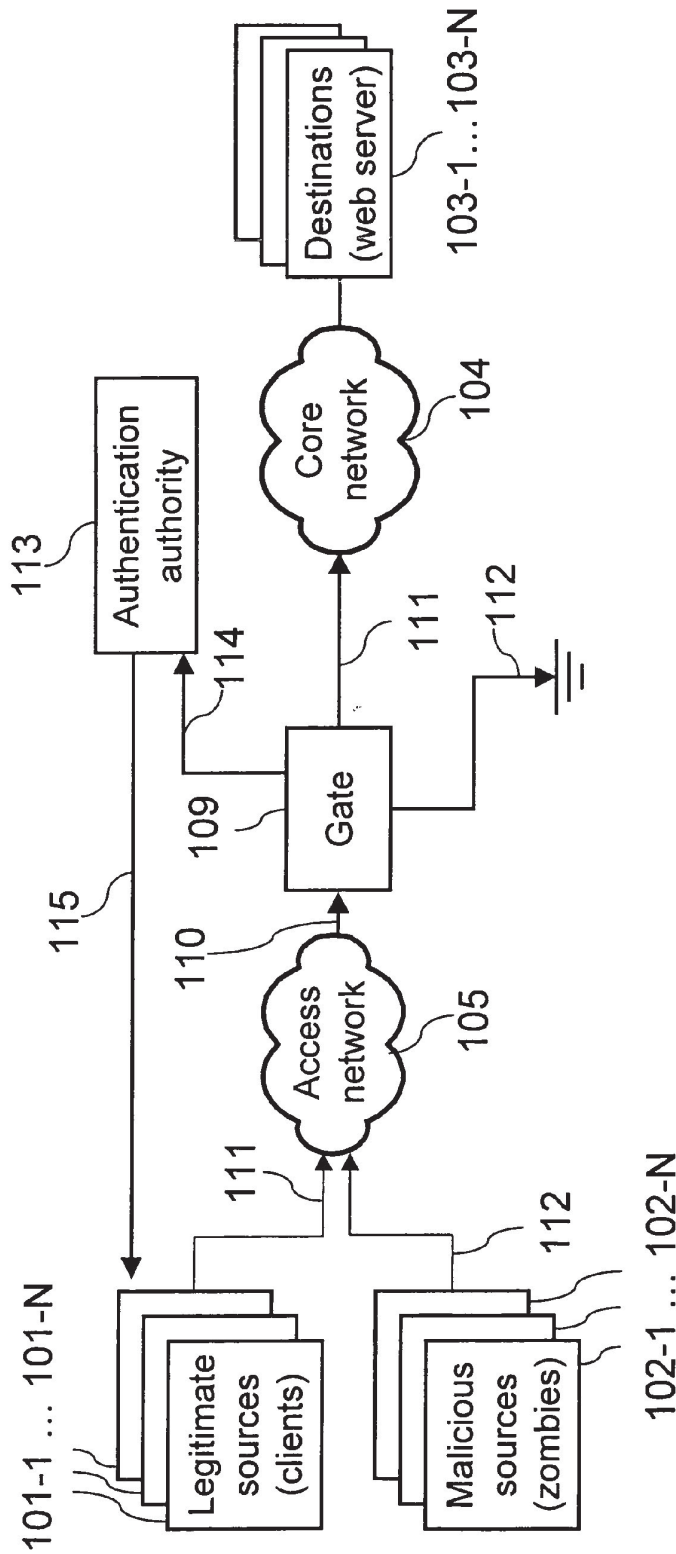


Fig. 3

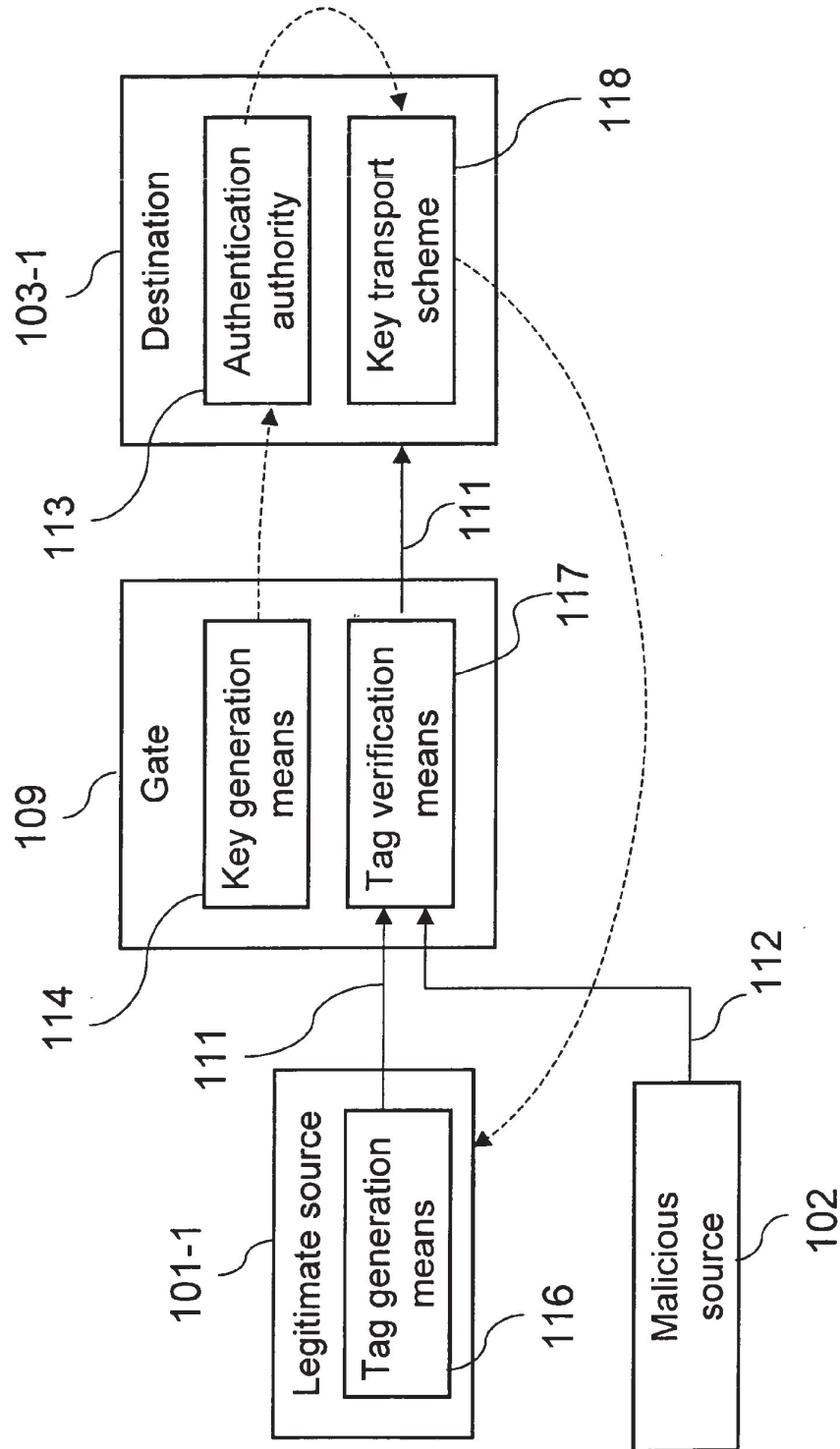


Fig. 4

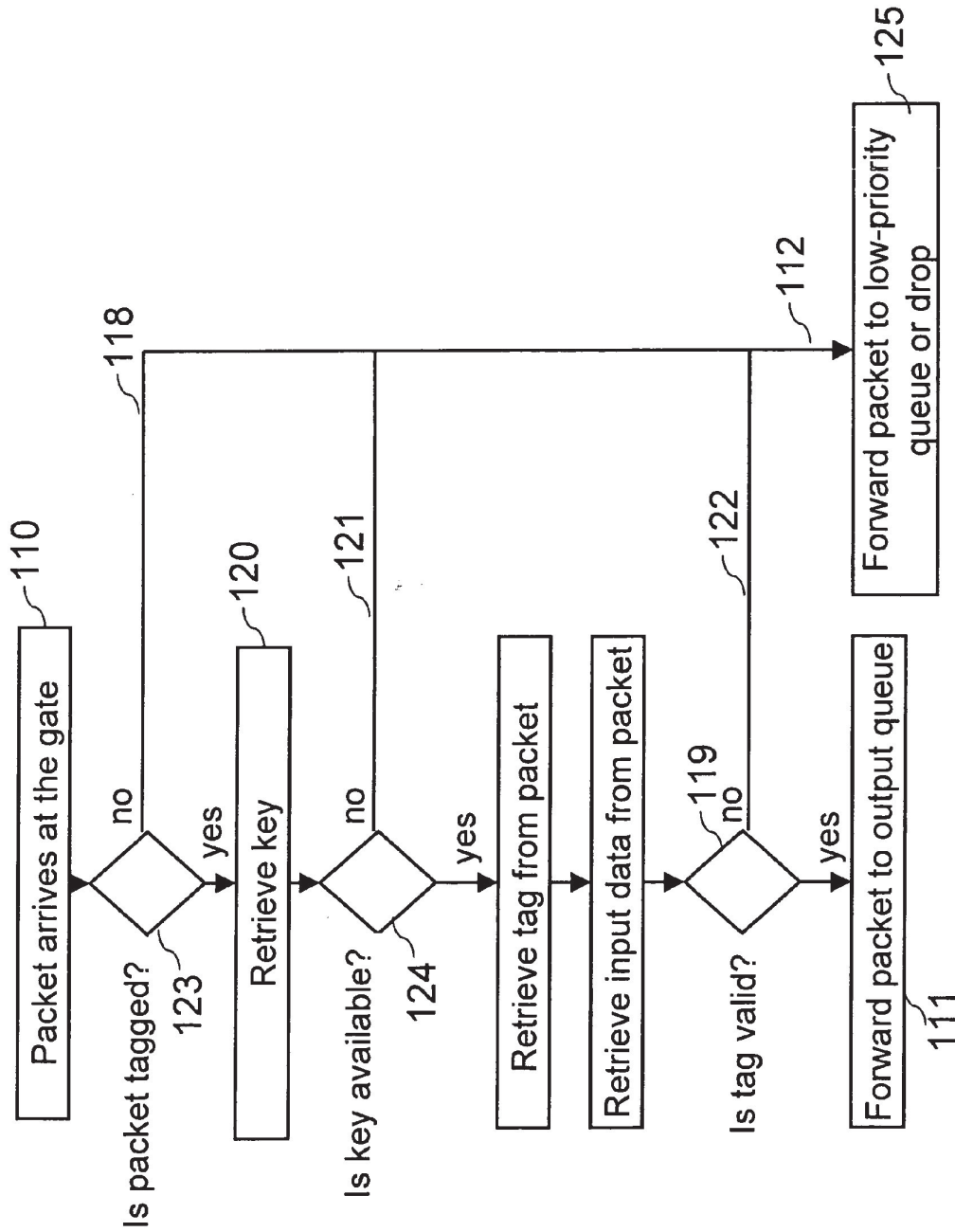


Fig. 5



European Patent
Office

EUROPEAN SEARCH REPORT

Application Number
EP 06 00 7693

DOCUMENTS CONSIDERED TO BE RELEVANT			
Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	US 2004/230797 A1 (OFEK YORAM [US] ET AL) 18 November 2004 (2004-11-18) * paragraphs [0085] - [0090] * * paragraphs [0098], [0099] * * paragraphs [0104] - [0110] * * paragraphs [0123] - [0128] * * paragraphs [0137], [0161] * * paragraphs [0225] - [0230] * * figures 1,2,4,5,8,9,18,19 * -----	1-18	INV. G06F21/00 H04L29/06
X	ANDERSON T ET AL: "PREVENTING INTERNET DENIAL-OF-SERVICE WITH CAPABILITIES" COMPUTER COMMUNICATION REVIEW, ACM, NEW YORK, NY, US, vol. 34, no. 1, January 2004 (2004-01), pages 39-44, XP001223359 ISSN: 0146-4833 * page 41, right-hand column, paragraph 2 * * page 42, left-hand column, line 14 - last line * * page 42, right-hand column, paragraph 4 - page 43, left-hand column, paragraph 1 * -----	1-18	TECHNICAL FIELDS SEARCHED (IPC) G06F H04L
D,A	EP 1 284 558 A1 (LUCENT TECHNOLOGIES INC [US]) 19 February 2003 (2003-02-19) * abstract * * paragraphs [0021], [0028] * * paragraphs [0034], [0035] * * figures 1-4 * -----	1-18	
The present search report has been drawn up for all claims			
Place of search Munich		Date of completion of the search 15 January 2007	Examiner BENGI-AKYUERER, K
CATEGORY OF CITED DOCUMENTS X : particularly relevant if taken alone Y : particularly relevant if combined with another document of the same category A : technological background O : non-written disclosure P : intermediate document T : theory or principle underlying the invention E : earlier patent document, but published on, or after the filing date D : document cited in the application L : document cited for other reasons & : member of the same patent family, corresponding document			

1
EPO FORM 1503 (03.02 (P04C01))

**ANNEX TO THE EUROPEAN SEARCH REPORT
ON EUROPEAN PATENT APPLICATION NO.**

EP 06 00 7693

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report.
The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

15-01-2007

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2004230797 A1	18-11-2004	NONE	

EP 1284558 A1	19-02-2003	DE 60201716 D1	02-12-2004
		DE 60201716 T2	22-12-2005
		US 2003036970 A1	20-02-2003

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- EP 1284558 B1 [0013]

Non-patent literature cited in the description

- **Y. KIM et al.** Packetscore: statistics-based overload control against distributed denial-of-service attacks. *Proceedings of IEEE INFOCOM'04*, March 2004 [0008]
- **S. SAVAGE et al.** Network support for IP-traceback. *IEEE/ACM Transactions on Networking*, June 2001, vol. 9 (3), 226-237 [0010]
- **A. YAAR et al.** A stateless internet flow filter to mitigate DDoS flooding attacks. *Proceedings of the IEEE Symposium on Security and Privacy*, May 2004 [0010]
- **X. YANG et al.** A DoS-limiting network architecture. *ACM SIGCOMM'05*, August 2005 [0011]
- **T. ANDERSON et al.** Preventing Internet Denial-of-Service with capabilities. *Proceedings of Hot Topics in Networking (HotNetsII)*, October 2003 [0012]
- CAPTCHA: Using Hard AI Problems for Security. **L. V. AHN et al.** *Advances in Cryptology-EUROCRYPT*. Springer Verlag, 2003, vol. 2656, 294-311 [0018]
- Using client puzzles to protect TLS. **D. DEAN et al.** *Proceedings of the 10th USENIX Security Symposium*. August 2001 [0018] [0083]
- **M. BELLARE et al.** The security of the cipher block chaining message authentication code. *Journal of Computer and Systems Sciences*, December 2000, vol. 61 (3), 362-399 [0076]
- CHAPTCHA: Using Hard AI Problems for Security. **L.V.AHN et al.** *Advances in Cryptology-EUROCRYPT*. Springer Verlag, 2003, vol. 2656, 294-311 [0078]
- Federal Information Processing Standards Publication (FIPS PUB 180-2. February 2004 [0088]

ATTACHMENT 10



(11) **EP 2 130 159 B1**

(12) **EUROPEAN PATENT SPECIFICATION**

(45) Date of publication and mention
of the grant of the patent:
19.06.2019 Bulletin 2019/25

(51) Int Cl.:
G06F 21/31 ^(2013.01) **H04L 9/32** ^(2006.01)

(21) Application number: **08730925.8**

(86) International application number:
PCT/US2008/055240

(22) Date of filing: **28.02.2008**

(87) International publication number:
WO 2008/118602 (02.10.2008 Gazette 2008/40)

(54) **SECURE DATA STORAGE AND RETRIEVAL INCORPORATING HUMAN PARTICIPATION**

SICHERE DATENSPEICHERUNG UND -ABRUFUNG UNTER MENSCHLICHER BETEILIGUNG

STOCKAGE DE DONNÉES SÉCURISÉES ET RÉCUPÉRATION AVEC PARTICIPATION HUMAINE

(84) Designated Contracting States:
**AT BE BG CH CY CZ DE DK EE ES FI FR GB GR
HR HU IE IS IT LI LT LU LV MC MT NL NO PL PT
RO SE SI SK TR**

(74) Representative: **Goddar, Heinz J.**
Boehmert & Boehmert
Anwaltpartnerschaft mbB
Pettenkoferstrasse 22
80336 München (DE)

(30) Priority: **23.03.2007 US 690685**

(43) Date of publication of application:
09.12.2009 Bulletin 2009/50

(56) References cited:
US-A- 5 921 548 US-A1- 2003 046 572
US-A1- 2004 146 164 US-A1- 2006 080 537
US-A1- 2006 287 963 US-B1- 6 754 827
US-B1- 6 944 765

(73) Proprietor: **Microsoft Technology Licensing, LLC**
Redmond, WA 98052 (US)

• **VON AHN L ET AL: "CAPTCHA: using hard AI
problems for security", SECURITY IN
COMMUNICATION NETWORKS : THIRD
INTERNATIONAL CONFERENCE ; REVISED
PAPERS / SCN 2002, AMALFI, ITALY,
SEPTEMBER 11 - 13, 2002; [LECTURE NOTES IN
COMPUTER SCIENCE , ISSN 0302-9743],
SPRINGER VERLAG, DE, vol. 2656, 8 May 2003
(2003-05-08), pages 294-311, XP009091663, ISBN:
978-3-540-24128-7**

(72) Inventors:
• **BAKER, Arthur H.**
Redmond, WA 98052-6399 (US)
• **GUARRACI, Brian J.**
Redmond, WA 98052-6399 (US)
• **TUCKER, Andrew Stewart**
Redmond, WA 98052-6399 (US)
• **MEDVINSKY, Gennady**
Redmond, WA 98052-6399 (US)
• **DUTTA, Tanmoy**
Redmond, WA 98052-6399 (US)

Note: Within nine months of the publication of the mention of the grant of the European patent in the European Patent Bulletin, any person may give notice to the European Patent Office of opposition to that patent, in accordance with the Implementing Regulations. Notice of opposition shall not be deemed to have been filed until the opposition fee has been paid. (Art. 99(1) European Patent Convention).

EP 2 130 159 B1

Description

TECHNICAL FIELD

[0001] The technical field relates generally to computer processing and more specifically to computer security.

BACKGROUND

[0002] Typical access-control mechanisms do not distinguish between input provided by people and input provided by software. As a result, rogue software running on a system can mimic the behavior of a human user and gain unauthorized access to protected resources. For example, a keystroke logger can capture a user's account name and password and later replay them as part of an access-verification sequence. Similarly, typical access-control mechanisms are susceptible to brute force attacks such as a dictionary attack or a "man in the middle" attack. Brute force attacks can be particularly detrimental to password protected storage of encrypted data.

[0003] US 5,921,548 A discloses puzzles characterized by one or a number of sets of visually identical, physically interchangeable, rotatable pieces that either contain magnets with varying north/south orientations embedded in their sides and/or have a mark or markings on one or more of their edges and computer versions of these puzzles. The object of the puzzles is to arrange the pieces in predetermined shapes and sequences in such a manner that the sides of the pieces which are in contact with the sides of other pieces have opposite magnetic poles facing each other so that the pieces attract, rather than repel or, if edge markings are used with or instead of magnets, so that adjacent edge markings comply with specified rules. In addition, the pieces may be all visually identical except for differences in edge markings, if any, or, alternatively, there may be several sets of visually identical pieces, e.g., pink pieces, blue pieces, yellow pieces and orange pieces, which together comprise a puzzle set. Because each solution to the puzzle is a particular sequence of the puzzle pieces, that sequence of pieces can be used in various ways to generate cryptographic keys which will enable ciphertext, which accompanies the puzzle, to be decoded. In addition, with some methods of generating cryptographic keys from the sequence of pieces in the solution to the puzzle, partial cryptographic keys are generated before the puzzle is completely solved, enabling partial solutions of the puzzle to be checked to determine if they are correct, thereby simplifying very difficult puzzles and making possible the solution of otherwise virtually impossible puzzles.

[0004] US 6,944,765 B1 discloses a method of authenticating anonymous users while reducing potential for "middleman" fraud includes the step of constructing a puzzle in response to information received from a software user. The puzzle includes the received information. The puzzle is sent to the user by a software provider. The user solves the puzzle and returns the solution to

the provider. The puzzle includes a portion of a value derived from an encrypted "cookie" and an exponentiation of the derived value. The cookie includes information about the user.

[0005] Von Ahn L et al: "CAPTCHA: using hard AI problems for security", Security in Communication Networks: Third International Conference; Revised Papers/SCN 2002, Amalfi, Italy, September 11-13, 2002; [LECTURE NOTES IN COMPUTER SCIENCE, ISSN 0302-9743], SPRINGER VERLAG, DE, vol. 2656, pages 294 - 311, discloses CAPTCHA, an automated test that humans can pass, but current computer programs can't pass. In particular, different applications of CAPTCHAs for practical security are introduced.

[0006] US 6,754,827 B1 discloses an implementation of CAPTCHA codes as an authentication means in the context of a data storage system.

SUMMARY

[0007] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the Detailed Description Of Illustrative Embodiments. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used to limit the scope of the claimed subject matter. The invention is defined in the appended claims.

[0008] A verification mechanism requires that a human being participate in an access verification sequence. To access data, a puzzle is provided. Proper solution of the puzzle requires human intervention. The puzzle is chosen such that its solution is within the capabilities of a human, but beyond the current state of the art for computer systems. In an example embodiment, the verification mechanism is utilized to ensure that data encrypted under a user password requires an actual human agent to participate in every attempt to access to the underlying cleartext data.

BRIEF DESCRIPTION OF THE DRAWINGS

[0009] The foregoing summary, as well as the following detailed description, is better understood when read in conjunction with the appended drawings. For the purpose of illustrating the secure data storage and retrieval incorporating human participation, there is shown in the drawings exemplary constructions thereof; however, providing functionality via the secure data storage and retrieval incorporating human participation is not limited to the specific methods and instrumentalities disclosed.

Figure 1 is a flow diagram of an example sequence of events for securely storing data.

Figure 2 is a flow diagram of an example sequence of events for securely retrieving data.

Figure 3 illustrates an example implementation of the storage sequence of the secure data storage and retrieval incorporating human participation.

Figure 4 illustrates an example implementation of the retrieval sequence of the secure data storage and retrieval incorporating human participation.

Figure 5 is a diagram of an exemplary processor for implementing the secure data storage and retrieval incorporating human participation.

Figure 6 is a depiction of an example suitable computing environment in which the secure data storage and retrieval incorporating human participation can be implemented.

DETAILED DESCRIPTION OF ILLUSTRATIVE EMBODIMENTS

[0010] Figure 1 and Figure 2 depict a sequential flow diagram of an example sequence of events for implementing a secure data storage and retrieval incorporating human participation. Figure 1 depicts an example sequence of events for storing secure data and Figure 2 depicts an example sequence of events for retrieving secure data. A cryptographic protocol for secure storage and retrieval of data requires authorization from a human being in order to perform data retrieval operations. In an example configuration, two agents are utilized to implement the protocol. As depicted in Figure 1, the first agent 14, a client, manages interactions with a human user 12. The second agent 16, a secure server (referred to as the human verification service or HVS), coordinates cryptographic operations and acts as a repository for protected data. The client agent 14 and the HVS server 16 are depicted separately, for the sake of clarity. It is to be understood however, that the client agent 14 and the HVS server 16 can comprise separate processors, the same processor, or a combination thereof. For example, the client agent 14 and HVS server 16 could be running in separate processes on a single computer system, running on separate hosts in a multi-computer network, or a combination thereof.

[0011] As mentioned above, Figure 1 is a flow diagram of an example sequence of events for storing secure data. At step 18, the client agent 14 generates a cryptographic key, K . The cryptographic key can comprise any appropriate random cryptographic key. In an example embodiment, the cryptographic key, K , is a symmetric key (e.g., a symmetric key in accordance with the Advanced Encryption Standard, AES). At step 20, the plain text of the data to be stored is encrypted with the cryptographic key, K , to generate encrypted data. Another cryptographic key, K_1 , is generated by the client agent 14 at step 22. The cryptographic key, K_1 , can comprise any appropriate cryptographic key, such as a symmetric cryptographic key, for example. It is to be understood

that the sequence of events as depicted in the Figures is exemplary and thus not limited thereto. The sequence of events can be modified as appropriate. For example, step 22 could be performed before step 18 or step 20.

[0012] The client agent 14 queries the user, at step 24, for a secret, such as a PIN, a password, or the like. The user responds to the query with the user's secret, S_U , at step 26. The client agent 14 operates on the user's secret, S_U , with a cryptographic hash function, at step 28. Hash functions are known in the art. A hash function is a function that transforms a variable-size input into a fixed size value. Typically, hash functions are one way, meaning that it is impracticable or impossible to determine the input value from the output (transformed) value. Providing the same input to a hash function will provide the same output. A slight change in the input typically results in a considerable change in the output. Example hash functions include MD2, MD5, SHA-1, SHA-224, SHA-256, SHA-384, and SHA-512.

[0013] At step 30, the cryptographic key, K , is encrypted with the cryptographic key, K_1 . At step 32, the Hash value indicative of the user's secret, $\text{Hash}(S_U)$, is encrypted with the cryptographic key, K_1 . In an example embodiment, the user's secret, S_U , is not stored in the client agent 14 after the hash function is performed. This can provide added security because the user's secret, S_U , is less susceptible to theft. At step 34, the cryptographic key, K_1 , is encrypted with the public key, of a public/private cryptographic key pair of the secure server 16. Public key cryptography, which utilizes a public/private key pair, is known in the art. One key is used to encrypt and the other is used to decrypt. Knowledge of one key does not provide knowledge of the other key. Typically one key is kept secret, and thus called the private key. The other key typically is made public. An example of public key cryptography is RSA public key cryptography. The information encrypted at steps 30, 32, and 33, referred to as a blob in Figure 1, is provided to the secure server 16, at step 36. That is, at step 36, the client agent 14 provides to the secure server 16, the cryptographic key, K , encrypted with the cryptographic key, K_1 , the hash value indicative of the user's secret, $\text{Hash}(S_U)$, encrypted with the cryptographic key, K_1 , and the cryptographic key, K_1 , encrypted with the public key, of the secure server's 16 public/private cryptographic key pair.

[0014] At step 38, the secure server 16 generates a unique tag value, referred to as a retrieval tag in Figure 1, to identify the blob received at step 36. The blob is stored in a database, memory, or the like of the secure server at step 40. The blob is indexed by the retrieval tag. Thus, the blob can be retrieved from the database, memory, or the like, via the retrieval tag. At step 42, the secure server 16 provides the retrieval tag to the client agent 14. At step 44, the client agent 14 stores the encrypted data generated at step 20 using the retrieval tag as an index. That is, the client agent 14 stores in a database, memory, or the like the plain text encrypted with the cryptographic key, K , and the stored encrypted data

is indexed utilizing the retrieval tag such that the retrieval tag can be used to retrieve the encrypted data from storage.

[0015] Figure 2 is a flow diagram of an example sequence of events for securely retrieving data. To retrieve the original plain text, the client agent 14 utilizes the retrieval tag that was generated (at step 38) during the storage process to access the secure server 16. A secure channel is established, at step 46, between the client agent 14 and the secure server 16. The secure channel can be established in accordance with any appropriate means. At step 48, the client agent 14 sends the retrieval tag and a request for a puzzle to the secure server 16.

[0016] A puzzle and the correct solution to the puzzle are generated at step 50. The puzzle can comprise any appropriate type of puzzle requiring human intervention to obtain a solution. In various embodiments, the puzzle can require a user to respond to a visual rendering, an audio rendering, or a combination thereof. For example, the puzzle could require a user to enter letters observed in a distorted image (e.g., a completely automated public Turing test, CAPTCHA). The puzzle could require a user to answer a question observed in a distorted image. The puzzle could require a user to enter letters heard in distorted audio. The puzzle could require a user to answer a question heard in distorted audio. In various embodiments, the user's response can be provided via a keyboard, a mouse, orally, or a combination thereof. The puzzle is chosen such that its solution is within the capabilities of a human, but beyond the current state of the art for computer systems. The puzzle can be generated by the secure server 16, by another processor, or a combination thereof. In an example embodiment, the puzzle is obtained via a library of pluggable puzzle generators. This embodiment allows puzzles to be replaced as the state of the art of computing technology makes some puzzles more susceptible to mechanical solution. In various embodiments, the puzzle can be generated by the secure server 16 each time a request is provided thereto, the puzzle can be a predetermined puzzle, a puzzle type (e.g., puzzle category) can be predetermined, or a combination thereof. If the puzzle is a predetermined puzzle or the puzzle type is predetermined, an indication of a puzzle or puzzle type could be provided at step 48.

[0017] At step 52, the secure server 16 generates a random cryptographic key, K_2 . The cryptographic key, K_2 , can comprise any appropriate cryptographic key, such as a symmetric cryptographic key, for example. At step 54, the correct solution to the puzzle is encrypted with the cryptographic key, K_2 . In an example embodiment, a timestamp is included with the correct solution and encrypted with the cryptographic key, K_2 . The encrypted correct solution and optional timestamp, referred to as a verification blob, and the puzzle, are provided by the secure server 16 to the client agent 14, at step 56. The puzzle is provided by the client agent 14 to the user 12 at step 58. The user 12 solves the puzzle incorporating human participation and provides a proposed solution to

the client agent 14, at step 60. The user 12 is queried for the user's secret at step 62. The user 12 responds to the query (step 62) with the user's secret, S_U , at step 64. At step 66, the client agent 14, sends to the secure server 16, the proposed solution (received at step 60), the hash value indicative of the user's secret ($\text{Hash}(S_U)$), and the verification blob (the encrypted correct solution and optional timestamp generated at step 54).

[0018] The secure server 16, at step 68, retrieves the indexed client blob utilizing the retrieval tag. That is, the secure server 16 retrieves, using the retrieval tag generated at step 38, the cryptographic key, K , encrypted with the cryptographic key, K_1 , the hash value indicative of the user's secret, $\text{Hash}(S_U)$, encrypted with the cryptographic key, K_1 , and the cryptographic key, K_1 , encrypted with the public key, of the secure server's 16 public/private cryptographic key pair, stored at step 40. At step 70, the cryptographic key, K_1 , that was encrypted (at step 34) with the public key, PK_{HVS} is decrypted with the private key, PK^{-1}_{HVS} , of the secure server's 16 public/private cryptographic key pair. At step 72, the hash value indicative of the user's secret ($\text{Hash}(S_U)$), that was encrypted (at step 32) is decrypted with cryptographic key, K_1 . At step 74, the verification blob (correct solution and optional timestamp generated) that was encrypted (at step 54) is decrypted with the cryptographic key, K_2 .

[0019] Verification is performed at step 76. If verification is successful, access to the stored data (plain text) is allowed to. If verification is unsuccessful, access to the data (plain text) is not allowed. In an example embodiment, verification fails if the hash value indicative of the user's secret provided at step 64 differs from the hash value indicative of the user's secret provided at step 26. That is, verification fails if the user's secret provided when the client agent 14 first queries the user 12 differs from the user's secret provided with the user's proposed solution to the puzzle. In an example embodiment, the user 12 is given a limited period of time to provide a proposed response to the puzzle. Accordingly, the verification fails if the current time, i.e., the time at which the secure server 16 is performing verification, differs from the time indicated by the time stamp in the verification blob (generated at step 54) by greater than a predetermined amount of time. For example, if the current time differs from the time indicated by the time stamp by more than 30 seconds, verification will fail. Verification fails if the proposed solution to the puzzle provided by the user 12 (at step 60) differs from the correct solution.

[0020] Figure 3 and Figure 4 illustrate an example implementation of the secure data storage and retrieval incorporating human participation utilizing a passive Universal Serial Bus (USB) device 90 as storage for the storage management application 86. Figure 3 illustrates an example implementation of the storage sequence of the secure data storage and retrieval incorporating human participation. The implementation comprises a storage management application 86 (e.g., the client agent), a human verification service (HVS) 88 (e.g., the secure HVS

server), the storage medium 90, and storage medium 92. The storage medium 90 can comprise any appropriate storage medium, such as a disk, semiconductor memory, a database, or the like, for example. As depicted in Figure 3, in an example embodiment, the storage medium 90 comprises a USB storage device (e.g., a key drive). The storage device 90 requires no special cryptographic capabilities. The storage medium 92 can comprise any appropriate storage medium, such as a disk, semiconductor memory, a database, or the like, for example.

[0021] To accomplish storage of secure data, the storage-management application 86 accepts a user data-key to be stored, a password, and a friendly display name from the user. A user data key can comprise any appropriate secret that guards sensitive user data. For example, a data key can comprise a password, an encryption, of the like. The display name functions as a reminder to the user of the identity of the user data-key. the display name is not be encrypted. After executing the storage sequence of the secure data storage and retrieval incorporating human participation as described above, the storage-management application 86 possesses a retrieval tag and a copy of the user-data key that has been encrypted with the cryptographic key, K. The storage management application 86 stores both the retrieval tag and the copy of the user-data key along with the data key's friendly display name on the USB device 90.

[0022] After executing the of the storage sequence, as described above, of the storage and retrieval incorporating human participation, the HVS 88 possesses a copy of the cryptographic key, K, and the hash value indicative of the user's secret, S_U , both of which are encrypted using the cryptographic key K_1 . The HVS 88 also possesses an encrypted copy of K_1 that the HVS 88 can recover by using the HVS's 88 private key. These items are stored as a group and indexed with the retrieval tag value. Taken together, this data will allow the HVS 88 to recreate the value of K and the hash value indicative of the user's secret.

[0023] Figure 4 illustrates an example implementation of the retrieval sequence of the secure data storage and retrieval incorporating human participation. When the user wants to recover the data key, the user executes the storage management application 86 again. The storage-management application 86 presents a list of friendly display names loaded from the USB device 90 to the user. The user can select one of the display names to recover the associated data key(s). During the retrieval sequence of the secure data storage and retrieval incorporating human participation, the storage management application 86 prompts the user for a password or the like and provides to the user a puzzle requiring human participation to obtain a solution. Upon receiving the user's response (e.g., the user's proposed solution to the puzzle), the storage management application 86, sends the user's response to the HVS 88. The HVS 88 evaluates the password and user's proposed puzzle solution. If the HVS determines that the password is correct and the user's

proposed puzzle solution is correct, the HVS 88 returns the original key, K, to the storage management application 88. The storage management application 86 uses cryptographic key, K, to decrypt the original user data key from the USB storage device 90. At no point during the storage sequence or the retrieval sequence, does the HVS 88 have access to the plaintext version of the user's data key.

[0024] Figure 5 is a diagram of an exemplary processor 94 for implementing the secure data storage and retrieval incorporating human participation. The processor 94 comprises a processing portion 96, a memory portion 98, and an input/output portion 100. The processing portion 96, memory portion 98, and input/output portion 100 are coupled together (coupling not shown in Figure 5) to allow communications therebetween. The input/output portion 100 is capable of providing and/or receiving components utilized to implement the secure data storage and retrieval incorporating human participation as described above. For example, the input/output portion 100 is cable of receiving and/or providing, a query for a user's secret, a user's secret, encrypted information, a cryptographic key, indexed information, a hash value, a retrieval tag, a puzzle, a proposed puzzle solution, or a combination thereof.

[0025] The processing portion 96 is capable of implementing the secure data storage and retrieval incorporating human participation as described above. For example, the processing portion 96 is capable of generating a cryptographic key, performing encryption, performing a hash function, generating a retrieval tag, storing an indexed information, establishing a secure channel, generating a puzzle requiring human participation to obtain a solution, retrieving indexed information, performing decryption, performing verification, determining if the current time differs from the time indicated by the time stamp by at least a predetermined amount of time, determining if a first user's secret differs from a second user's secret, determining if the proposed puzzle solution is correct, or a combination thereof.

[0026] The processor 94 can be implemented as a client processor and/or a server processor. In a basic configuration, the processor 94 can include at least one processing portion 96 and memory portion 98. The memory portion 98 can store any information utilized in conjunction with implementing the secure data storage and retrieval incorporating human participation. For example, the memory portion 98 can store a cryptographic key, a user's secret, a puzzle, a puzzle solution, a hash value, indexed information, a retrieval tag, or a combination thereof. Depending upon the exact configuration and type of processor, the memory portion 98 can be volatile (such as RAM) 102, non-volatile (such as ROM, flash memory, etc.) 104, or a combination thereof. The processor 94 can have additional features/functionality. For example, the processor 94 can include additional storage (removable storage 106 and/or non-removable storage 108) including, but not limited to, magnetic or optical

disks, tape, flash, smart cards or a combination thereof. Computer storage media, such as memory portion 98, 102, 104, 106, and 108, include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer readable instructions, data structures, program modules, or other data. Computer storage media include, but are not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, universal serial bus (USB) compatible memory, smart cards, or any other medium which can be used to store the desired information and which can be accessed by the processor 94. Any such computer storage media can be part of the processor 94.

[0027] The processor 94 can also contain communications connection(s) 114 that allow the processor 94 to communicate with other devices, for example. Communications connection(s) 114 is an example of communication media. Communication media typically embody computer readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. The term computer readable media as used herein includes both storage media and communication media. The processor 94 also can have input device(s) 112 such as keyboard, mouse, pen, voice input device, touch input device, *etc.* Output device(s) 110 such as a display, speakers, printer, *etc.* also can be included.

[0028] Figure 6 and the following discussion provide a brief general description of an example suitable computing environment in which the secure data storage and retrieval incorporating human participation can be implemented. Although not required, various aspects of the secure data storage and retrieval incorporating human participation can be described in the general context of computer executable instructions, such as program modules, being executed by a computer, such as a client workstation or a server. Generally, program modules include routines, programs, objects, components, data structures and the like that perform particular tasks or implement particular abstract data types. Moreover, implementation of the secure data storage and retrieval incorporating human participation can be practiced with other computer system configurations, including hand held devices, multi processor systems, microprocessor based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like.

Further, the provision of functionality via the secure data storage and retrieval incorporating human participation also can be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules can be located in both local and remote memory storage devices.

[0029] A computer system can be roughly divided into three component groups: the hardware component, the hardware/software interface system component, and the applications programs component (also referred to as the "user component" or "software component"). In various embodiments of a computer system the hardware component may comprise the central processing unit (CPU) 521, the memory (both ROM 564 and RAM 525), the basic input/output system (BIOS) 566, and various input/output (I/O) devices such as a keyboard 540, a mouse 542, a monitor 547, and/or a printer (not shown), among other things. The hardware component comprises the basic physical infrastructure for the computer system.

[0030] The applications programs component comprises various software programs including but not limited to compilers, database systems, word processors, business programs, videogames, and so forth. Application programs provide the means by which computer resources are utilized to solve problems, provide solutions, and process data for various users (machines, other computer systems, and/or end-users). In an example embodiment, application programs perform the functions associated with the secure data storage and retrieval incorporating human participation as described above.

[0031] The hardware/software interface system component comprises (and, in some embodiments, may solely consist of) an operating system that itself comprises, in most cases, a shell and a kernel. An "operating system" (OS) is a special program that acts as an intermediary between application programs and computer hardware. The hardware/software interface system component may also comprise a virtual machine manager (VMM), a Common Language Runtime (CLR) or its functional equivalent, a Java Virtual Machine (JVM) or its functional equivalent, or other such software components in the place of or in addition to the operating system in a computer system. A purpose of a hardware/software interface system is to provide an environment in which a user can execute application programs.

[0032] The hardware/software interface system is generally loaded into a computer system at startup and thereafter manages all of the application programs in the computer system. The application programs interact with the hardware/software interface system by requesting services via an application program interface (API). Some application programs enable end-users to interact with the hardware/software interface system via a user interface such as a command language or a graphical user interface (GUI).

[0033] A hardware/software interface system traditionally performs a variety of services for applications. In a multitasking hardware/software interface system where multiple programs may be running at the same time, the hardware/software interface system determines which applications should run in what order and how much time should be allowed for each application before switching to another application for a turn. The hardware/software interface system also manages the sharing of internal memory among multiple applications, and handles input and output to and from attached hardware devices such as hard disks, printers, and dial-up ports. The hardware/software interface system also sends messages to each application (and, in certain case, to the end-user) regarding the status of operations and any errors that may have occurred. The hardware/software interface system can also offload the management of batch jobs (e.g., printing) so that the initiating application is freed from this work and can resume other processing and/or operations. On computers that can provide parallel processing, a hardware/software interface system also manages dividing a program so that it runs on more than one processor at a time.

[0034] A hardware/software interface system shell (referred to as a "shell") is an interactive end-user interface to a hardware/software interface system. (A shell may also be referred to as a "command interpreter" or, in an operating system, as an "operating system shell"). A shell is the outer layer of a hardware/software interface system that is directly accessible by application programs and/or end-users. In contrast to a shell, a kernel is a hardware/software interface system's innermost layer that interacts directly with the hardware components.

[0035] As shown in Figure 6, an exemplary general purpose computing system includes a conventional computing device 560 or the like, including a processing unit 521, a system memory 562, and a system bus 523 that couples various system components including the system memory to the processing unit 521. The system bus 523 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. The system memory includes read only memory (ROM) 564 and random access memory (RAM) 525. A basic input/output system 566 (BIOS), containing basic routines that help to transfer information between elements within the computing device 560, such as during start up, is stored in ROM 564. The computing device 560 may further include a hard disk drive 527 for reading from and writing to a hard disk (hard disk not shown), a magnetic disk drive 528 (e.g., floppy drive) for reading from or writing to a removable magnetic disk 529 (e.g., floppy disk, removal storage), and an optical disk drive 530 for reading from or writing to a removable optical disk 531 such as a CD ROM or other optical media. The hard disk drive 527, magnetic disk drive 528, and optical disk drive 530 are connected to the system bus 523 by a hard disk drive interface 532, a magnetic disk drive interface 533, and

an optical drive interface 534, respectively. The drives and their associated computer readable media provide non volatile storage of computer readable instructions, data structures, program modules and other data for the computing device 560. Although the exemplary environment described herein employs a hard disk, a removable magnetic disk 529, and a removable optical disk 531, it should be appreciated by those skilled in the art that other types of computer readable media which can store data that is accessible by a computer, such as magnetic cassettes, flash memory cards, digital video disks, Bernoulli cartridges, random access memories (RAMs), read only memories (ROMs), and the like may also be used in the exemplary operating environment. Likewise, the exemplary environment may also include many types of monitoring devices such as heat sensors and security or fire alarm systems, and other sources of information.

[0036] A number of program modules can be stored on the hard disk, magnetic disk 529, optical disk 531, ROM 564, or RAM 525, including an operating system 535, one or more application programs 536, other program modules 537, and program data 538. A user may enter commands and information into the computing device 560 through input devices such as a keyboard 540 and pointing device 542 (e.g., mouse). Other input devices (not shown) may include a microphone, joystick, game pad, satellite disk, scanner, or the like. These and other input devices are often connected to the processing unit 521 through a serial port interface 546 that is coupled to the system bus, but may be connected by other interfaces, such as a parallel port, game port, or universal serial bus (USB). A monitor 547 or other type of display device is also connected to the system bus 523 via an interface, such as a video adapter 548. In addition to the monitor 547, computing devices typically include other peripheral output devices (not shown), such as speakers and printers. The exemplary environment of Figure 6 also includes a host adapter 555, Small Computer System Interface (SCSI) bus 556, and an external storage device 562 connected to the SCSI bus 556.

[0037] The computing device 560 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 549. The remote computer 549 may be another computing device (e.g., personal computer), a server, a router, a network PC, a peer device, or other common network node, and typically includes many or all of the elements described above relative to the computing device 560, although only a memory storage device 550 (floppy drive) has been illustrated in Figure 6. The logical connections depicted in Figure 6 include a local area network (LAN) 551 and a wide area network (WAN) 552. Such networking environments are commonplace in offices, enterprise wide computer networks, intranets and the Internet.

[0038] When used in a LAN networking environment, the computing device 560 is connected to the LAN 551 through a network interface or adapter 553. When used in a WAN networking environment, the computing device

560 can include a modem 554 or other means for establishing communications over the wide area network 552, such as the Internet. The modem 554, which may be internal or external, is connected to the system bus 523 via the serial port interface 546. In a networked environment, program modules depicted relative to the computing device 560, or portions thereof, may be stored in the remote memory storage device. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

[0039] While it is envisioned that numerous embodiments of the secure data storage and retrieval incorporating human participation are particularly well-suited for computerized systems, nothing in this document is intended to limit the invention to such embodiments. On the contrary, as used herein the term "computer system" is intended to encompass any and all devices capable of storing and processing information and/or capable of using the stored information to control the behavior or execution of the device itself, regardless of whether such devices are electronic, mechanical, logical, or virtual in nature.

[0040] The various techniques described herein can be implemented in connection with hardware or software or, where appropriate, with a combination of both. Thus, the methods and apparatuses for implementing the provision of functionality the secure data storage and retrieval incorporating human participation, or certain aspects or portions thereof, can take the form of program code (i.e., instructions) embodied in tangible media, such as floppy diskettes, CD-ROMs, hard drives, or any other machine-readable storage medium, wherein, when the program code is loaded into and executed by a machine, such as a computer, the machine becomes an apparatus for providing functionality via the secure data storage and retrieval incorporating human participation.

[0041] The program(s) can be implemented in assembly or machine language, if desired. In any case, the language can be a compiled or interpreted language, and combined with hardware implementations. The methods and apparatuses for implementing the secure data storage and retrieval incorporating human participation also can be practiced via communications embodied in the form of program code that is transmitted over some transmission medium, such as over electrical wiring or cabling, through fiber optics, or via any other form of transmission, wherein, when the program code is received and loaded into and executed by a machine, such as an EPROM, a gate array, a programmable logic device (PLD), a client computer, or the like. When implemented on a general-purpose processor, the program code combines with the processor to provide a unique apparatus that operates to invoke the functionality of the secure data storage and retrieval incorporating human participation. Additionally, any storage techniques used in connection with the secure data storage and retrieval incorporating human participation can invariably be a combination of hardware

and software.

[0042] While secure data storage and retrieval incorporating human participation has been described in connection with the example embodiments of the various figures, it is to be understood that other similar embodiments can be used or modifications and additions can be made to the described embodiments for performing the same functions of secure data storage and retrieval incorporating human participation without deviating therefrom. Therefore, secure data storage and retrieval incorporating human participation as described herein should not be limited to any single embodiment, but rather should be construed in breadth and scope in accordance with the appended claims.

Claims

1. A computer related security method incorporating human participation, the method comprising:

receiving a request for data stored in a storage medium;
responsive to the request, transmitting (48) from a client agent (14) to a server (16), a puzzle request and a retrieval tag, wherein the retrieval tag is suitable for enabling the retrieval of a verification blob comprising an encrypted solution to a puzzle;
receiving (56), at the client agent (14), from the server (16), the puzzle, wherein human participation is intended to obtain a solution to the puzzle, and a correct solution to the puzzle;
transmitting (58) the puzzle from the client agent (14) to a user (12);
receiving (60), from the user (12), a proposed solution to the puzzle;
transmitting (62) to the user, a query seeking identification of a stored secret;
receiving (64) from the user, a response to the query seeking identification of the stored secret;
transmitting (66) to the server, the proposed solution to the puzzle and the response to the query;
using (68) in the server, the retrieval tag received from the client agent to retrieve the verification blob;
determining if the proposed solution and the response to the query is correct; and
if the proposed solution and the response to the query is correct, allowing access to the data stored in the storage medium.

2. A method in accordance with claim 1, further comprising, if an amount of time between providing the puzzle and receiving the proposed solution is greater than a predetermined amount of time, not allowing access to the data.

3. A method in accordance with claim 1, further comprising:

prior to receiving (60) the proposed solution, requesting a first secret; 5
 receiving the first secret;
 subsequent to receiving the request for data, requesting a second secret;
 receiving the second secret; 10
 comparing the first secret with the second secret; and
 if the first secret does not match the second secret, not allowing access to the data.

4. A method in accordance with claim 1, wherein: 15

the puzzle is intended to be visually rendered; and
 a solution to the puzzle is intended to be obtained via observation of the visually rendering of the puzzle. 20

5. A method in accordance with claim 1, wherein:

the puzzle is intended to be audibly rendered; and 25
 a solution to the puzzle is intended to be obtained via observation of the audio rendering of the puzzle. 30

6. A method in accordance with claim 1, further comprising obtaining the puzzle from a library of pluggable puzzle generators.

7. A method in accordance with claim 1, further comprising: 35

encrypting the data with a first cryptographic key;
 encrypting the first cryptographic key with a second cryptographic key; 40
 storing in the storage medium, the encrypted data and the encrypted first cryptographic key; and
 if the proposed solution is correct: 45

decrypting, with the second cryptographic key, the encrypted first cryptographic key; and
 providing the decrypted first cryptographic key. 50

8. A system for implementing computer related security incorporating human participation, the system comprising: 55

an input/output portion configured to receive a request for data stored in a memory portion;
 a first processing portion of a client agent (14),

the first processing portion configured to:

responsive to the request, transmit (48) from the client agent (14) to a server (16), a puzzle request and a retrieval tag, wherein the retrieval tag is suitable for enabling the retrieval of a verification blob comprising an encrypted solution to a puzzle;
 receive (56), at the client agent (14), from a server (16), the puzzle, wherein human participation is intended to obtain a solution to the puzzle, and a correct solution to the puzzle;
 transmit (58) the puzzle from the client agent (14) to a user (12);
 and
 receive, from the user (12), a proposed solution to the puzzle;
 transmit (62) to the user, a query seeking identification of a stored secret;
 receive (64) from the user, a response to the query seeking identification of the stored secret;
 transmit (66) to the server, the proposed solution to the puzzle and the response to the query;

a second processing portion of the server, the second processing portion configured to:

using (68) in the server, the retrieval tag received from the client agent to retrieve the verification blob;
 determine if the proposed solution and the response to the query is correct; and if the proposed solution and the response to the query is correct, allow access to the data stored in the memory portion; and
 the memory portion configured to store the data.

9. A system in accordance with claim 8, wherein the processing portion is further configured to, if an amount of time between providing the puzzle and receiving the proposed solution is greater than a predetermined amount of time, not allow access to the data.

10. A system in accordance with claim 8, wherein:

the processing portion is further configured to:

prior to receiving the proposed solution, request a first secret;
 subsequent to receiving the request for data, request a second secret;
 compare the first secret with the second secret; and
 if the first secret does not match the second

- secret, not allow access to the data; and
- the input/output portion is further configured to:
- receive the first secret; and 5
receive the second secret.
11. A system in accordance with claim 8, wherein:
- the puzzle is intended to be visually rendered; 10
and
a solution to the puzzle is intended to be obtained via observation of the visually rendering of the puzzle. 15
12. A system in accordance with claim 8, wherein:
- the puzzle is intended to be audibly rendered; and
a solution to the puzzle is intended to be obtained via observation of the audio rendering of the puzzle. 20
13. A system in accordance with claim 8, the processing portion further configured to obtain the puzzle from a library of pluggable puzzle generators. 25
14. A system in accordance with claim 8, wherein:
- the processing portion is further configured to: 30
- encrypt the data with a first cryptographic key;
encrypt the first cryptographic key with a second cryptographic key; 35
store in the memory portion, the encrypted data and the encrypted first cryptographic key; and
if the proposed solution is correct, decrypt, with the second cryptographic key, the encrypted first cryptographic key; and 40
- the input/output portion is further configured to provide the decrypted first cryptographic key. 45
15. A computer-readable medium having stored thereon computer-executable instructions for performing computer related security incorporating human participation, by performing the method in accordance with one of the claims 1 to 7. 50

Patentansprüche

1. Computerbezogenes Sicherheitsverfahren mit menschlicher Beteiligung, wobei das Verfahren umfasst:

Empfangen einer Anforderung für Daten, die auf einem Speichermedium gespeichert sind;
als Reaktion auf die Anforderung, Übertragen (48) einer Rätselanforderung und eines Abruf-Tags von einem Client-Agenten (14) an einen Server (16), wobei das Abruf-Tag geeignet ist, das Abrufen eines Verifizierungsblobs zu ermöglichen, der eine verschlüsselte Lösung für ein Puzzle umfasst;
Empfangen (56) des Puzzles beim Client-Agenten (14) vom Server (16), wobei die menschliche Teilnahme dazu bestimmt ist, eine Lösung für das Puzzle und eine korrekte Lösung für das Puzzle zu erhalten;
Übertragen (58) des Puzzles von dem Client-Agenten (14) zu einem Benutzer (12);
Empfangen (60) einer vorgeschlagenen Lösung für das Puzzle vom Benutzer (12);
Übertragen (62) einer Anfrage an den Benutzer, die die Identifizierung eines gespeicherten Geheimnisses zum Ziel hat;
Empfangen (64) einer Antwort auf die Anfrage, die die Identifizierung des gespeicherten Geheimnisses zum Ziel hat, vom Benutzer;
Übertragen (66) an den Server, die vorgeschlagene Lösung für das Puzzle und die Antwort auf die Anfrage;
Verwenden (68) des vom Client-Agenten empfangenen Abruf-Tags im Server, um den Verifizierungsblob abzurufen;
Bestimmen, ob die vorgeschlagene Lösung und die Antwort auf die Anfrage korrekt sind; und
wenn die vorgeschlagene Lösung und die Antwort auf die Anfrage korrekt sind, Erlauben von Zugriff auf die auf dem Speichermedium gespeicherten Daten.

2. Verfahren nach Anspruch 1, ferner umfassend, wenn eine Zeitspanne zwischen dem Bereitstellen des Puzzles und dem Empfangen der vorgeschlagenen Lösung größer ist als eine vordefinierte Zeitspanne, nicht Erlauben von Zugriff auf die Daten.

3. Verfahren nach Anspruch 1, ferner umfassend:

vor dem Empfangen (60) der vorgeschlagenen Lösung, Anfordern eines ersten Geheimnisses;
Erhalten des ersten Geheimnisses;
nach Erhalt der Anfrage für Daten, Anfordern eines zweiten Geheimnisses;
Empfangen des zweiten Geheimnisses;
Vergleichen des ersten Geheimnisses mit dem zweiten Geheimnis; und
wenn das erste Geheimnis nicht mit dem zweiten Geheimnis übereinstimmt, nicht Erlauben von Zugriff auf die Daten.

4. Verfahren nach Anspruch 1, wobei:

das Puzzle dazu bestimmt ist, visuell wiedergegeben zu werden; und
eine Lösung für das Puzzle durch Beobachtung der visuellen Darstellung des Puzzles erhalten werden soll.

5

5. Verfahren nach Anspruch 1, wobei:

das Puzzle dazu bestimmt ist, hörbar wiedergegeben zu werden; und
eine Lösung für das Puzzle durch Beobachtung der Audiowiedergabe des Puzzles erhalten werden soll.

10

6. Verfahren nach Anspruch 1, ferner umfassend das Erhalten des Puzzles aus einer Bibliothek von steckbaren Puzzle-Generatoren.

15

7. Verfahren nach Anspruch 1, ferner umfassend:

20

Verschlüsseln der Daten mit einem ersten kryptographischen Schlüssel;
Verschlüsseln des ersten kryptografischen Schlüssels mit einem zweiten kryptografischen Schlüssel;
Speichern der verschlüsselten Daten und des verschlüsselten ersten kryptographischen Schlüssels auf dem Speichermedium; und
wenn die vorgeschlagene Lösung korrekt ist:

25

30

Entschlüsseln des verschlüsselten ersten kryptographischen Schlüssels mit dem zweiten kryptographischen Schlüssel; und
Bereitstellen des entschlüsselten ersten kryptografischen Schlüssels.

35

8. System zur Implementierung computerbezogener Sicherheit unter Einbeziehung menschlicher Partizipation, wobei das System umfasst:

40

einen Ein-/Ausgabebereich, der konfiguriert ist, um eine Anforderung für in einem Speicherabschnitt gespeicherte Daten zu empfangen;
einen ersten Verarbeitungsabschnitt eines Client-Agenten (14), wobei der erste Verarbeitungsabschnitt konfiguriert ist für:

45

als Reaktion auf die Anforderung, Übertragen (48) einer Rätselanforderung und eines Abrufkennzeichens vom Client-Agenten (14) an einen Server (16), wobei das Abrufkennzeichen geeignet ist, das Abrufen eines Verifizierungsblobs zu ermöglichen, der eine verschlüsselte Lösung für ein Puzzle umfasst;
Empfangen (56), beim Client-Agenten (14), des Rätsels von einem Server (16), wobei die menschliche Teilnahme dazu bestimmt

50

55

ist, eine Lösung für das Rätsel und eine korrekte Lösung für das Rätsel zu erhalten;
Übertragen (58) des Puzzles von dem Client-Agenten (14) zu einem Benutzer (12); und
Erhalten, vom Benutzer (12), eines Lösungsvorschlags für das Puzzle;
Übertragen (62) einer Anfrage, die Identifizierung eines gespeicherten Geheimnisses zum Ziel hat, an den Benutzer;
Empfangen (64) einer Antwort auf die Anfrage, die Identifizierung des gespeicherten Geheimnisses zum Ziel hat, vom Benutzer;
Übertragen (66), an den Server, der vorgeschlagenen Lösung für das Puzzle und die Antwort auf die Anfrage;

einen zweiten Verarbeitungsabschnitt des Servers, wobei der zweite Verarbeitungsabschnitt konfiguriert ist, um:

Verwenden (68) des vom Client-Agenten empfangenen Abruf-Tags im Server, um den Verifizierungs-Blob abzurufen;
Feststellen, ob die vorgeschlagene Lösung und die Antwort auf die Anfrage korrekt sind; und
wenn die vorgeschlagene Lösung und die Antwort auf die Anfrage korrekt sind, Erlauben von Zugriff auf die im Speicherabschnitt gespeicherten Daten; und
wobei der Speicherabschnitt konfiguriert ist, um die Daten zu speichern.

9. System nach Anspruch 8, wobei der Verarbeitungsabschnitt ferner konfiguriert ist, um, wenn eine Zeitspanne zwischen dem Bereitstellen des Puzzles und dem Empfangen der vorgeschlagenen Lösung größer als eine vorbestimmte Zeitspanne ist, keinen Zugriff auf die Daten zu erlauben.

10. System nach Anspruch 8, wobei:

der Verarbeitungsabschnitt weiter konfiguriert ist für:

vor dem Erhalten der vorgeschlagene Lösung, Anfordern eines ersten Geheimnisses;
nach Erhalt der Datenanfrage Anfordern eines zweiten Geheimnisses;
Vergleichen des ersten Geheimnisses mit dem zweiten Geheimnis; und
wenn das erste Geheimnis nicht mit dem zweiten Geheimnis übereinstimmt, nicht Erlauben von Zugriff auf die Daten; und

der Ein-/Ausgangsabschnitt ist weiterhin konfiguriert für:

Empfangen des ersten Geheimnisses; und
Empfangen des zweiten Geheimnisses.

11. System nach Anspruch 8, wobei:

das Puzzle dazu bestimmt ist, visuell wiedergegeben zu werden; und
eine Lösung für das Puzzle durch Beobachtung der visuellen Darstellung des Puzzles erhalten werden soll.

12. System nach Anspruch 8, wobei:

das Puzzle dazu bestimmt ist, hörbar wiedergegeben zu werden; und
eine Lösung für das Puzzle soll durch Beobachtung der Audiowiedergabe des Puzzles erhalten werden.

13. System nach Anspruch 8, wobei der Verarbeitungsabschnitt ferner konfiguriert ist, um das Puzzle aus einer Bibliothek von steckbaren Puzzle-Generatoren zu beziehen.

14. System nach Anspruch 8, wobei:

der Verarbeitungsabschnitt weiter konfiguriert ist für:

Verschlüsseln der Daten mit einem ersten kryptographischen Schlüssel;
Verschlüsseln des ersten kryptografischen Schlüssels mit einem zweiten kryptografischen Schlüssel;
Speichern der verschlüsselten Daten und des verschlüsselten ersten kryptografischen Schlüssels im Speicherabschnitt; und
wenn die vorgeschlagene Lösung korrekt ist, Entschlüsseln, mit dem zweiten kryptographischen Schlüssel, des verschlüsselten ersten kryptographischen Schlüssels; und

der Ein-/Ausgabebereich ist ferner konfiguriert, um den entschlüsselten ersten kryptografischen Schlüssel bereitzustellen.

15. Computerlesbares Medium, auf dem computerausführbare Anweisungen zur Durchführung computerbezogener Sicherheit unter Einbeziehung menschlicher Beteiligung gespeichert sind, durch Durchführung des Verfahrens nach einem der Ansprüche 1 bis 7.

Revendications

1. Procédé de sécurité informatique, impliquant une participation humaine, le procédé comprenant les étapes ci-dessous consistant à :

recevoir une demande concernant des données stockées sur un support de stockage ;
en réponse à la demande, transmettre (48), d'un agent client (14) à un serveur (16), une demande de casse-tête et une étiquette de récupération, dans lequel l'étiquette de récupération est pertinente pour permettre la récupération d'un grand objet binaire de vérification comprenant une solution chiffrée à un casse-tête ;
recevoir (56), au niveau de l'agent client (14), en provenance du serveur (16), le casse-tête, dans lequel la participation humaine a pour but d'obtenir une solution au casse-tête, et une solution correcte au casse-tête ;
transmettre (58) le casse-tête, de l'agent client (14) à un utilisateur (12) ;
recevoir (60), en provenance de l'utilisateur (12), une solution proposée au casse-tête ;
transmettre (62) à l'utilisateur, une requête demandant l'identification d'un secret stocké ;
recevoir (64), en provenance de l'utilisateur, une réponse à la requête demandant l'identification du secret stocké ;
transmettre (66), au serveur, la solution proposée au casse-tête et la réponse à la requête ;
utiliser (68), dans le serveur, l'étiquette de récupération reçue en provenance de l'agent client, en vue de récupérer le grand objet binaire de vérification ;
déterminer si la solution proposée et la réponse à la requête sont correctes ; et
si la solution proposée et la réponse à la requête sont correctes, permettre l'accès aux données stockées sur le support de stockage.

2. Procédé selon la revendication 1, comprenant en outre l'étape consistant à, si une durée s'écoulant entre la fourniture du casse-tête et la réception de la solution proposée est supérieure à une durée prédéterminée, ne pas permettre l'accès aux données.

3. Procédé selon la revendication 1, comprenant en outre les étapes ci-dessous consistant à :

avant de recevoir (60) la solution proposée, demander un premier secret ;
recevoir le premier secret ;
subséquemment à la demande de données, demander un second secret ;
recevoir le second secret ;
comparer le premier secret au second secret ; et
si le premier secret ne correspond pas au se-

cond secret, ne pas permettre l'accès aux données.

4. Procédé selon la revendication 1, dans lequel :

le casse-tête est destiné à être rendu visuellement ; et
une solution au casse-tête est destinée à être obtenue par le biais de l'observation du rendu visuel du casse-tête.

5. Procédé selon la revendication 1, dans lequel :

le casse-tête est destiné à être rendu de façon audible ; et
une solution au casse-tête est destinée à être obtenue par le biais de l'observation du rendu audio du casse-tête.

6. Procédé selon la revendication 1, comprenant en outre l'étape consistant à obtenir le casse-tête à partir d'une bibliothèque de générateurs de casse-têtes enfichables.

7. Procédé selon la revendication 1, comprenant en outre les étapes ci-dessous consistant à :

chiffrer les données au moyen d'une première clé cryptographique ;
chiffrer la première clé cryptographique au moyen d'une seconde clé cryptographique ;
stocker, sur le support de stockage, les données chiffrées et la première clé cryptographique chiffrée ; et
si la solution proposée est correcte :

déchiffrer, au moyen de la seconde clé cryptographique, la première clé cryptographique chiffrée ; et
fournir la première clé cryptographique déchiffrée.

8. Système destiné à mettre en oeuvre une sécurité informatique impliquant une participation humaine, le système comprenant :

une partie d'entrée/sortie configurée de manière à recevoir une demande concernant des données stockées dans une partie de mémoire ;
une première partie de traitement d'un agent client (14), la première partie de traitement étant configurée de manière à :

en réponse à la demande, transmettre (48), d'un agent client (14) à un serveur (16), une demande de casse-tête et une étiquette de récupération, dans lequel l'étiquette de récupération est pertinente pour permettre la

récupération d'un grand objet binaire de vérification comprenant une solution chiffrée à un casse-tête ;

recevoir (56), au niveau de l'agent client (14), en provenance du serveur (16), le casse-tête, dans lequel la participation humaine a pour but d'obtenir une solution au casse-tête, et une solution correcte au casse-tête ;
transmettre (58) le casse-tête, de l'agent client (14) à un utilisateur (12) ;
recevoir, en provenance de l'utilisateur (12), une solution proposée au casse-tête ;
transmettre (62) à l'utilisateur, une requête demandant l'identification d'un secret stocké ;
recevoir (64), en provenance de l'utilisateur, une réponse à la requête demandant l'identification du secret stocké ;
transmettre (66), au serveur, la solution proposée au casse-tête et la réponse à la requête ;

une seconde partie de traitement du serveur, la seconde partie de traitement étant configurée de manière à

utiliser (68), dans le serveur, l'étiquette de récupération reçue en provenance de l'agent client, en vue de récupérer le grand objet binaire de vérification ;
déterminer si la solution proposée et la réponse à la requête sont correctes ; et
si la solution proposée et la réponse à la requête sont correctes, permettre l'accès aux données stockées dans la partie de mémoire ; et

dans lequel la partie de mémoire est configurée de manière à stocker les données.

9. Système selon la revendication 8, dans lequel la partie de traitement est en outre configurée de manière à, si une durée s'écoulant entre la fourniture du casse-tête et la réception de la solution proposée est supérieure à une durée prédéterminée, ne pas permettre l'accès aux données.

10. Système selon la revendication 8, dans lequel :

la partie de traitement est en outre configurée de manière à :

avant de recevoir la solution proposée, demander un premier secret ;
subséquemment à la réception de la demande de données, demander un second secret ;

comparer le premier secret au second secret ; et
si le premier secret ne correspond pas au second secret, ne pas permettre l'accès aux données ; et

5

la partie d'entrée/sortie est en outre configurée de manière à :

recevoir le premier secret ; et
recevoir le second secret.

10

11. Système selon la revendication 8, dans lequel :

le casse-tête est destiné à être rendu visuellement ; et
une solution au casse-tête est destinée à être obtenue par le biais de l'observation du rendu visuel du casse-tête.

15

20

12. Système selon la revendication 8, dans lequel :

le casse-tête est destiné à être rendu de façon audible ; et
une solution au casse-tête est destinée à être obtenue par le biais de l'observation du rendu audio du casse-tête.

25

13. Système selon la revendication 8, dans lequel la partie de traitement est en outre configurée de manière à obtenir le casse-tête à partir d'une bibliothèque de générateurs de casse-têtes enfichables.

30

14. Système selon la revendication 8, dans lequel :

35

la partie de traitement est en outre configurée de manière à :

chiffrer les données au moyen d'une première clé cryptographique ;
chiffrer la première clé cryptographique au moyen d'une seconde clé cryptographique ;
stocker, dans la partie de mémoire, les données chiffrées et la première clé cryptographique chiffrée ; et
si la solution proposée est correcte, déchiffrer, au moyen de la seconde clé cryptographique, la première clé cryptographique chiffrée ; et

40

45

50

la partie d'entrée/sortie est en outre configurée de manière à fournir la première clé cryptographique déchiffrée.

55

15. Support lisible par ordinateur sur lequel sont stockées des instructions exécutables par ordinateur pour mettre en oeuvre une sécurité informatique im-

pliant une participation humaine, en mettant en oeuvre le procédé selon l'une quelconque des revendications 1 à 7.

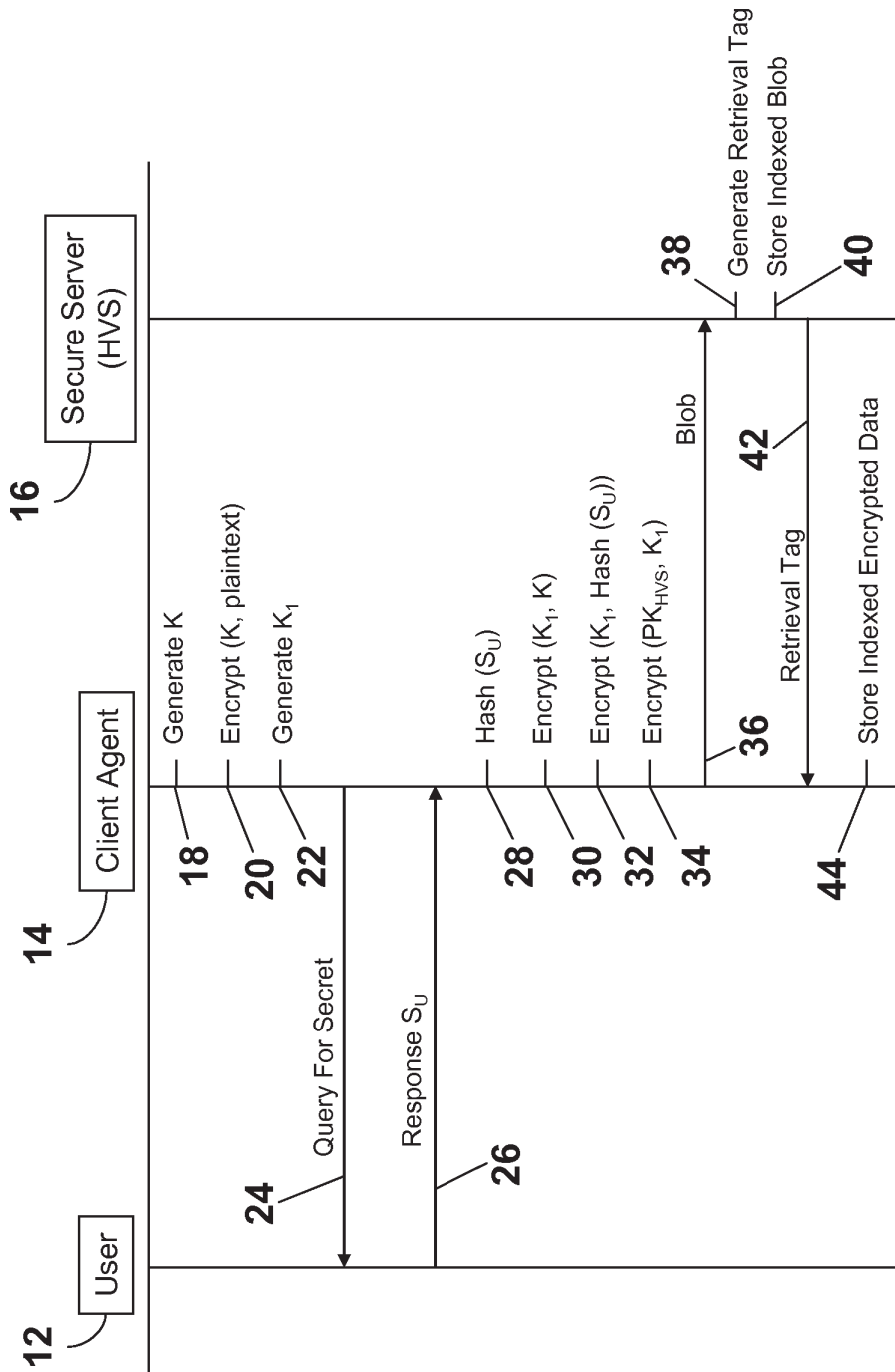


FIGURE 1

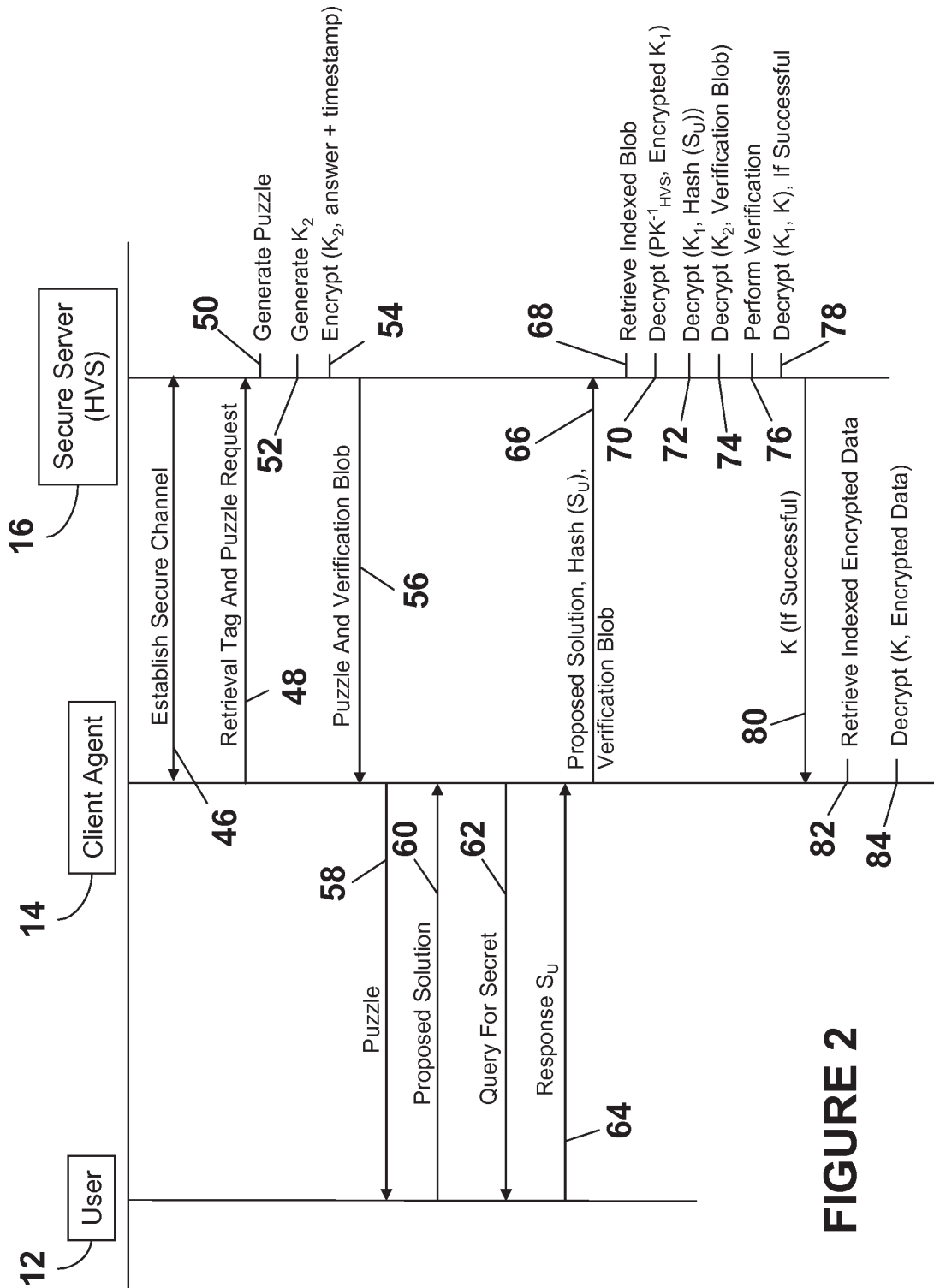


FIGURE 2



FIGURE 3

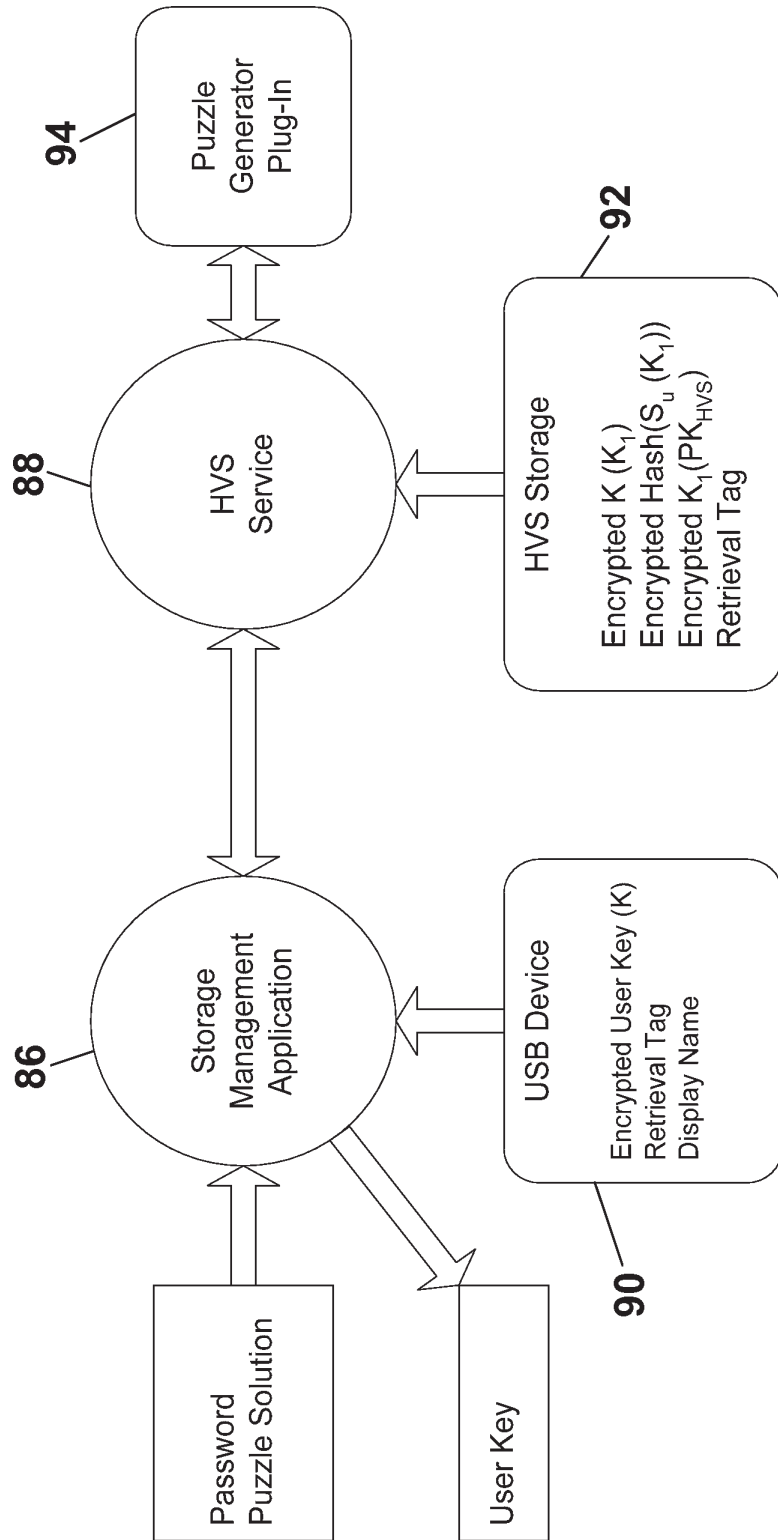


FIGURE 4

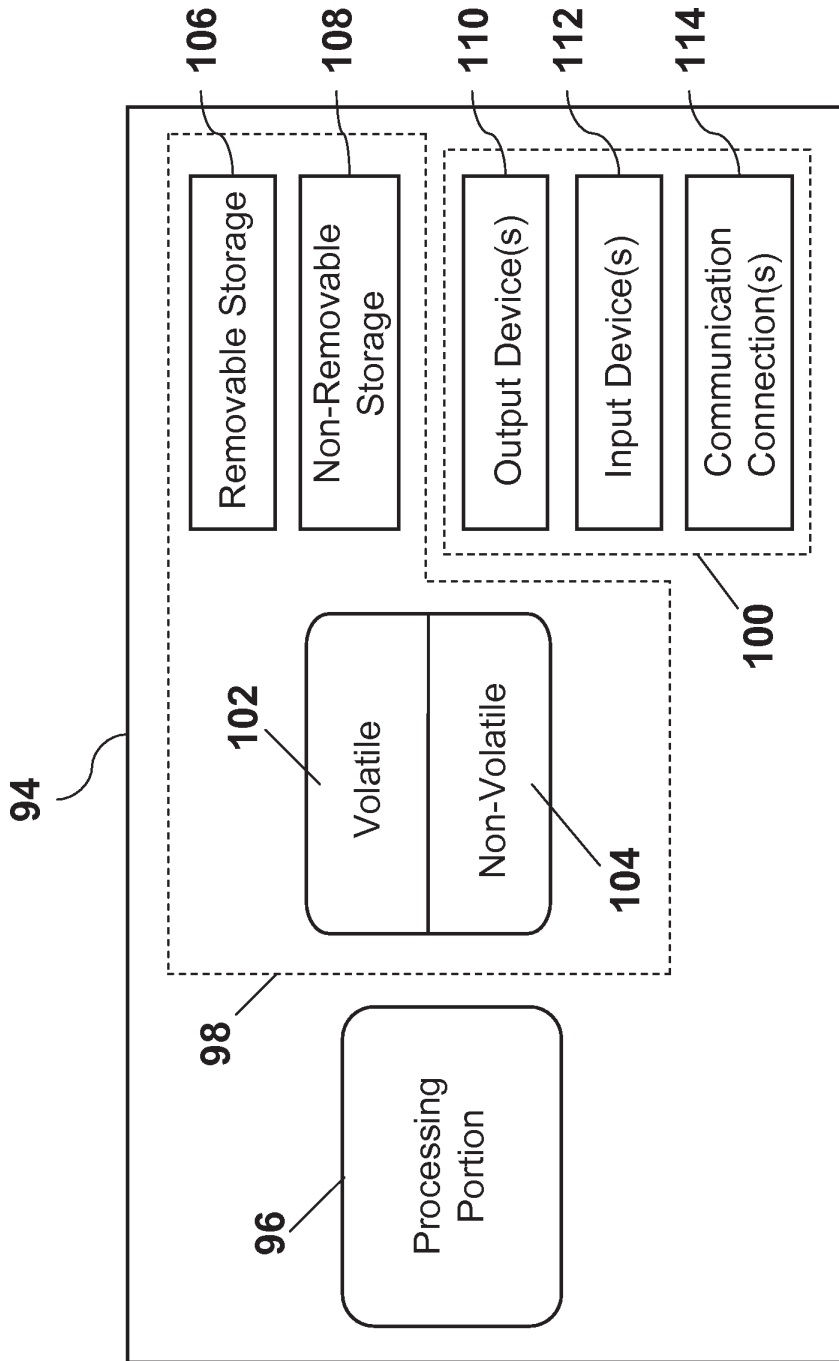


FIGURE 5

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Patent documents cited in the description

- US 5921548 A [0003]
- US 6944765 B1 [0004]
- US 6754827 B1 [0006]

Non-patent literature cited in the description

- **AHN L et al.** CAPTCHA: using hard AI problems for security. *Security in Communication Networks: Third International Conference; Revised Papers/SCN 2002*, 11 September 2002 [0005]
- LECTURE NOTES IN COMPUTER SCIENCE. SPRINGER VERLAG, vol. 2656, 294-311 [0005]

ATTACHMENT 11

Citations of Document 2, Luis von Ahn, Manuel Blum, Nicholas J. Hopper, John Langford. CAPTCHA: Using Hard AI Problems . International Conference on the Theory and Applications of Cryptographic Techniques. EUROCRYPT 2003: Advances in Cryptology. 2004-2008, from Google Scholar https://scholar.google.com/scholar?start=20&hl=en&as_sdt=5,28&sciodt=0,28&as_ylo=2004&as_yhi=2008&cites=11408617951429904733&scipsc=

Abadi, Martin, et al. "Moderately hard, memory-bound functions." *ACM Transactions on Internet Technology (TOIT)* 5.2 (2005): 299-327.

Anderson, Ross. *Security engineering*. John Wiley & Sons, 2008.

Barth, Adam, Collin Jackson, and John C. Mitchell. "Robust defenses for cross-site request forgery." *Proceedings of the 15th ACM conference on Computer and communications security*. 2008.

Bethencourt, John, Jason Franklin, and Mary K. Vernon. "Mapping Internet Sensors with Probe Response Attacks." *USENIX security symposium*. 2005.

Borisov, Nikita. "Computational puzzles as sybil defenses." *Sixth IEEE International Conference on Peer-to-Peer Computing (P2P'06)*. IEEE, 2006.

Casado, Martin, and Michael J. Freedman. "Peering through the shroud: The effect of edge opacity on IP-based client identification." *4th {USENIX} Symposium on Networked Systems Design & Implementation ({NSDI} 07)*. 2007.

Chen, Kuan-Ta, et al. "Identifying MMORPG bots: A traffic analysis approach." *EURASIP Journal on Advances in Signal Processing* 2009.1 (2008): 797159.

Chew, Monica, and J. Doug Tygar. "Image recognition captchas." *International Conference on Information Security*. Springer, Berlin, Heidelberg, 2004.

Crandall, Jedidiah R., et al. "On deriving unknown vulnerabilities from zero-day polymorphic and metamorphic worm exploits." *Proceedings of the 12th ACM conference on Computer and communications security*. 2005.

Elson, Jeremy, et al. "Asirra: a CAPTCHA that exploits interest-aligned manual image categorization." *ACM Conference on Computer and Communications Security*. Vol. 7. 2007.

Enck, William, et al. "Exploiting open functionality in SMS-capable cellular networks." *Proceedings of the 12th ACM conference on Computer and communications security*. 2005.

Feng, Wu-chi, E. Kaiser, and Antoine Luu. "Design and implementation of network puzzles." *Proceedings IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies*. Vol. 4. IEEE, 2005.

Golle, Philippe, and Nicolas Ducheneaut. "Preventing bots from playing online games." *Computers in Entertainment (CIE)* 3.3 (2005): 3-3.

Heymann, Paul, Georgia Koutrika, and Hector Garcia-Molina. "Fighting spam on social web sites: A survey of approaches and future challenges." *IEEE Internet Computing* 11.6 (2007): 36-45.

Kandula, Srikanth, et al. "Botz-4-sale: Surviving organized DDoS attacks that mimic flash crowds." *Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation-Volume 2*. 2005.

Lakshminarayanan, Karthik, et al. "Taming IP packet flooding attacks." *ACM SIGCOMM Computer Communication Review* 34.1 (2004): 45-50.

Lam, Shyong K., and John Riedl. "Shilling recommender systems for fun and profit." *Proceedings of the 13th international conference on World Wide Web*. 2004.

Liu, Bing. *Web data mining: exploring hyperlinks, contents, and usage data*. Springer Science & Business Media, 2007.

Mirkovic, J., Dietrich, S., Dittrich, D., & Reiher, P. (2004). *Internet denial of service: attack and defense mechanisms (Radia Perlman Computer Networking and Security)*. Prentice Hall PTR.

Moy, Gabriel, et al. "Distortion estimation techniques in solving visual CAPTCHAs." *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004.*. Vol. 2. IEEE, 2004.

Nambiar, Arjun, and Matthew Wright. "Salsa: a structured approach to large-scale anonymity." *Proceedings of the 13th ACM conference on Computer and communications security*. 2006.

Pinkas, Binyamin, et al. "Establishing a secure channel with a human user." U.S. Patent No. 7,149,899. 12 Dec. 2006.

Pope, Clark, and Khushpreet Kaur. "Is it human or computer? Defending E-commerce with Captchas." *IT professional* 7.2 (2005): 43-49.

Rui, Yong, and Zicheng Liu. "Artificial: Automated reverse turing test using facial features." *Multimedia Systems* 9.6 (2004): 493-502.

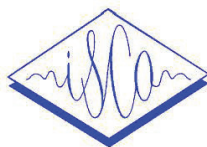
Shyong, K., Dan Frankowski, and John Riedl. "Do you trust your recommendations? An exploration of security and privacy issues in recommender systems." *International Conference on Emerging Trends in Information and Communication Security*. Springer, Berlin, Heidelberg, 2006.

Waters, Brent, et al. "New client puzzle outsourcing techniques for DoS resistance." *Proceedings of the 11th ACM conference on Computer and communications security*. 2004.

Yu, Haifeng, et al. "Sybillimit: A near-optimal social network defense against sybil attacks." *2008 IEEE Symposium on Security and Privacy (sp 2008)*. IEEE, 2008.

Zhang, Yanchao, and Yuguang Fang. "A fine-grained reputation system for reliable service selection in peer-to-peer networks." *IEEE Transactions on Parallel and Distributed Systems* 18.8 (2007): 1134-1145.

ATTACHMENT 12



A REVERSE TURING TEST USING SPEECH

Greg Kochanski, Daniel Lopresti, and Chilin Shih

Bell Laboratories, Lucent Technologies
{gpk,dpl,cls}@research.bell-labs.com, 600 Mountain Ave, Murray Hill, NJ 07974

ABSTRACT

“Hackers” have written malicious programs to exploit on-line services intended for human users. As a result, service providers need a method to tell whether a web site is being accessed by a human or a machine. We expect a parallel scenario as spoken language interfaces become common.

In this paper, we describe a *Reverse Turing Test* (i.e., an algorithm that can distinguish between humans and computers) using speech. We present a test that depends on the fact that human recognition of distorted speech is far more robust than automatic speech recognition techniques.

Our analysis of 18 different sets of distortions demonstrates that there are a variety of ways to make the problem hard for machines. In addition, humans and speech recognition systems make different kinds of mistakes, and this difference can be employed to improve discrimination.

1. INTRODUCTION

The use of the Internet as a means for distributing valuable information and content have also made it an attractive target for “hackers.” Attacks involving malicious programs (“bots”) that attempt to exploit online services intended for human users are already common. These programs consume resources, harass users, make attempts to guess passwords, steal and re-purpose copyrighted content, and invade privacy by reconstructing sensitive data from public views.

As a result, there is a need for automatic methods to tell whether the entity attempting to access a service is a human or a machine. This has come to be known as a *Reverse Turing Test*, or *RTT* (or sometimes a *Human Interactive Proof*). Unlike the test originally proposed by Alan Turing [1], a Reverse Turing Test is administered by a computer, not a human. For the test to be considered effective, humans should be able to pass it with ease, but machines should have a low probability of passing.

While it may seem that passwords and/or biometrics could provide a solution, one must keep in mind that these approaches require pre-registration. An RTT will work even if the user is anonymous and has never used the service before. The RTT problem is fundamentally different from validating a known user. Indeed, a Reverse Turing Test should

be applied before authentication to prevent automated attacks on passwords.

Coates, Baird, and Fateman [2], and von Ahn *et al.* [3] have developed such a test based on a visual perception task. Their ideas are based on the observation that optical character recognition (OCR) systems are not as adept at reading degraded word images as humans are. That RTT is now used commercially to protect a free email service [4].

Speech-based services are proliferating because of their ease-of-use, portability, and potential for hands-free operation. Building a “bot” to navigate a spoken language interface is a tractable problem, especially if there is a fixed sequence of predefined prompts. Hence, we anticipate attacks on such systems and a similar need to prevent machines from abusing speech-based resources intended for human users. Previous work on speech RTTs can be found in [5, 6].

As in the vision RTTs, we exploit the fact that certain pattern recognition tasks are significantly harder for machines than they are for humans. We will use text-to-speech synthesis (TTS) to generate tests, and make use of the limitations of state-of-the-art automatic speech recognition (ASR) technology. (We require only that RTTs cannot be broken cheaply or rapidly. Clearly, any RTT can be broken by hiring a human.)

In this paper, we present the core of a spoken language RTT. We assume a user with a cell phone; the test may consist of having the system speak: “Please enter the following digits on your keypad: ...” followed by a short, random digit string. The speech would be synthesized in a way that ASR is likely to fail the test, e.g., by distorting the signal or adding “difficult” background noise to it after synthesis.

2. PROCEDURES

We designed a set of 18 RTTs based on different distortions of a speech signal (Table 2) to explore a broad range of possibilities. To test our RTTs, we synthesized 200 random 5-digit sequences using the Bell Labs English text-to-speech system [7], with the default male voice. We next distorted the signals and ran the Bell Labs speech recognition system [8] on them, with a grammar that allowed any digit se-

Name	Description	Error Ratio
<i>white</i>	• White Gaussian noise, 4000 Hz bandwidth.	15
<i>buzz</i>	• Sine waves at 700 Hz, 2100 Hz, 3500 Hz.	≥ 20
<i>song</i>	• Bell Labs Song (pop/rock).	> 15
<i>chopin</i>	• Chopin Polonaise for Piano No. 6, Op. 53.	> 20
<i>chant</i>	• Gregorian chant.	> 20
<i>female</i>	• Three overlapping instances of a female voice reading numbers.	> 20
<i>pow</i>	• 10 ms bursts of white Gaussian noise, repeated every 100 ms.	> 20
<i>rnoise</i>	• Every 100 ms, a section of the signal is replaced by white noise of the same RMS amplitude.	> 20
<i>cell</i>	• For each 30 ms window, decide if the data was lost. If so, and previous not lost, duplicate previous. If so, and previous is lost, set to zero. Simulates a bad cell phone channel.	> 10
<i>echo</i>	• Three echoes.	> 20
<i>filter</i>	• A random zero-phase filter.	> 5
<i>distort</i>	• Apply AGC on a 60 ms window, raise to a power, multiply by original amplitude.	≥ 20
<i>mx</i>	• <i>rnoise</i> + <i>chopin</i>	> 20
<i>mb</i>	• <i>song</i> + <i>echo</i>	> 20
<i>mc</i>	• <i>white</i> + <i>pow</i>	≥ 20
<i>md</i>	• <i>female</i> + <i>buzz</i>	≥ 5
<i>me</i>	• <i>rnoise</i> + <i>distort</i>	≥ 20
<i>mf</i>	• <i>filter</i> + <i>distort</i>	3

Table 1. Tested RTTs. The error ratio is an estimate of the largest ratio of the ASR to human utterance error rates. The components of all mixtures were chosen to cause equal error rates for ASR utterance error rates near 85%.

quence.

The recognition results were compared to the original digit sequences using approximate string matching ([9] and references therein) to identify added, missing, and substituted digits. From this we obtained, for each distortion, per-digit and per-utterance error rates, as well as a confusion matrix showing the frequency of each type of error.

A similar procedure was followed to test how well humans could recognize the signals. The authors each listened to 10 randomly-chosen digit sequences for each type of distortion. In these tests, the audio signals were presented twice with a one second pause in between. The subject then typed his/her interpretation and pressed return to listen to the next signal. The tests began at the most severe distortion of each type, and terminated when the subject correctly identified all 10 sequences. One set (*white*) received 67 repetitions per person for a more accurate confusion matrix.

The confusion matrices are scaled to make all diagonal confusion matrices identical. This step is necessary because the distance between raw confusion matrices is generally nonzero, even if there are no errors (*i.e.* the matrices are diagonal). This happens because we do not explicitly balance

the frequencies of each digit, so that one test may see more instances of, say, “3” than another. Hence, we use a scaled confusion matrix, $S = P \cdot C \cdot Q$, where P and Q are diagonal matrices chosen so that the row- and column-sums of S are unity. While deletions and insertions are conventionally placed in the same matrix as substitutions they are actually a fundamentally different kind of error. Specifically, $C_{del,ins}$ is on the diagonal and is missing (zero). Thus, insertions and deletions must be treated differently in the scaling. We make an ad-hoc modification to C before scaling: we set $C_{del,ins} = \sum_{i,j} C_{i,j}/N$ (where $N = 11$ is the number of rows and columns in C), and then set $S_{del,ins} = 0$.

3. ANALYSIS: IS IT POSSIBLE TO DISTINGUISH HUMANS FROM MACHINES?

Human perception of speech in noisy environments is fairly robust. Normal-hearing listeners need a signal-to-noise ratio (SNR) of approximately 1.5 dB to recognize speech [10], while ASR systems require a much more favorable SNR of 5 to 15 dB [11]. Our test results show an even wider gap between human and ASR performance.

Figures 1-4 plot the error rates we measured in four experiments; these correspond to four distinct types of distortion: additive noise, deleting segments of the speech, adding echoes and filtering the signal. The four curves in each chart represent the error rates for ASR on a per-utterance (square) and per-symbol (diamond) basis along with per-utterance (X) and per-symbol (triangle) rates for humans.

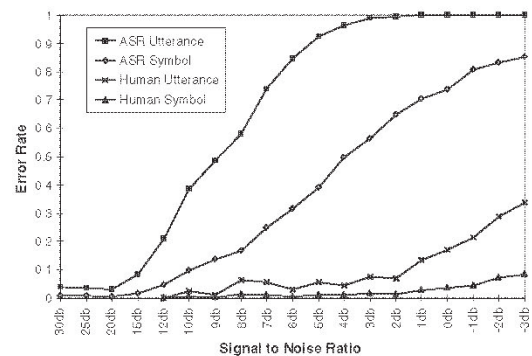


Fig. 1. Results for white noise experiment (*white*).

In figure 1, *white* is simply additive white noise. The x-axis shows SNR, and the y-axis represents the rate of recognition errors. In the ASR curves, each datum represents scores from 200 utterances or 1,000 digits, while in the human curves, each datum represents 201 utterances or 1,005 digits as pooled from the three listeners. ASR performance starts deteriorating when the SNR reaches 15 dB, and breaks down completely by 3 dB. Human performance is only starting to deteriorate at 0 dB SNR.

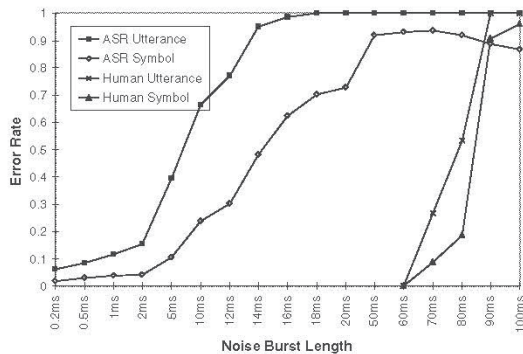


Fig. 2. Results for replacement noise experiment (*noise*).

Figure 2 shows the results for *noise*, which replaces a segment of speech with white noise every 100 ms, as we vary the length of the replaced segment. Human scores in this figure and in Figures 3-4 are based on 30 utterances (150 digits) per datum. As can be seen, ASR starts having problems when 2 ms (2%) of the speech is replaced. Amazingly, human recognition remains perfect at 60 ms (60% replacement), and 80% of the symbols are still correct even when 80% of the speech is missing.

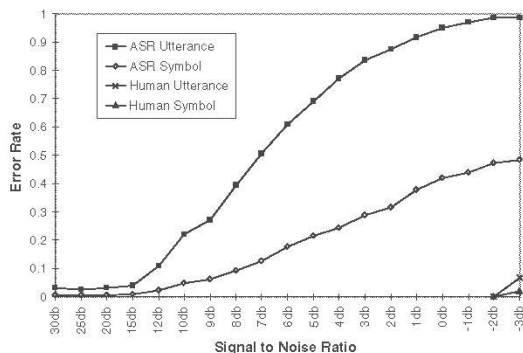


Fig. 3. Results for echo experiment (*echo*).

Our results for *echo*, which adds three echoes to the speech with delays of 60, 132, and 192 ms, are presented in Figure 3. The horizontal axis shows the relative amplitude of the first echo to the speech, while later echoes are 5 dB and 10 dB quieter. ASR performance starts declining when the SNR is 12 dB, while human performance is perfect until -3 dB. This behavior is typical of many of the other tests we performed.

Among our tests, *filter* (Figure 4) showed the smallest difference between ASR and humans. This is a zero-phase frequency domain filter, with a frequency response chosen randomly every 30 Hz. The control parameter sets the standard deviation of the gain, expressed in dB.

Our findings are that the gap between ASR and human

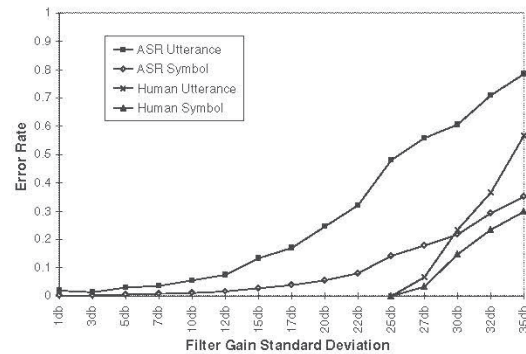


Fig. 4. Results for random filter experiment (*filter*).

performance appears to be wide enough to administer a robust Reverse Turing Test. In general, our tests show that humans can handle noise levels about 15 dB higher than ASR can. Likewise, humans can understand digit strings when more than half of the signal is missing, a point at which the machine already has a 100% error rate for utterances.

4. ANALYSIS: DO HUMANS MAKE THE SAME KINDS OF ERRORS AS MACHINES?

Not only is the error rate larger for an ASR system than for humans, but the pattern of errors is significantly different as well. Figure 5 shows a gray-scale plot of errors made by the machine and humans on the *white* data set. The figures represent noise levels chosen to give matching error rates.

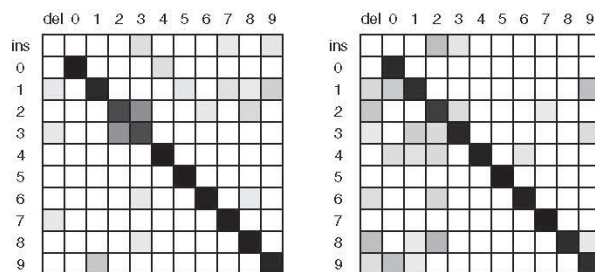


Fig. 5. Scaled confusion matrices at 35% error rate for humans (-3db SNR, left) and machine (+10db SNR, right).

Based on this observation, we could improve an RTT by looking at the kinds of errors that are made. For instance, humans readily confuse the digits “2” and “3” in the presence of white noise, so such an error should not be considered evidence that a machine is attempting to access the system. On the other hand, an “8” is never misunderstood as a “2” by humans but is sometimes by ASR, so such an error would suggest a machine’s presence.

5. ANALYSIS: HOW MANY DISTINCT KINDS OF DISTORTION ARE THERE?

One obvious attack on this kind of RTT is to build a classifier that, working from the input signal, attempts to identify the distortion that was applied, and then sends the input to an ASR system trained specifically for that distortion. ASR can be tuned to work well in the presence of noise [12], but it first needs to be trained with a large representative corpus collected from the environment in question.

Assuming that it is possible to build the necessary classifier, the question then arises “How many different ASR systems would one need to train in order to break the RTT?” To make this question tractable, we approach it by examining the confusion matrices derived from the experiments we have performed; these will serve as a proxy for training and testing a large number of ASR systems.

To do this, we follow logic presented elsewhere in the context of OCR systems [13]. We assume that if the ASR confusion matrices corresponding to two distorted signals are different enough, then separate recognizers will be needed because the two signals are fundamentally different. To make a quantitative comparison, we need to define a distance measure: we use the 2-norm of the difference of the scaled confusion matrices, $D(\alpha, \beta)^2 = \sum_{i,j} (S_{i,j}(\alpha) - S_{i,j}(\beta))^2$, where α and β refer to two different distortions, and S is a function of how the speech is distorted.

We can calibrate our notion of “different enough” by picking an error rate at which the ASR system fails and measuring $D(\alpha, \text{perfect})$ for various types of distortion at that rate. We choose noise levels that yield an utterance error rate of $85\% \pm 6\%$, with a symbol error rate of $34\% \pm 5\%$. The confusion matrices at these levels were found to be distance $D_{85} = 2.1 \pm 0.1$ from the perfect, error-free case. We assume that if the confusion matrices for any two distortions differ by this amount or more, an ASR system trained for one distortion will not be able to function on the other.

In our experiments, the average distance between a pair of distortions (excluding the mixtures) is 2.0 ± 0.9 , quite close to D_{85} , so we would expect that, in general, an ASR system could not be trained to handle two different types of distortions simultaneously. The mixtures tend to be closer to their components, a distance of 1.8 ± 0.6 away, but this is still far enough to conclude that an ASR system trained on either component of the mixture would probably not perform well on the mixture itself. Consequently, it appears there are enough ways of generating fundamentally distinct distortions that the RTTs we have described would resist an attack using any single, well-trained ASR system.

6. CONCLUSIONS

In this paper, we have described our work towards building a speech-based Reverse Turing Test. We show that the gap between ASR and human performance is wide for a variety of noise effects, and that there are opportunities to exploit differences between the patterns of errors that humans and machines make. The RTT is a fundamentally new way of using speech synthesis and recognition technologies.

The authors thank Olivier Siohan for assistance.

7. REFERENCES

- [1] A. Turing, “Computing machinery and intelligence,” *Mind*, vol. 59, no. 236, pp. 433–460, 1950.
- [2] A. L. Coates, H. S. Baird, and R. J. Fatahan, “Pessimistic print: A Reverse Turing Test,” in *Proc. of the Sixth Intl. Conf. on Document Analysis and Recognition*, Seattle, WA, September 2001, pp. 1154–1158.
- [3] L. von Ahn, M. Blum, J. Langford, and U. Manber, “The CAPTCHA project: Telling humans and computers apart (automatically),” October 2001, <http://www.captcha.net/>.
- [4] Yahoo! Inc., March 2002, <http://mail.yahoo.com>.
- [5] D. Lopresti, C. Shih, and G. Kochanski, “Human interactive proofs for spoken language interfaces,” in *Proc. of the Workshop on Human Interactive Proofs*, Palo Alto, CA, January 2002, pp. 30–34.
- [6] N. Chan, “Abstract of sound oriented CAPTCHA,” in *Proc. of the Workshop on Human Interactive Proofs*, Palo Alto, CA, January 2002, p. 35.
- [7] R. W. Sproat, Ed., *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*, Kluwer, 1998.
- [8] Q. Zhou and W. Chou, “An approach to continuous speech recognition based on layered self-adjusting decoding graph,” in *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, Munich, Germany, April 1997, pp. 1779–1782.
- [9] D. Sankoff and B. Kruskal, Joseph, *Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparison*, CSLI Publications, 2 edition, 1999, ISBN 1-57586-217-4. Originally Addison-Wesley, 1993.
- [10] A. Stuart and D. P. Phillips, “Word recognition in continuous and interrupted broadband noise by young normal-hearing, older normal-hearing, and presbycusis listeners,” *Ear & Hearing*, vol. 17, pp. 478–489, 1996.
- [11] E. Woudenberg, F. K. Soong, and J. E. West, “Acoustic echo cancellation for hands-free ASR applications in noise,” in *Proc. of the Workshop on Acoustic Echo and Noise Control*, 1999, pp. 160–163.
- [12] P. J. Moreno, B. Raj, and R. M. Stern, “Data-driven environmental compensation for speech recognition: A unified approach,” *Speech Communication*, vol. 24, pp. 267–285, 1998.
- [13] Y. Li, D. Lopresti, G. Nagy, and A. Tomkins, “Validation of image defect models for optical character recognition,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 2, pp. 99–108, 1996.

ATTACHMENT 13

A REVERSE TURING TEST USING SPEECH

Greg Kochanski, Daniel Lopresti, and Chilin Shih

Bell Laboratories, Lucent Technologies
{gpk,dpl,cls}@research.bell-labs.com, 600 Mountain Ave, Murray Hill, NJ 07974

ABSTRACT

“Hackers” have written malicious programs to exploit on-line services intended for human users. As a result, service providers need a method to tell whether a web site is being accessed by a human or a machine. We expect a parallel scenario as spoken language interfaces become common.

In this paper, we describe a *Reverse Turing Test* (i.e., an algorithm that can distinguish between humans and computers) using speech. We present a test that depends on the fact that human recognition of distorted speech is far more robust than automatic speech recognition techniques.

Our analysis of 18 different sets of distortions demonstrates that there are a variety of ways to make the problem hard for machines. In addition, humans and speech recognition systems make different kinds of mistakes, and this difference can be employed to improve discrimination.

1. INTRODUCTION

The use of the Internet as a means for distributing valuable information and content have also made it an attractive target for “hackers.” Attacks involving malicious programs (“bots”) that attempt to exploit online services intended for human users are already common. These programs consume resources, harass users, make attempts to guess passwords, steal and re-purpose copyrighted content, and invade privacy by reconstructing sensitive data from public views.

As a result, there is a need for automatic methods to tell whether the entity attempting to access a service is a human or a machine. This has come to be known as a *Reverse Turing Test*, or *RTT* (or sometimes a *Human Interactive Proof*). Unlike the test originally proposed by Alan Turing [1], a Reverse Turing Test is administered by a computer, not a human. For the test to be considered effective, humans should be able to pass it with ease, but machines should have a low probability of passing.

While it may seem that passwords and/or biometrics could provide a solution, one must keep in mind that these approaches require pre-registration. An RTT will work even if the user is anonymous and has never used the service before. The RTT problem is fundamentally different from validating a known user. Indeed, a Reverse Turing Test should

be applied before authentication to prevent automated attacks on passwords.

Coates, Baird, and Fateman [2], and von Ahn *et al.* [3] have developed such a test based on a visual perception task. Their ideas are based on the observation that optical character recognition (OCR) systems are not as adept at reading degraded word images as humans are. That RTT is now used commercially to protect a free email service [4].

Speech-based services are proliferating because of their ease-of-use, portability, and potential for hands-free operation. Building a “bot” to navigate a spoken language interface is a tractable problem, especially if there is a fixed sequence of predefined prompts. Hence, we anticipate attacks on such systems and a similar need to prevent machines from abusing speech-based resources intended for human users. Previous work on speech RTTs can be found in [5, 6].

As in the vision RTTs, we exploit the fact that certain pattern recognition tasks are significantly harder for machines than they are for humans. We will use text-to-speech synthesis (TTS) to generate tests, and make use of the limitations of state-of-the-art automatic speech recognition (ASR) technology. (We require only that RTTs cannot be broken cheaply or rapidly. Clearly, any RTT can be broken by hiring a human.)

In this paper, we present the core of a spoken language RTT. We assume a user with a cell phone; the test may consist of having the system speak: “Please enter the following digits on your keypad: ...” followed by a short, random digit string. The speech would be synthesized in a way that ASR is likely to fail the test, e.g., by distorting the signal or adding “difficult” background noise to it after synthesis.

2. PROCEDURES

We designed a set of 18 RTTs based on different distortions of a speech signal (Table 2) to explore a broad range of possibilities. To test our RTTs, we synthesized 200 random 5-digit sequences using the Bell Labs English text-to-speech system [7], with the default male voice. We next distorted the signals and ran the Bell Labs speech recognition system [8] on them, with a grammar that allowed any digit se-

Name	Description	Error Ratio
<i>white</i>	• White Gaussian noise, 4000 Hz bandwidth.	15
<i>buzz</i>	• Sine waves at 700 Hz, 2100 Hz, 3500 Hz.	≥ 20
<i>song</i>	• Bell Labs Song (pop/rock).	> 15
<i>chopin</i>	• Chopin Polonaise for Piano No. 6, Op. 53.	> 20
<i>chant</i>	• Gregorian chant.	> 20
<i>female</i>	• Three overlapping instances of a female voice reading numbers.	> 20
<i>pow</i>	• 10 ms bursts of white Gaussian noise, repeated every 100 ms.	> 20
<i>rnoise</i>	• Every 100 ms, a section of the signal is replaced by white noise of the same RMS amplitude.	> 20
<i>cell</i>	• For each 30 ms window, decide if the data was lost. If so, and previous not lost, duplicate previous. If so, and previous is lost, set to zero. Simulates a bad cell phone channel.	> 10
<i>echo</i>	• Three echoes.	> 20
<i>filter</i>	• A random zero-phase filter.	> 5
<i>distort</i>	• Apply AGC on a 60 ms window, raise to a power, multiply by original amplitude.	≥ 20
<i>mx</i>	• <i>rnoise</i> + <i>chopin</i>	> 20
<i>mx</i>	• <i>song</i> + <i>echo</i>	> 20
<i>mx</i>	• <i>white</i> + <i>pow</i>	≥ 20
<i>mx</i>	• <i>female</i> + <i>buzz</i>	≥ 5
<i>mx</i>	• <i>rnoise</i> + <i>distort</i>	≥ 20
<i>mx</i>	• <i>filter</i> + <i>distort</i>	3

Table 1. Tested RTTs. The error ratio is an estimate of the largest ratio of the ASR to human utterance error rates. The components of all mixtures were chosen to cause equal error rates for ASR utterance error rates near 85%.

quence.

The recognition results were compared to the original digit sequences using approximate string matching ([9] and references therein) to identify added, missing, and substituted digits. From this we obtained, for each distortion, per-digit and per-utterance error rates, as well as a confusion matrix showing the frequency of each type of error.

A similar procedure was followed to test how well humans could recognize the signals. The authors each listened to 10 randomly-chosen digit sequences for each type of distortion. In these tests, the audio signals were presented twice with a one second pause in between. The subject then typed his/her interpretation and pressed return to listen to the next signal. The tests began at the most severe distortion of each type, and terminated when the subject correctly identified all 10 sequences. One set (*white*) received 67 repetitions per person for a more accurate confusion matrix.

The confusion matrices are scaled to make all diagonal confusion matrices identical. This step is necessary because the distance between raw confusion matrices is generally nonzero, even if there are no errors (*i.e.* the matrices are diagonal). This happens because we do not explicitly balance

the frequencies of each digit, so that one test may see more instances of, say, “3” than another. Hence, we use a scaled confusion matrix, $S = P \cdot C \cdot Q$, where P and Q are diagonal matrices chosen so that the row- and column-sums of S are unity. While deletions and insertions are conventionally placed in the same matrix as substitutions they are actually a fundamentally different kind of error. Specifically, $C_{del,ins}$ is on the diagonal and is missing (zero). Thus, insertions and deletions must be treated differently in the scaling. We make an ad-hoc modification to C before scaling: we set $C_{del,ins} \leftarrow \sum_{i,j} C_{i,j} / N$ (where $N = 11$ is the number of rows and columns in C), and then set $S_{del,ins} \leftarrow 0$.

3. ANALYSIS: IS IT POSSIBLE TO DISTINGUISH HUMANS FROM MACHINES?

Human perception of speech in noisy environments is fairly robust. Normal-hearing listeners need a signal-to-noise ratio (SNR) of approximately 1.5 dB to recognize speech [10], while ASR systems require a much more favorable SNR of 5 to 15 dB [11]. Our test results show an even wider gap between human and ASR performance.

Figures 1-4 plot the error rates we measured in four experiments; these correspond to four distinct types of distortion: additive noise, deleting segments of the speech, adding echoes and filtering the signal. The four curves in each chart represent the error rates for ASR on a per-utterance (square) and per-symbol (diamond) basis along with per-utterance (X) and per-symbol (triangle) rates for humans.

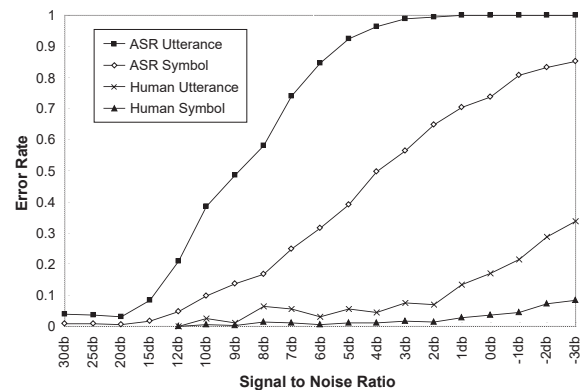


Fig. 1. Results for white noise experiment (*white*).

In figure 1, *white* is simply additive white noise. The x-axis shows SNR, and the y-axis represents the rate of recognition errors. In the ASR curves, each datum represents scores from 200 utterances or 1,000 digits, while in the human curves, each datum represents 201 utterances or 1,005 digits as pooled from the three listeners. ASR performance starts deteriorating when the SNR reaches 15 dB, and breaks down completely by 3 dB. Human performance is only starting to deteriorate at 0 dB SNR.

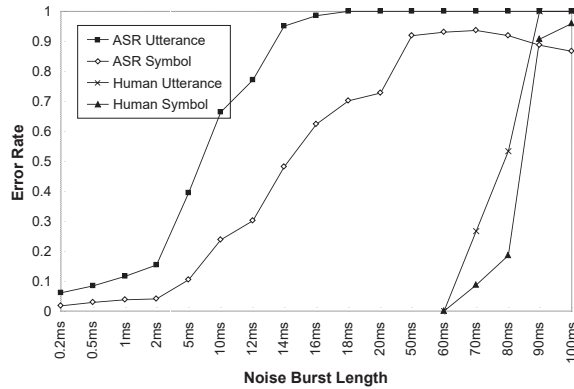


Fig. 2. Results for replacement noise experiment (*rnoise*).

Figure 2 shows the results for *rnoise*, which replaces a segment of speech with white noise every 100 ms, as we vary the length of the replaced segment. Human scores in this figure and in Figures 3-4 are based on 30 utterances (150 digits) per datum. As can be seen, ASR starts having problems when 2 ms (2%) of the speech is replaced. Amazingly, human recognition remains perfect at 60 ms (60% replacement), and 80% of the symbols are still correct even when 80% of the speech is missing.

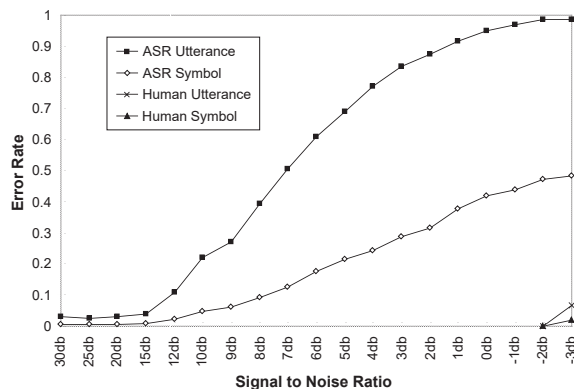


Fig. 3. Results for echo experiment (*echo*).

Our results for *echo*, which adds three echoes to the speech with delays of 60, 132, and 192 ms, are presented in Figure 3. The horizontal axis shows the relative amplitude of the first echo to the speech, while later echoes are 5 dB and 10 dB quieter. ASR performance starts declining when the SNR is 12 dB, while human performance is perfect until -3 dB. This behavior is typical of many of the other tests we performed.

Among our tests, *filter* (Figure 4) showed the smallest difference between ASR and humans. This is a zero-phase frequency domain filter, with a frequency response chosen randomly every 30 Hz. The control parameter sets the standard deviation of the gain, expressed in dB.

Our findings are that the gap between ASR and human

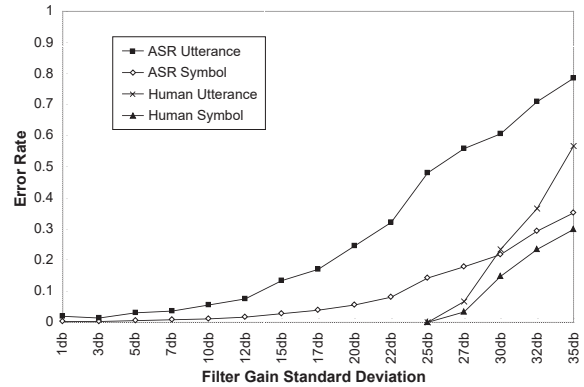


Fig. 4. Results for random filter experiment (*filter*).

performance appears to be wide enough to administer a robust Reverse Turing Test. In general, our tests show that humans can handle noise levels about 15 dB higher than ASR can. Likewise, humans can understand digit strings when more than half of the signal is missing, a point at which the machine already has a 100% error rate for utterances.

4. ANALYSIS: DO HUMANS MAKE THE SAME KINDS OF ERRORS AS MACHINES?

Not only is the error rate larger for an ASR system than for humans, but the pattern of errors is significantly different as well. Figure 5 shows a gray-scale plot of errors made by the machine and humans on the *white* data set. The figures represent noise levels chosen to give matching error rates.

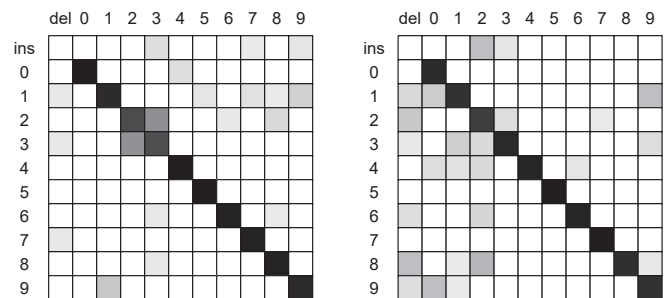


Fig. 5. Scaled confusion matrices at 35% error rate for humans (-3dB SNR, left) and machine (+10dB SNR, right).

Based on this observation, we could improve an RTT by looking at the kinds of errors that are made. For instance, humans readily confuse the digits “2” and “3” in the presence of white noise, so such an error should not be considered evidence that a machine is attempting to access the system. On the other hand, an “8” is never misunderstood as a “2” by humans but is sometimes by ASR, so such an error would suggest a machine's presence.

5. ANALYSIS: HOW MANY DISTINCT KINDS OF DISTORTION ARE THERE?

One obvious attack on this kind of RTT is to build a classifier that, working from the input signal, attempts to identify the distortion that was applied, and then sends the input to an ASR system trained specifically for that distortion. ASR can be tuned to work well in the presence of noise [12], but it first needs to be trained with a large representative corpus collected from the environment in question.

Assuming that it is possible to build the necessary classifier, the question then arises “How many different ASR systems would one need to train in order to break the RTT?” To make this question tractable, we approach it by examining the confusion matrices derived from the experiments we have performed; these will serve as a proxy for training and testing a large number of ASR systems.

To do this, we follow logic presented elsewhere in the context of OCR systems [13]. We assume that if the ASR confusion matrices corresponding to two distorted signals are different enough, then separate recognizers will be needed because the two signals are fundamentally different. To make a quantitative comparison, we need to define a distance measure: we use the 2-norm of the difference of the scaled confusion matrices, $D(\alpha, \beta)^2 = \sum_{i,j} (S_{i,j}(\alpha) - S_{i,j}(\beta))^2$, where α and β refer to two different distortions, and S is a function of how the speech is distorted.

We can calibrate our notion of “different enough” by picking an error rate at which the ASR system fails and measuring $D(\alpha, \text{perfect})$ for various types of distortion at that rate. We choose noise levels that yield an utterance error rate of $85\% \pm 6\%$, with a symbol error rate of $34\% \pm 5\%$. The confusion matrices at these levels were found to be distance $D_{85} = 2.1 \pm 0.1$ from the perfect, error-free case. We assume that if the confusion matrices for any two distortions differ by this amount or more, an ASR system trained for one distortion will not be able to function on the other.

In our experiments, the average distance between a pair of distortions (excluding the mixtures) is 2.0 ± 0.9 , quite close to D_{85} , so we would expect that, in general, an ASR system could not be trained to handle two different types of distortions simultaneously. The mixtures tend to be closer to their components, a distance of 1.8 ± 0.6 away, but this is still far enough to conclude that an ASR system trained on either component of the mixture would probably not perform well on the mixture itself. Consequently, it appears there are enough ways of generating fundamentally distinct distortions that the RTTs we have described would resist an attack using any single, well-trained ASR system.

6. CONCLUSIONS

In this paper, we have described our work towards building a speech-based Reverse Turing Test. We show that the gap between ASR and human performance is wide for a variety of noise effects, and that there are opportunities to exploit differences between the patterns of errors that humans and machines make. The RTT is a fundamentally new way of using speech synthesis and recognition technologies.

The authors thank Olivier Siohan for assistance.

7. REFERENCES

- [1] A. Turing, “Computing machinery and intelligence,” *Mind*, vol. 59, no. 236, pp. 433–460, 1950.
- [2] A. L. Coates, H. S. Baird, and R. J. Fateman, “Pessimistic print: A Reverse Turing Test,” in *Proc. of the Sixth Intl. Conf. on Document Analysis and Recognition*, Seattle, WA, September 2001, pp. 1154–1158.
- [3] L. von Ahn, M. Blum, J. Langford, and U. Manber, “The CAPTCHA project: Telling humans and computers apart (automatically),” October 2001, <http://www.captcha.net/>.
- [4] Yahoo! Inc., “March 2002,” <http://mail.yahoo.com>.
- [5] D. Lopresti, C. Shih, and G. Kochanski, “Human interactive proofs for spoken language interfaces,” in *Proc. of the Workshop on Human Interactive Proofs*, Palo Alto, CA, January 2002, pp. 30–34.
- [6] N. Chan, “Abstract of sound oriented CAPTCHA,” in *Proc. of the Workshop on Human Interactive Proofs*, Palo Alto, CA, January 2002, p. 35.
- [7] R. W. Sproat, Ed., *Multilingual Text-to-Speech Synthesis: The Bell Labs Approach*, Kluwer, 1998.
- [8] Q. Zhou and W. Chou, “An approach to continuous speech recognition based on layered self-adjusting decoding graph,” in *Proc. of the Intl. Conf. on Acoustics, Speech, and Signal Processing*, Munich, Germany, April 1997, pp. 1779–1782.
- [9] D. Sankoff and B. Kruskal, Joseph, *Time Warps, String Edits and Macromolecules: the Theory and Practice of Sequence Comparison*, CSLI Publications, 2 edition, 1999, ISBN 1-57586-217-4. Originally Addison-Wesley, 1993.
- [10] A. Stuart and D. P. Phillips, “Word recognition in continuous and interrupted broadband noise by young normal-hearing, older normal-hearing, and presbycusis listeners,” *Ear & Hearing*, vol. 17, pp. 478–489, 1996.
- [11] E. Woudenbergh, F. K. Soong, and J. E. West, “Acoustic echo cancellation for hands-free ASR applications in noise,” in *Proc. of the Workshop on Acoustic Echo and Noise Control*, 1999, pp. 160–163.
- [12] P. J. Moreno, B. Raj, and R. M. Stern, “Data-driven environmental compensation for speech recognition: A unified approach,” *Speech Communication*, vol. 24, pp. 267–285, 1998.
- [13] Y. Li, D. Lopresti, G. Nagy, and A. Tomkins, “Validation of image defect models for optical character recognition,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 2, pp. 99–108, 1996.

ATTACHMENT 14

ATTACHMENT 15

ome

Tools

CAPTCHA-ATTAC...

CAPTCHA-ATTAC...

☆

📄

🔍

📶

📶

📶

1

/

4

📶

📶

📶

📶

📶

📶

124%

📶

📶

📶

📶

📶

Document Properties

Description

File: CAPTCHA-ATTACHMENT10.pdf

Title: rtt.dvi

Author:

Subject:

Keywords:

Additional Metadata...

Advanced

Created: 4/1/02, 12:25:09 PM

Modified: 9/10/20, 12:27:35 PM

Application: dvipsk 5.58c Copyright 1986, 1994 Radical Eye Software

Advanced

PDF Producer: Acrobat Distiller 3.01 for Windows

PDF Version: 1.6 (Acrobat 7.x)

Location: /Users/Mary/Documents/consult/

File Size: 80,96 KB (82,906 Bytes)

Page Size: 8.50 x 11.00 in

Number of Pages: 4

Tagged PDF: No

Fast Web View: Yes

Help

Cancel

Scan & OCR

Protect

Your current plan is Creative Cloud

Learn More

A REVERSE TURING TEST USING SPEECH

Greg Kochanski, Daniel Lopresti, and Chilin Shih

Bell Laboratories, Lucent Technologies
{gpk,dpl,cls}@research.bell-labs.com, 600 Mountain Ave, Murray Hill, NJ

be applied before authentication to attacks on passwords.

Coates, Baird, and Fateman [2], have developed such a test based on an actor recognition (OCR) systems are not as adept at reading degraded word images as humans are. That RTT is now used commercially to protect a free email service [4].

Speech-based services are proliferating because of their ease-of-use, portability, and potential for hands-free operation. Building a “bot” to navigate a spoken language interface is a tractable problem, especially if there is a fixed sequence of predefined prompts. Hence, we anticipate at-

ATTACHMENT 16

Search 189,434,870 papers from all ...

Search

FAQ

About

Contact

•

Sign In

Create Free Account

Share This Paper

Corpus ID: 63634578

Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP'2002), Denver-Colorado, USA, September 2002

Z. Tan, P. Dalsgaard • Published 2002 • Computer Science

Save To Library

Create Alert

Cite

Launch Research Feed

Top 3 of 9 Citations

View All

Affective Dialogue Management Using Factored POMDPs

Trung H. Bui, J. Zwiers, ... A. Nijholt • Interactive Collaborative Information Systems • 2010

14

Beyond multi-class : structured learning for machine translation

Yizhao Ni • 2010

2

Confidence Measures in Automatic Speech Recognition Systems for Err...

Julia Olcoz, A. Ortega, ... E. Lleida • IberSPEECH • 2014

ABSTRACT

9 CITATIONS

RELATED PAPERS

9 Citations

Q

Fields Of Study

Citation Type

Has PDF

More Filters

More Filters

Filters

Sort by Relevance

Testing the performance of spoken dialogue systems by means of an artificially simulated user

R. López-Cózar, Zoraida Callejas Carrión, M. McTear • Computer Science • 2007

39

Save

Alert

Feed

Affective Dialogue Management Using Factored POMDPs

Trung H. Bui, J. Zwiers, M. Poel, A. Nijholt • Computer Science • 2010

14

Save

Alert

Feed

Confidence Measures in Automatic Speech Recognition Systems for Error Detection in Restricted Domains

Julia Olcoz, A. Ortega, A. Miguel, E. Lleida • Computer Science • 2014

Save

Alert

Feed

Venetan to English machine translation: issues and possible solutions

Suhel Jaber, Sara Tonelli, Rodolfo Delmonte • Computer Science • 2011

“1” Save Alert Feed

Voice conversion by mapping the speaker-specific features using pitch synchronous approach

K. Rao · Computer Science · 2010

67 Save Alert Feed

Czech-Sign Speech Corpus for Semantic Based Machine Translation

J. Kanis, J. Zahradil, Filip Jurčíček, L. Müller · Computer Science · 2006

14 Save Alert Feed

Beyond multi-class : structured learning for machine translation

Yizhao Ni · Computer Science · 2010

2 Save Alert Feed

Exploitation of Machine Learning Techniques in Modelling Phrase Movements for Machine Translation

Yizhao Ni, C. Saunders, S. Szedmák, M. Niranjan • Computer Science • 2011

14 Save Alert Feed

SmartKom: adaptive and flexible multimodal access to multiple applications

Norbert Reithinger, J. Alexandersson, +8 authors Valentin Tschernomas · Computer Science · 2003

78 Save Alert Feed

Related Papers

▼ Show More

2/0

ATTACHMENT 17

A REVERSE TURING TEST USING SPEECH

Greg Kochanski, Daniel Lopresti, and Chilin Shih

Bell Laboratories, Lucent Technologies

{gpk,dpl,cls}@research.bell-labs.com, 600 Mountain Ave, Murray Hill, NJ 07974

ABSTRACT

"Hackers" have written malicious programs to exploit online services intended for human users. As a result, service providers need a method to tell whether a web site is being accessed by a human or a machine. We expect a parallel scenario as spoken language interfaces become common.

In this paper, we describe a *Reverse Turing Test* (i.e., an algorithm that can distinguish between humans and computers) using speech. We present a test that depends on the fact that human recognition of distorted speech is far more robust than automatic speech recognition techniques.

Our analysis of 18 different sets of distortions demonstrates that there are a variety of ways to make the problem hard for machines. In addition, humans and speech recognition systems make different kinds of mistakes, and this dif

Lightshot Screenshot

1 INTRODUCTION

be applied before authentication to prevent automated attacks on passwords.

Coates, Baird, and Fateman [2], and von Ahn *et al.* [3] have developed such a test based on a visual perception task. Their ideas are based on the observation that optical character recognition (OCR) systems are not as adept at reading degraded word images as humans are. That RTT is now used commercially to protect a free email service [4].

Speech-based services are proliferating because of their ease-of-use, portability, and potential for hands-free operation. Building a "bot" to navigate a spoken language interface is a tractable problem, especially if there is a fixed sequence of predefined prompts. Hence, we anticipate attacks on such systems and a similar need to prevent machines from abusing speech-based resources intended for human users. Previous work on speech RTTs can be found in [5, 6].

As in the vision RTTs, we exploit the fact that certain

A-131

ATTACHMENT 18

```

leader 00955cam a2200277 a 4500
001 991000901079704436
008 061020s2005      paau      bm      000 0 eng d
035 ##$a1025961-olcmu_inst
100 1#$aVon Ahn, Luis.
245 10$aHuman Computation / $cLuis von Ahn.
260 ##$aPittsburgh, PA : $b Carnegie Mellon University, $c2005.
300 ##$a87 p. : $bill. ; $c 29cm.
502 ##$aThesis (Ph.D) -- Carnegie Mellon University, 2005.
504 ##$aIncludes bibliographical references (p. 83-87).
596 ##$a9
610 20$aCarnegie Mellon University $xDissertations.
700 1#$aBlum, Manuel, $eadvisor.
710 2$aCarnegie Mellon University. $b Dept. of Computer Science.
902 ##$a1025961
903 ##$aa1025961
916 ##$a20061020
917 ##$a20061020
918 ##$aCATALOGER
919 ##$a20170314
920 ##$aCATALOGER
927 ##$aNone
999 ##$aTHESIS VON,HUMA PH.D $wALPHANUM $c1 $iF097061 $dd10/8/2009 $e9/27/2009 $lBY-REQUEST $mOFFSITE $n1 $rY $sY $tTHESIS $u10/20/2006 FORM=MARC

```

ATTACHMENT 19

ProQuest Dissertations & Theses A&I

Preview Available | Dissertations & Theses

Human computation

von Ahn, Luis.Carnegie Mellon University, ProQuest Dissertations Publishing, 2005. 3205378.


PDF


Order a copy


Cite


Email


All Options

Abstract

Tasks like image recognition are trivial for humans, but continue to challenge even the most sophisticated computer programs. This thesis introduces a paradigm for utilizing human processing power to solve problems that computers cannot yet solve. Traditional approaches to solving such problems focus on improving software. I advocate a novel approach: constructively channel human brainpower using computer games. For example, the ESP Game, introduced in this thesis, is an enjoyable online game—many people play over 40 hours a week—and when people play, they help label images on the Web with descriptive keywords. These keywords can be used to significantly improve the accuracy of image search. People play the game not because they want to help, but because they enjoy it.

I introduce three other examples of "games with a purpose": Peekaboom, which helps determine the location

More

Details

Subject	Computer science; Artificial intelligence
Classification	0984: Computer science 0800: Artificial intelligence
Identifier / keyword	Applied sciences; CAPTCHA; ESP game; Human computation; Peekaboom
Title	Human computation
Author	von Ahn, Luis
Number of pages	87
Publication year	2005
	2005

Degree date	
School code	0041
Source	DAI-B 67/02, Dissertation Abstracts International
Place of publication	Ann Arbor
Country of publication	United States
ISBN	978-0-542-54148-3
Advisor	Blum, Manuel
University/institution	Carnegie Mellon University
University location	United States -- Pennsylvania
Degree	Ph.D.
Source type	Dissertations & Theses
Language	English
Document type	Dissertation/Thesis
Dissertation/thesis number	3205378
ProQuest document ID	305006687
Document URL	http://libproxy.unl.edu/login?url=https://www-proquest-com.libproxy.unl.edu/docview/305006687?accountid=8116
Copyright	Database copyright ProQuest LLC; ProQuest does not claim copyright in the individual underlying works.
Database	ProQuest Dissertations & Theses A&I



Ask a Question

The University of Nebraska-Lincoln

