Near-Field Communication-Based Secure Mobile Payment Service

Kiran S. Kadambi[1]
Hewlett-Packard Company
19447 Pruneridge Avenue
Cupertino, CA 95014
972-693-4062
kiran.kadambi@gmail.com

Jun Li
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304
650-857-4087
jun.li@hp.com

Alan H. Karp
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304
650-857-3967
alan.karp@hp.com

ABSTRACT
The loss of customer account information makes headlines all too often. The Payment Card Industry (PCI) Data Security Standard (DSS) has generated resistance from merchants because of its cost and complexity. Unfortunately, innovating in the Point of Sale (PoS) space is hard because the protocols have been standardized, tightly constraining new ideas. Using mobile phones as payment devices opens up some opportunities, but any solution must minimize changes in the back-end processing. This paper describes a Near Field Communication (NFC) based mobile phone payment solution in a service-oriented environment, which provides the needed data protection without requiring costly changes to the payment processing infrastructure. In the payment transaction path, a personal mobile phone is viewed as a single, user-trusted touch point. Compared to other solutions available, our approach better protects user credentials, provides better user control over the transaction, and supports both proximity and remote transactions. The prototype has been integrated with an in-store kiosk application and a HP multi-channel banking platform to demonstrate its value in the retail environment.

Categories and Subject Descriptors
K.4.4 [**Computers and Society**]: Electronic Commerce–Security
H. 3.5 [**Information Storage and Retrieval**]: Online Information Services–Web-based Services

General Terms
Management, Design, Security, Human Factors

Keywords
NFC, mobile payment, payment authorization, SOA

1. INTRODUCTION
With the increase in mobile ownership and cheaper data services, the mobile commerce market was expected to generate close to $55 billion in revenue in 2008, with significant growth predictions for the future [3]. The growth of the mobile banking channel provides opportunities for various players in this emerging market, including mobile phone manufacturers and carriers, banking institutions, and service providers. For example, this channel introduces a new revenue stream for banks from mobile micro-payments and reduces customer churn due to the convenience provided by mobile payments [churn reduction of 10% was reported in [3]). To the service providers in retail, online music download, public transportation, *etc*., it offers new, more convenient touch points to the customers and enhanced security features compared to traditional plastic credit or debit cards. This new channel provides customers with a single point of spending once personal mobile devices become wallet replacements.

The loss of customer account information makes headlines all too often [6, 7]. To protect cardholder data, merchants and service providers are required to comply with the Payment Card Industry (PCI) Data Security Standard (DSS) [24]. Merchants, or their service providers (such as payment gateways), are subject to the PCI DSS compliance if they store, collect and transmit cardholder data on their systems or the systems of their service providers. To meet compliance requirements, merchants and their service providers often have to implement costly upgrades to reinforce systems that are already complex enough. This also puts small and medium sized merchants and e-commerce startups at a competitive disadvantage.

To explore and develop this mobile banking channel and its integration with systems from other players, such as network carriers and retailers, while at the same time striving for a solution that provides needed data protection for cardholder data, we[1] started with a contact-less mobile payment scheme based on Near-Field Communication (NFC) and integrated it with a retailer-based service framework prototyped within HP. An NFC-enabled phone is a cell phone with a smartcard microchip embedded that can communicate with a local host computer within a small range (<10 cm) over a RF channel (13.6 MHz). The NFC-based microchips have two key security features: Public Key Infrastructure (PKI) based cryptographic functionalities, and a well-controlled, point-to-point, short-range communication channel. The commercially available NFC-based chips follow the widely adopted EMV [9], a standard that defines the interaction

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, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
ICEC'09, August 12–15, 2009, Taipei, Taiwan.
Copyright © 2009 ACM 978-1-60558-586-4/09/08…$10.00.

[1] Kiran Kadambi's present address: Darden School of Business, University of Virginia.

SAMSUNG 1019

between IC cards (i.e., microchip cards) and IC card processing devices, such as Point-of-Sale (POS) terminals and ATMs, to authenticate credit and debit card payment transactions.

Compared to other mobile payment services and systems, the NFC-based mobile payment end-to-end solution that we developed exhibits the following core technical innovations:

- **End-To-End Secure Transaction Protocol:** Other solutions rely on the mobile phone to communicate the user's payment information, which includes the payment amount and credit card number, to the transaction issuer device (*e.g.*, an in-store self-service terminal) over a point-to-point secure communication channel. The merchant may want to hold such sensitive information for various reasons, even long after the transaction occurs [11]. In our protocol, the phone issues a payment authorization token with payment information such as credit card number completely opaque to the merchant. The backend transaction is performed after the token is validated. Thus, our protocol mitigates the customer data breach risk at the merchant and significantly reduces the effort to (if not completely removes) the PCI DSS compliance requirement, because PCI DSS does not apply if primary account number (card number) are not stored, processed, or transmitted [24, 25]. Furthermore, because the authorization token is linked to the transaction identifier at both the merchant and the bank, the merchant is able to defend itself in a dispute.
- **Personal Cell Phone as the Single User-Trusted Touch Point:** As a personal trusted device equipped with sufficient display capabilities, the mobile phone is used to present and interpret the payment amount, confirm the payment, and issue the payment-specific authorization token. No personal information ever gets revealed to the transaction issuer owned by the merchant, which is occasionally open to compromise as reported in [6], and thus not fully trusted by the customer.
- **Support of Proximity Transaction and Remote Transaction:** The existing NFC payment systems are designed to serve only transaction requests issued from a nearby device. Our solution supports online purchases by having downloadable software installed to a local computer, which mediates the interaction between the mobile phone and the remote web application. As a result, aside from an in-store service terminal, our solution can use the same secure transaction protocol from a home shopping station such as a PC desktop or special home appliance like the HP TouchSmart, with the personal mobile phone being the transaction authorization token issuer.
- **Flexible and Multiple Payment Options:** Due to the banking infrastructure's multi-channel support, mobile payments can be routed directly to an account provided by the bank, or to a digital wallet offered by the phone carrier that maintains shadow balances to the chosen banking account [17]. Thus, the mobile phone can become a single point of spending.

The rest of the paper is structured as follows. In Section 2, we introduce the PCI DSS standard to help understand merchant specific compliance requirements for data protection. In Section 3, we then describe the two solutions systems that we developed in HP, namely Adaptive Bank and Retail Store Assistant (RSA), upon which our mobile payment system was built. We then detail the NFC based mobile payment solution architecture in Section 4, and have Sections 5, 6, 7 cover the core technical aspects, namely, the secure transaction protocol, the interaction between the phone and the local machine, and the interaction between the phone and the remote service mediated by a local machine, respectively. We report our prototype implementation in Section 8, which details the hardware we used for our prototype, the SDK that we used for development, and how we managed to use different middleware to integrate the mobile phone, the retail solution based on the RSA kiosk, and the Adaptive Bank to form a complete end-to-end payment system prototype. In Section 9 we contrast our mobile payment solution with existing mobile payment solutions. The paper is concluded in Section 10.

2. PCI DSS

The Payment Card Industry (PCI) Data Security Standard (DSS) [24] specifies regulatory compliance requirements for merchants (retail store, mail order, e-commerce sites, *etc*.) and service providers (payment gateways, e-commerce host providers, credit report agencies, *etc*.) that involve payment processing, and thus store, process, or transmit cardholder information. The Point-of-Sale (POS) terminal environment may be excluded from merchant's compliance validation, if it is not IP network based or has no access to the merchant location [25]. Merchants are subject to different levels of compliance, from a high of level 1 to a low of level 4, depending on annual payment transaction volumes (level 1 for more than 6 million credit card transactions per year).

Information subject to protection consists of *cardholder data* that further includes Primary Account Number (PAN), cardholder name, service code and expiration Date, and sensitive *authentication data* that further includes full magnetic stripe data, validation code, and PIN. Following the defined storage and protection requirements, cardholder data is allowed to be stored on some storage medium with sufficient measures to ensure the protection of the PANs. Other cardholder data mentioned above need only be protected if it is stored in conjunction with the PANs. The sensitive authentication data mentioned above must not be stored at all.

The PCI DSS is divided into six categories, with twelve high-level requirements that govern network security, data protection, access control, access monitoring, *etc*. A template of assessment procedures is described in [25]. The high-level requirement on cardholder data protection is further refined into: (1) keep cardholder data storage to a minimum, with the necessary data retention and disposal policies; (2) no storing of sensitive authentication data after payment authorization (even if encrypted); (3) mask PANs when displayed; (4) render PAN unreadable anywhere it is stored, including truncation, encryption, *etc*. (5) protect the cryptographic keys used for encryption of cardholder data; (6) document and implement key-management processes and procedures for cryptographic keys used for encryption of cardholder data.

Note that the standard [25] states that *"…. PCI DSS, however, does not apply if PANs are not stored, processed, and transmitted"*. Clearly, the data protection requirement will be effectively satisfied with minimum risk if cardholder data is not stored at all. We interpret the statement to mean that the

requirement is satisfied if the data is stored in a form that the merchant cannot interpret, which in our solution is achieved by applying strong public-key based crypto support to encrypt card holder data into payment authorization tokens that are transmitted across the end-to-end payment system.

3. RETAIL STORE ASSISTANT & ADAPTIVE BANK

Our mobile payment system is built on two Service Oriented Architecture (SOA) based solution systems: the Retail Store Assistant (RSA) that delivers personalized shopping services and experiences to in-store customers, and the Adaptive Bank platform that supports multiple front-end banking channels to the core banking process. Both systems are interoperable at the Web Service SOAP messaging level. Our mobile payment system uses the RSA as a front-end customer touch point and the Adaptive Bank for backend payment processing.

3.1 Retail Store Assistant (RSA)

The RSA system is designed to uniformly support shopping-related kiosk applications at different touch points (in physical stores, at home, and through mobile devices). The kiosk-based application prototype is currently accessible for online demonstration [27]. These touch points serve as the front-end to a network of services provided by the retailer and its business partners, which include manufacturers and financial institutions. The system aims to reduce or eliminate common shopping frustrations and to help buyers with shopping rather than only helping retailers sell. The RSA application prototype currently provides in-store personalized coupon offering, product information search, personal event management, voice and image based shopping reminders, *etc*. The mobile version of the RSA system has also been prototyped [16].

NFC mobile payment is a key part of the integrated solution to improve the shopping experience, not only because it is convenient for customer payment transactions, but also because it naturally pulls other shopping and purchase related applications together onto the personalized mobile phones, primarily due to its secure wireless communication protocol and the "tap-and-go" simple phone usage model. Examples of these applications can include the storing of accepted personalized coupons and subsequently redeeming the coupons at check out, the on-the-phone storing of transaction receipts to keep track of shopping purchases, *etc*.

3.2 Adaptive Bank

Adaptive Bank is a SOA-based, multi-channel integration framework that was developed within HP to integrate front office applications with the backend business services and systems by a uniform connectivity and messaging interface [1].

The infrastructure framework of Adaptive Bank is shown in Figure 1. It supports multiple channel applications that include teller, call center, ATM, web, *etc*. Each channel application is associated with a different communication mechanism (TCP/IP, HTTP and Web-Services, *etc*.) and message formats that includes Fixed Length, XML and IFX (following the Interactive Financial eXchange standard). Adaptive Bank exposes a uniform channel messaging and connectivity interface to integrate different channel applications with the backend infrastructure. The business events generated from all these different channels are

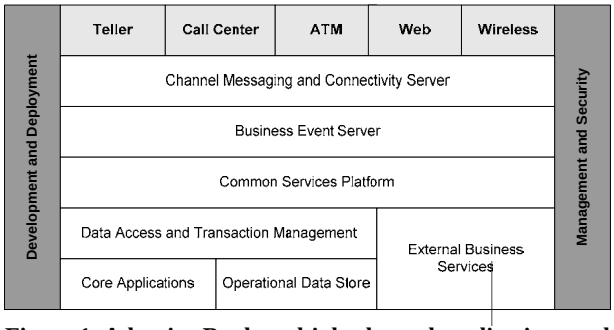


Figure 1. Adaptive Bank multiple channel applications and its layered infrastructure.

routed through the business event server for further backend processing. These events are also collected for business activity monitoring, which can be either a common service from the common services platform or provided by external business services.

Seamless customer experiences are offered over multiple banking channels. Specifically, the operational data store hosts the data model and integration technology essential to creating a unified customer view across the channels by integrating various data sources that span the infrastructure stack, such as host customer transactions, customer profile, customer activity monitoring, and targeted customer offerings. Each channel application can fully leverage comprehensive customer behavior information aggregated from all different channels.

4. MOBILE PAYMENT SOLUTION ARCHITECTURE

The entire secure mobile payment solution spans the NFC-enabled cell phone, the service terminal (such as an in-store kiosk enabled by the RSA) with its backend services, and the Adaptive Bank platform. The overall system is shown in Figure 2.

We introduced a mobile banking channel to Adaptive Bank. The Adaptive Bank provides front-end processing of the *issuing* bank (*i.e.*, the bank that issues the payment card to the card holders, who then use the issued card at the merchant [26]). In our system, the issuing bank owns the mobile user account. This channel receives the transaction request from the *acquiring* bank (*i.e.*, the bank that takes care of payment processing for the merchant [26]), processes the request, and forwards the processed result to the core banking processor of the issuing bank, where the credit/debit processing updates the mobile user accounts to reflect payment transaction. The merchant account can be either immediately credited in this credit/debit processing, or deferred to later bank-to-bank reconciliation with the settlement bank involved. The payment transaction request originates at the local transaction issuer, which can be a kiosk or a point-of-sale terminal at a retail store.

The transaction request message over this mobile banking channel carries two parts: the *payment information* and the *authorization token* issued from the mobile phone. The payment information can include, for example, the merchant bank account number, the payment amount to be collected from the customer's account, and a transaction identifier issued by the merchant for future tracking purposes, *e.g.*, when the purchased items are returned and a

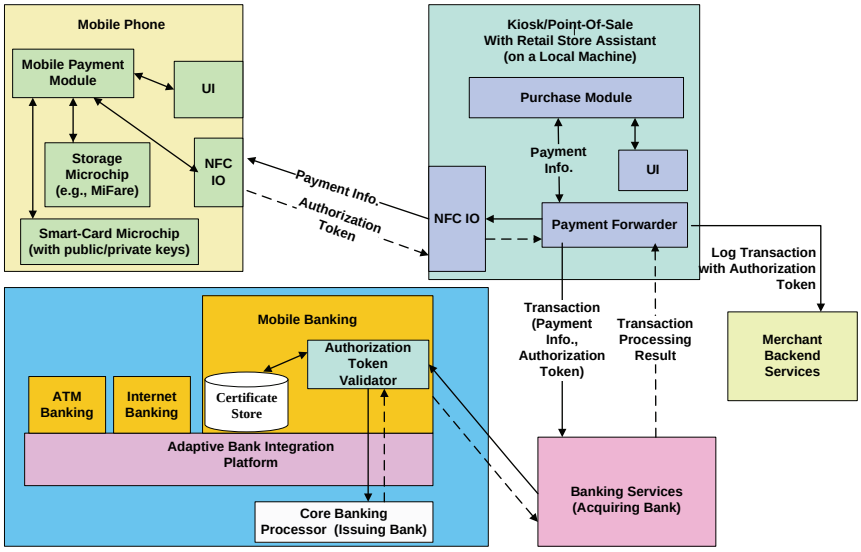


Figure 2. The overall NFC-enabled mobile payment system.

refund is issued. The authorization token is constructed by the mobile phone based on the payment information sent from the local transaction issuer (detailed in the description of the secure transaction protocol in Section 5), and is tamper-resistant with strong public-key based cryptographic support built into the mobile phone's smart-card microchips.

While our payment message conforms to the traditional payment processing protocol, our proposal replaces the sensitive information, such as bank account number (PAN) and password (PIN), with an authorization token.

The processing of the transaction request at the mobile banking channel involves the validation of the authorization token (the validation procedure will be detailed in Section 5). The credit/debit transaction relies on a validated authorization token to ensure the correct customer account and actual payment amount specified by the mobile user.

Each purchase transaction and the associated authorization token are logged by the local transaction issuer and the logs are stored at the merchant's backend services. The transaction can be either a purchase transaction or a refund transaction. For a refund transaction, the authorization token is not issued from the mobile phone. Instead, the merchant relies on the recorded transaction identifier to retrieve the authorization token issued for the previous purchase transaction. Afterwards, the retrieved authorization is incorporated into the refund transaction request. A transaction data flow similar to that shown in Figure 2 happens and ultimately the issuing bank credits the customer the refund amount, and the acquiring bank deducts the corresponding amount from the merchant account.

In a typical merchant-based deployment environment, a dedicated secure communication payment network is established between the merchant's transaction issuer and Adaptive Bank to handle actual payment transactions. For example, the merchant has its local transaction issuer (such as a POS terminal) connect to the acquiring bank. The acquiring bank then forward the transaction request to the payment network that it belongs to, such as VisaNet or BankNet, for transaction routing. The transaction request eventually reaches the Adaptive Bank, because the issuing bank belongs to the same payment network, or because the two payment networks that the issuing bank and the acquiring bank are associated with can be bridged through a payment processing gateway [26].

In other deployment environments that are more open, such as a home-based remote transaction (detailed later in Section 7), a secure communication channel like HTTPS is required between the local machine and the merchant's backend with the above mentioned dedicated communication channel.

The merchant's acquiring bank serves as only a middleman in the banking network to forward mobile payment transaction requests. Its functionalities can be logically encapsulated by some merchant's backend services, as shown in Figure 2. In the remainder of this paper, we are only interested in the issuing bank, at which the Adaptive Bank is hosted. We refer to the issuing bank as simply the *bank*.

Note that the end-to-end payment processing network described in Figure 2 involves only the NFC communication channel and the traditional payment processing network. Thus no mobile carrier networks are required in our mobile payment solution.

5. SECURE TRANSACTION PROTOCOL

5.1 Payment Authorization Token

The smart-card chip embedded in the phone holds a public/private key pair assigned to the user, as well as the public key of the user's bank. The local transaction issuer sends the required payment amount to the user's device via the NFC channel. After the user confirms the payment amount, a digital signature is created from the user's private key and the user-side payment information that includes the user's bank account number, the payment amount, a transaction counter (to prevent replay attacks), and a time stamp. The payment standards require that all

transactions have to be time stamped to the second with GMT offsets.

The digital signature and the user-side payment information are then encrypted with the bank's public key. The encrypted result can not be interpreted by the merchant, and thus we need a clear-text part to specify: (1) the payment amount that the mobile user promises, such that the merchant can know beforehand the amount that the user claims to be willing to pay and (2) the bank's routing number, because the payment network needs the number to route the transaction request to the correct bank.

The overall data structure that includes both the encrypted part and the clear-text part is called an *authorization token*, as shown in Figure 3.

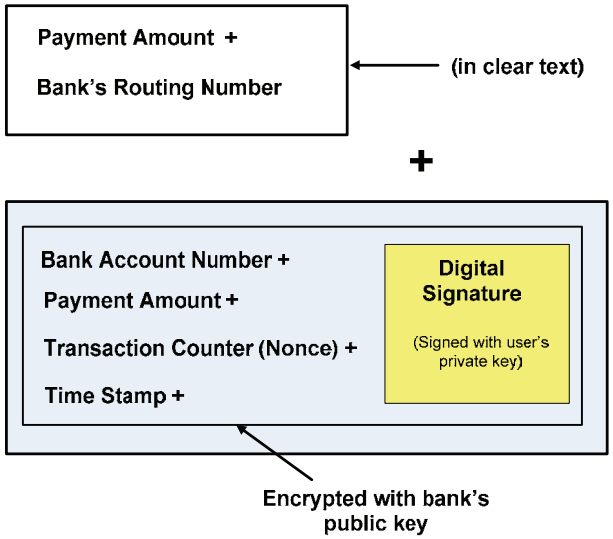


Figure 3. Payment authorization token issued from mobile phones.

Note that there are two payment amounts encapsulated in the authorization token: the amount in clear-text, which the merchant can check, and the amount in encrypted form, which is the true amount that the mobile user agrees to pay. The Adaptive Bank's mobile banking channel will detect if the two payment amounts differ, perhaps because the merchant deliberately or inadvertently changed the clear-text part, and will refuse to forward the transaction request down to the issuing bank's core banking processor. Similarly, if the customer lies to the merchant by deliberately setting the amount in clear-text larger that what is put in the encrypted form, although the merchant will proceed to issue the transaction request to the backend payment network, the merchant will be notified about transaction failure by the Adaptive Bank's mobile banking channel. Consequently, the merchant will not release what is to be purchased by the customer. Thus the customer gains nothing from such cheating.

It is up to the merchant to protect the integrity of the clear text message part and the authorization token in the same way that the merchant would have to protect a check made out to "Cash". If the communications channel is secure, there is no risk that the token will be stolen. If not, it is as likely that attackers can subvert the key exchange process as it is that they will steal the token. The mobile users never expose their identities, or even their public keys, to the merchant. As a result, we do not have the mobile phone sign the entire authorization token, since the merchant is not able to verify this signature.

5.2 Detailed Communication Protocol

The authorization token is issued by the mobile phone to the transaction requester from the local transaction issuer over the NFC channel. The local transaction issuer inspects the clear text part to check the amount the user agrees to pay. After checking, the authorization token is incorporated into the transaction request message by the local transaction issuer for backend payment processing. This token finally arrives at the Adaptive Bank's mobile banking channel through necessary routing network, guided by the bank's routing number embedded in the transaction request. The authorization token is then decrypted using the bank's private key. The digital signature that is included in the decrypted authorization token is checked, after the mobile banking channel using the decrypted bank account number to retrieve the corresponding customer's public key from the bank's central certificate store. After the valid digital signature is confirmed by the mobile banking channel, at the bank's core banking processor, the user account is debited the specified amount. For the core banking processor to make the final decision, other bank account information is evaluated as well, such as whether the account is blocked or frozen, whether the card is stolen, and whether the transaction is over the limit. The end-to-end payment transaction result eventually returns to the transaction issuer at the local machine right next to the mobile phone.

The end-to-end secure transaction spanning the mobile phone, the merchant, and Adaptive Bank, is illustrated by the sequence diagram in Figure 4.

All customer personal information is in the encrypted part of the authorization token, which is opaque to the merchant. Thus, with our secure payment solution, customer related data breach incidents, such as the one at T.J. Maxx [6], will not compromise customer account numbers. Since the authorization token is not designated to a particular merchant, the merchant that receives the token has an incentive to protect it from being tampered with, lost or stolen. In the case of a return or refund, the merchant can retrieve from its backend office the original authorization token and provide the retrieved token to the bank, along with the refund amount that the merchant is to pay. The bank decrypts the authorization token, validates the authorization token, extracts the embedded user's account number, and credits the refund amount to the user from the merchant's account.

With respect to the PCI DSS regulatory compliance described in Section 2, because the authorization tokens that come out of the mobile phone are encrypted with the bank's public key, the merchants can never obtain sensitive card holder information, in particular, the primary account number (PAN) embedded in the authorization token. Therefore, with the NFC mobile banking channel, the merchants should be relieved from the compliance requirements imposed by the PCI DSS.

5.3 Key Management

The user's public/private key pair is generated locally within the mobile phone during the initial mobile application installation,

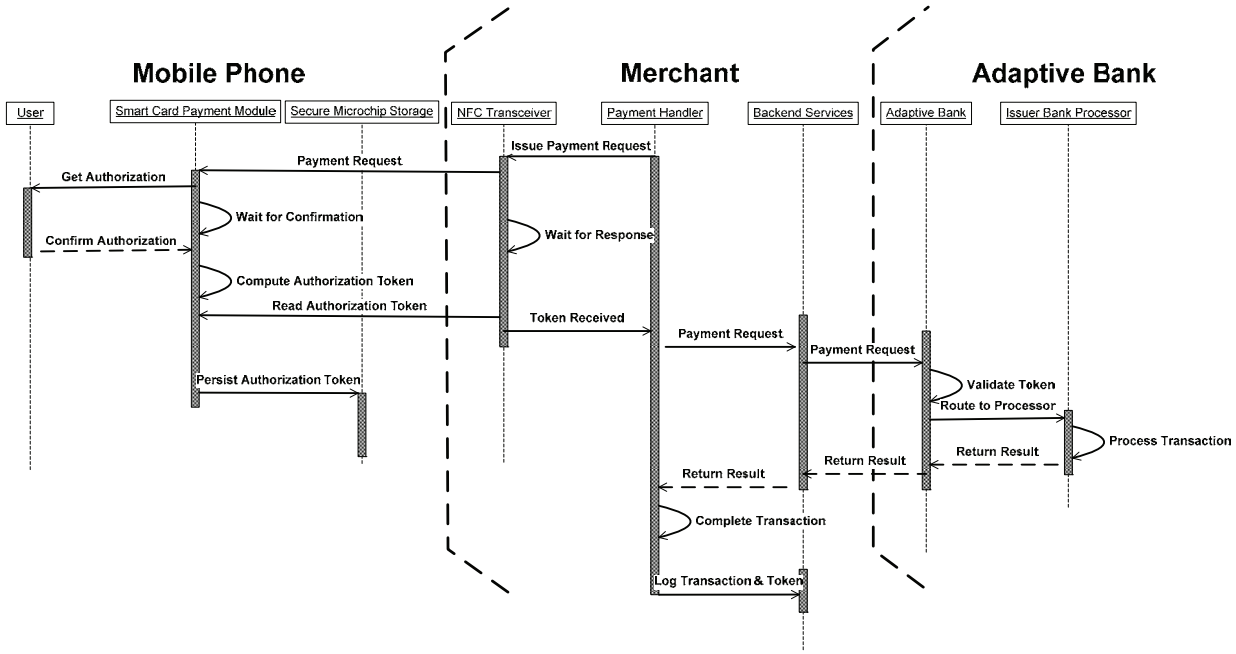


Figure 4. Sequence interaction on mobile phone, local machine and Adaptive Bank.

which can take place at a bank branch office. The private key never leaves the mobile phone. The bank's public key is also imported to the mobile phone's local certificate store (on the NFC chip) during software installation. Once the mobile payment application is installed on the phone, user registration can take place at the bank's branch as well. The mobile phone exports the user's public key to a local machine at the bank's branch, from which the public key is eventually stored in the bank's central certificate store and associated with the user's bank account number. When the user's bank account is closed, the corresponding public key at the central certificate store is revoked, disallowing further mobile payment transaction processing.

If the user has multiple accounts at the bank (e.g., checking and saving), he or she can select the desired account to associate with the authorization token. However, they share the same public/private key pair in the mobile phone. The mobile payment solution can support multiple banks as well as multiple accounts in one bank. Under this situation, each bank will supply its own public key and bank routing number to the mobile payment software installed at the mobile phone.

6. USER INTERACTION COORDINATION AND MANAGEMENT

The mobile phone and local machine exchange data over the NFC channel, as shown in Figure 4. Due to the NFC proximity requirement, the mobile phone needs to be positioned close to the local machine's NFC reader to enable message exchange twice: to receive the payment amount for user verification, and to send back the authorization token. Correspondingly, the customer is prompted to do the following steps by the transaction issuer device (local machine):

(1) Put the mobile phone close to the NFC reader so that the payment amount can be written to the NFC IO accessible storage (shown as smart-card microchip in Figure 2) on the mobile phone by the local machine;

(2) Confirm payment amount on the mobile phone and ultimately the authorization token is written to the NFC IO accessible storage on the mobile phone;

(3) Put the mobile phone close to the NFC reader again so that the local machine can read mobile phone's NFC IO accessible storage that contains the latest authorization token.

The send/receive status at the transaction issuer (local machine)'s NFC IO is communicated to its payment processing UI, as shown in Figure 5 (left-hand side), to keep human in the loop. Figure 5 (right-hand side) shows the payment confirmation step on the mobile phone.

Notice that the UI of the mobile phone is critical in our solution, a concept called Splitting Trust [2]. The mobile phone is personally owned and trusted by the consumer but not the merchant. The local machine that issues the transaction request is trusted by the merchant, but not the user. By having the phone UI show the required payment amount and then allow user to confirm the amount, the local machine cannot cheat or alter the actual amount that the merchant includes into its transaction request to the bank, because the authorization token is only valid for the amount confirmed by the customer.

A user profile hosted on the phone simplifies the interaction and provides further payment protection. The user profile is created when the mobile payment software is installed to the phone and can be updated later by the user. The user profile can support the following payment related configuration information. A payment threshold can be defined to skip the verification step for small amounts, or to activate password or biometric based authentication on the mobile device for large transactions. A limit can be also set for the total amount that can be issued from the phone within a day. A payment choice can be given to the user to debit a different banking account or a digital wallet [17]. Because the mobile phone is viewed as a personal device, rather than being shared among multiple users, our system only supports a single user profile.





Figure 5. Mobile phone to local machine interaction (the left-hand side screenshot shows the steps displayed on the RSA based kiosk to guard user interaction, and the right-hand side screenshot shows payment confirmation on the mobile phone).

Apart from the smart-card chip, a second micro-chip (called storage micro-chip in Figure 2) can be installed in the phone for secure storage, allowing mobile users to store more shopping-related information on the phone, such as transaction receipts, coupons, *etc*., when other mobile services (*e.g.*, coupon offering) are provided as part of the integrated solution package to enrich consumer mobile experiences.

7. REMOTE TRANSACTION ENABED BY NFC

The proximity interaction detailed in Section 6 occurs when the transaction issuer is an in-store kiosk or point-of-sale, *i.e.*, a machine close to the mobile phone. Our solution also supports remote transactions by extending the solution architecture detailed in Section 4 with a web browser located at the store and a remote web-based payment application, while following the secure transaction protocol detailed in Section 5. Such remote transactions enable secure online shopping, such as from a home PC or a home multimedia appliance like HP TouchSmart. A typical online shopping site nowadays requires full credit card information. Even though the payment transaction is done via HTTPS and thus protected during message transmission, the online merchant will receive and then store the user's full credit card information, for example, for future shopping purchases or refunds. As a result, online merchants and traditional brick-and-mortar merchants share the same customer data retention behavior and thus potential risk of a data breach. Our secure transaction protocol can help online merchants avoid such risk. The security provided by our solution is similar to what Citi's Virtual Account Numbers provide [5], but we eliminate the effort of acquiring the virtual account number over the web for every online purchase.

The detailed protocol for NFC enabled online payment transaction is shown in Figure 6. The user makes an online purchase from a web site. The shopping session is linked to a *global payment session identifier* at the backend. At the payment web page, once a payment button is clicked, a certified .NET ClickOnce application [21] or a Java applet, called a *mediator*, is downloaded to and then launched from the machine that hosts the web browser. Once the mediator detects that a NFC reader is attached to the local computer, it starts a local payment session. The local payment session will inherit the global payment session identifier at the time the mediator is launched. For example, in .NET, the ClickOnce application can accept input parameters at the time it is launched as environment variables that are then retrievable in the launched ClickOnce application.

The local session handles the exchange of the payment information and authorization token between the mobile device and the local machine through a local NFC reader, following the same procedure described previously. Once the authorization token is acquired, the downloaded mediator migrates the purchase session back to the remote purchase web site and terminates itself at the local machine. The session migration is achieved by having the mediator invoke the backend session management service, whose end-point reference is pre-specified in the downloaded mediator application. This web service invocation is done via the HTTPS channel, with input parameters that include the global payment session identifier and the authorization token. Once the session management service receives the global payment session identifier and the authorization token, the previous online shopping application that had been suspended will resume.

At this point, the transaction continues at the online purchase site, through the merchant backend and then Adaptive Bank, following

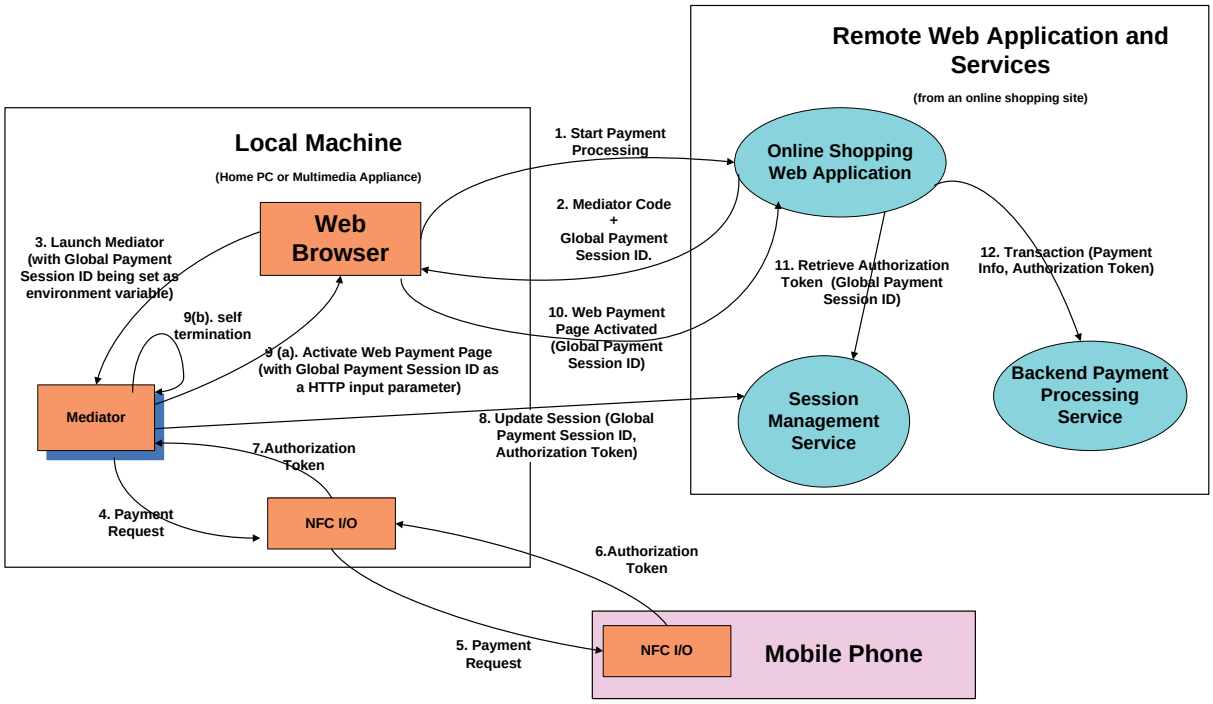


Figure 6. Remote web-based payment transactions enabled by local NFC.

the steps described previously in Section 5. Ultimately, the transaction result is acknowledged to the user through a web page from the online purchase site.

The downloaded mediator's security guarantee is rooted at the trusted .NET runtime installed at the local machine. Code integrity is checked before the mediator is launched. Restriction on local resource access is enforced by the .NET runtime through the Code Access Security mechanism (CAS) [21]. The access control policies enforced by CAS consist of the ones that are defined by the local machine, and the ones that are embedded in the downloaded code and defined by the code publisher (for example, to restrict remote communication only back to the publisher site). In our solution, the code publisher is the online purchase site that supports NFC-based payment solution.

Note that this protocol is not subject to keystroke loggers because critical information is passed via the NFC channel, not the keyboard.

8. PROTOTYPE IMPLEMENTATION

In our prototype, we used a Nokia 3220 mobile phone with a NFC enabled jacket and a secure Mifare memory chip. The RSA web-based application serves the same function as the in-store kiosk touch point for store purchases. The web-based kiosk application is implemented in ASP.NET and it communicates with the merchant backend via .NET web services. The merchant backend directly interacts with the Adaptive Bank mobile banking channel for transaction requests, by using the XML document exchange supported by the .NET Web Service Enhancement (WSE) 3.0. The Adaptive Bank mobile channel has a special HTTP listener installed to handle mobile payment transaction requests.

On the mobile phone, the payment application consists of two parts. The UI is a Java-based Midlet designed to run on the J2ME virtual machine. The payment module is a Java card Applet application [13] developed in Java using the JCOP API's from IBM and the NFC-related SDK from Nokia. We also used the JetZ API from NXP for secure access to the Philips' Mifare chip, which works as our secure storage chip. Since the kiosk application is web-based, it relies on our remote transaction technique (with .NET ClickOnce application download) to communicate with the mobile phone. The NFC reader on the kiosk was developed in Java. A CORBA communication framework [19] enables the cross-language communication between the NFC reader and the .NET ClickOnce application. Should a .NET based NFC reader SDK become available, we will not need the CORBA-based cross-language communication.

9. RELATED WORK

There is a large collection of mobile banking applications that has been developed, piloted and deployed since late 1990's. Here we only contrast our work with other mobile payment solutions, with or without the NFC enablement.

The contactless smart card relies on NFC to send data stored on the smart card to host devices equipped with NFC readers. The Sony Felica Card is one of the most successful contactless smart cards [10]. A Felica card stores an integer counter representing a pre-paid credit. The reader reduces the amount of available credit through a secret-key sharing protocol. Different data access rights

are granted based on the secret presented by the reader. The services supported by Felica include transportation passes, electronic wallets, loyalty programs. Such contactless cards are very convenient and ease-to-use, due to their tap-and-go interaction with the reader. The payment process is completely local and no backend processing is involved. Because the card lacks a UI, users must fully trust the host devices to deduct only the agreed amount.

Contactless smart cards (NFC chips) can be combined with mobile phones to form NFC-enabled mobile phones. Such mobile phones, including NTT DoCoMo's i-mode Felica [15] and Nokia NFC phones [20], use the phone UI to display secure data (for example, the remaining credit) stored on the attached chip, and allow or disallow the payment transaction based on UI commands. However, secure transactions still go through the NFC chip's local data store, as done on the i-mode Felica. Alternatively, in MasterCards's NFC pilot with Nokia [3], the NFC chip allowed the reader to fetch the customer's Primary Account Number (PAN). By contrast, in our solution, the payment transaction processing happens at the backend, and the NFC chip only issues authorization tokens. No transaction data is updated on the mobile phone. Hence, our solution is more flexible for the merchant, because the merchant is not required to have the secret keys pre-distributed for access to the NFC chips [14]. That means the merchant is not locked into a particular payment service provider. Furthermore, since only the authorization tokens, but not NFC stored personal data are revealed to the merchant, our solution is more secure as well.

NFC-enabled mobile phones do not require mobile carrier networks for payment transactions. There are alternative payment solutions that do. Some are based on SMS text messaging [8]. Others are either mobile browser applications or mobile device applications, which all rely on the WAP enabled internet [23, 22]. All of these payment solutions require a mobile carrier to provide the wireless communication channel. In SMS messaging based solutions, the payment request (payer's phone number, amount, and even PIN) is carried in one SMS message. A major disadvantage is that such SMS payment solutions are inherently not real-time.

The mobile browser or device applications mentioned above are no different from what is already provided in today's online payment services, such as PayPal or Google Checkout, except that such applications run on mobile devices. The user's password is typically presented to the service provider for authentication. Although the mobile payment process is secured by HTTPS, it is suitable only for online purchases, *i.e.*, the ones with the transaction requests being originally issued from the mobile environment and completed in the mobile environment. They are not designed to have the mobile device to be inserted into the transaction path in which the transaction is an in-store purchase issued from an in-store POS terminal, or an online purchase issued from a browser running on a home desktop. In contrast, our NFC solution works for both in-store purchase terminals and home desktops via the same secure transaction protocol.

In all mobile payment schemes, a virus could take full control of the phone and communicate with a malicious party within and beyond the carrier network, as reported by the mobile phone-based Trojan named Redbrowser.a [29] that attacked mobile devices that run the J2ME version of the Java virtual machine. The only protection our protocol provides is the confirmation step, but it is not clear that a virus cannot simulate a button press. The best solution is to provide a more secure operating system that better isolates applications running on the phone [28].

Finally, with respect to secure payment protocol, Hamann et al. [12] described a Java-based online transaction technique that relies on a smart card device attached to the local machine through a particular wiring protocol (*e.g.*, USB) for online transaction authorization. Fundamentally, the local machine is viewed as a trusted machine and thus the system is designed to have PIN number entered from the local machine in order to activate the smart card authorization process. Chaum's digital cash protocol [4] is designed for anonymous cash in which the bank that issues the cash can not distinguish the cash holders. Although our payment solution has the authorization tokens anonymous to the merchants, the tokens have to be decrypted and then validated by the bank, and thus the tokens in our solution can be fully tracked by the bank.

Overall, our solution significantly differs from the work described above. It provides integrated features that include the trusted personal phone UI, the authorization-based secure transaction protocol that relies on signing the customer account number and then obfuscating it using the bank's public key, the dual support of proximity and remote transactions to bring the same mobile payment experience to both in-store shopping and online shopping, and the flexible payment integration powered by Adaptive Bank.

10. CONCLUSIONS

We have designed and implemented the NFC-based mobile payment solution. It demonstrates the integration of the mobile devices with current existing Point-of-Sale (POS) retail payment solutions. By issuing payment authorization tokens from the mobile devices, the solution eliminates the transfer of any sensitive user information over the public networks, such as bank account number (PAN) and user passwords (PIN). The prototype has been showcased to various HP prospective customers. In our next steps, we plan to perform end-to-end system performance and mobile user evaluation to better character our solution architecture. To enrich the solution features, we plan to integrate the digital wallet [17] as a special payment option in our solution. To further assess the solution's practicality, we will investigate how to adopt our secure transaction protocol and security tokens in the current VisaNet routing network, which routes the transactions between the acquiring banks and the issuing banks. Finally, to improve security and privacy of our solution, we will look into anonymous digital cash, and explore the mobile Trusted Platform Module (TPM) for building secure mobile applications [18].

11. ACKNOWLEDGEMENTS

Bob Lyons introduced the digital wallet concept and provided insightful suggestions.

12. REFERENCES

[1] Adaptive Bank, http://h71028.www7.hp.com/enterprise/cache/255537-0-0-225-121.html.

[2] D. Balfanz and E. W. Felten, “Hand-Held Computers Can Be Better Smart Cards,” Proceedings of the 8th USENIX Security Symposium, 1999.

[3] Celent, “Mobile Commerce: Dealing with the Devil in the Details—How Banks, Carriers, and Device Manufacturers Are Gearing Up for Mobile Commerce,” Feb. 2006.

[4] D. Chaum, “Security without Identification: Transaction Systems to Make Big Brother Obsolete,” *Commun. ACM* 28, 10 (Oct. 1985), 1030-1044.

[5] Citibank Virtual Account Numbers, https://www.citibank.com/us/cards/vanpromo/cmc_pop/index2.htm.

[6] CNET news: T. J. Maxx Hack Exposes Consumer Data, available at http://www.news.com/T.J.-Maxx-hack-exposes-consumer-data/2100-1029_3-6151017.html.

[7] Computerworld security news: Heartland data breach could be bigger than TJX's, available at http://www.computerworld.com/action/article.do?command=viewArticleBasic&articleId=9126379.

[8] R. Dodin, “Text Message Payment,” Google’s United States Patent Application, No. US 2007/0203836, published on Aug. 30, 2007.

[9] EMV, http://en.wikipedia.org/wiki/EMV.

[10] Felica Product Information, accessible at http://www.sony.net/Products/felica/.

[11] Forrester Consulting, “The State of PCI Compliance,” Sept. 2007.

[12] E. Hamann, H. Henn, T. Schack, and F. Seliger, “Securing e-Business Applications using Smart Card,” *IBM System Journal*, Volume 40, Number 3, 2001. http://www.research.ibm.com/journal/sj/403/hamann.pdf.

[13] Java Card, http://java.sun.com/products/javacard/.

[14] M. Jurgensen, S. B. Guthery, *Smart Cards: The Developer’s Toolkit*, Prentice Hall, 2002.

[15] A. Kumar and D. Jaffe, “Mobile Payments in Asia: Breaking New Ground,” Financial Insights, April 2006.

[16] J. Li, I. Ari, J. Jain, A. H. Karp and M. Dekhil, “Mobile In-Store Personalized Services,” Proceedings of International Conference on Web Services, 2009.

[17] B. Lyons and R. Murthy, “Method and System for Managing Monetary Value on A Mobile Device,” HP Patent Application (Ref. 200501925-1).

[18] Mobile Phone Work Group in Trusted Computing Group, https://www.trustedcomputinggroup.org/groups/mobile.

[19] K. Moore, and E. Kirshenbaum, “Building Evolvable Systems: the ORBlite Project,” Hewlett-Packard Journal, pp. 62-72, Vol. 48, No.1, Feb. 1997.

[20] Nokia NFC Phone Web Site, http://www.forum.nokia.com/Resources_and_Information/Explore/Mobile_Technologies/Near_Field_Communication/.

[21] B. Noyes, *Smart Client Deployment with ClickOnce: Deploying Windows Forms Applications with ClickOnce*, Addison Wesley Professional, Dec. 2006.

[22] Obopay, http://www.obopay.com/

[23] PayPal Mobile Checkout, https://cms.paypal.com/us/cgi-bin/?cmd=_render-content&content_ID=developer/howto_api_mobile_checkout

[24] PCI DSS (Version 1.2), Oct. 2008. https://www.pcisecuritystandards.org/security_standards/pci_dss_download.html.

[25] PCI DSS Requirements and Security Assessment Procedures (Version 1.2), Oct. 2008. https://www.pcisecuritystandards.org/security_standards/download.html?id=pci_dss_v1-2.pdf.

[26] C. Radu, *Implementing Electronic Card Payment Systems*, Artech House, 2003.

[27] RSA in-store web-based kiosk application, accessible at http://opra.hpl.hp.com/RSALaunch/flash.html (required: 1280x1024 for display setting; pop-up enabled on Internet Explorer).

[28] D. Toll, P. Karger, E. Palmer, S. McIntosh, S. Weber, “The Caernarvon Secure Embedded Operating System,” SIGOPS Operating Systems Review , Vol. 42, Issue 1, Jan. 2008.

[29] Trojan.Redbrowser.A, http://www.symantec.com/security_response/writeup.jsp?docid=2006-022814-5027-99.

ACM DIGITAL LIBRARY

Proceedings

Search within Proceedings

ICEC '09: Proceedings of the 11th International Conference on Electronic Commerce

2009 Proceeding

Conference Chairs: Patrick Y. K. Chau, Kalle Lyytinen, + 3

Publisher: Association for Computing Machinery, New York, NY, United States

Conference: ICEC '09: International Conference on E-Commerce • Taipei Taiwan • August 12 - 15, 2009

ISBN: 978-1-60558-586-4

Sponsors: School of Business, The University of Hong Kong, Hong Kong, Sayling Wen Cultural & Educational Foundation, Ministry of Education, Taiwan, College of Information Science and Technology, Drexel University, USA, Weatherhead School of Management, Case Western Reserve University, USA, College of Technology Management, National Tsing Hua University, Taiwan, National Science Council, Taiwan, Chinese Enterprise Resource Planning Society, Taiwan, International Center for Electronic Commerce, Korea Advanced Institute of Science & Technology, Korea

Recommend ACM DL

ALREADY A SUBSCRIBER? SIGN IN

Alerts Binder Citation

Feedback