Secure Mobile Payment via Trusted Computing

Qi Li[1], Xinwen Zhang[2], Jean-Pierre Seifert[2], Hulin Zhong[3]
[1] Dept. of Computer Science, Tsinghua University, Beijing, China
liqi@csnet1.cs.tsinghua.edu.cn
[2] Samsung Information Systems America, San Jose, CA, USA
{xinwen.z, j.seifert}@samsung.com
[3] Lutong Network Technologies Co., Ltd., Shenzhen, China
zhonghulin@lutongnet.com

Abstract

Mobile payment (m-payment) received significant attention because it enables an easy payment mechanism and becomes an important complement to traditional payment means. However, m-payment over open devices and networks poses security challenges of a new dimension. Although many researchers address security issues in m-payment, there are still some security problems that are not well resolved, such as platform integrity and user privacy protection. In this paper, we propose a general payment architecture with Trusted Computing (TC) technologies to secure mobile payment. Using only a simple mobile payment infrastructure, a platform integrity protection solution is proposed to secure payment software downloading, application initialization, and secure payment transactions. We further propose two schemes to enhance the performance and flexibility of our solution. The first scheme provides platform attestation using an identity-based signature (IBS) algorithm instead of a traditional credential-based public-key signature algorithm within Trusted Computing Group (TCG) technologies, which fully utilizes the merits of the mobile computing infrastructure and improves the flexibility and performance of the payment solution. The second scheme provides attestation caching without sacrificing security achievements. We have implemented a real prototype system based on an emulated payment environment. Our security analysis and experimental results prove that our scheme can effectively meet the security requirements of a practical m-payment with acceptable performance.

1 Introduction

The increasingly popular mobile communication technologies have demonstrated the applicability of programmable communication devices, including mobile phones and PDAs, for e-commerce applications. These devices are effective for authorizing and managing payment transactions, offering security and convenience advantages compared to online payment via PCs, especially with recent advanced communication technologies for mobile device such as General Packet Radio Service (GPRS) and Near Field Communication (NFC) [5]. However, with the growing use of open and general-purpose mobile devices, increasing software-based attacks have shown that existing embedded operating systems (OS) cannot provide sufficient integrity and isolation protection for the security demands of mobile payment applications. Thus, mobile users are at great risk leaking their sensitive information to malicious software, such as viruses, worms, or Trojan horses, while they perform mobile payments. According to F-Secure [3], currently there are more than 200 mobile malware programs (or viruses) in circulation.

While the majority of existing research focuses on secure payment transactions, there is no intensive research on platform integrity protection for secure payments on mobile devices. For example, most payment schemes only focus on communication integrity and confidentiality, and user authentication and authorization [26], while less research focuses on the security of the client platform itself. Recent software-based attacks are emerging even on mobile devices which is a big threat to payment applications and the respective data stored on mobile platforms, such as credit card and payment history information. If a payment platform is compromised, software and data integrity, and user privacy can be compromised and thus existing proposed secure payment schemes are unable to completely secure payment transactions. Thus, how to protect and verify the platform integrity and achieve high assurance mobile transactions is still an open and important problem. Current secure payment schemes fail to provide a *platform integrity*

978-0-7695-3363-6/08 $25.00 © 2008 IEEE
DOI 10.1109/APTC.2008.24

protection solution for mobile payment transactions. For instance, although secure electronic wallet (e-wallet) applications have been proposed [23], the problem how to establish and verify a secure runtime environment of an e-wallet software was never addressed until now. The present paper aims to provide a general security solution towards this open problem for secure mobile payment schemes.

In practice, there are two classical types of electronic payment (e-payment) applications, cash-like and check-like payment systems, which require different network technologies to provide communication channels [12]. Especially, the recent emerging NFC technology has obtained a considerable attention for mobile payments because of its ease of use and relatively secure communication channel [30]. However, securing the payment channel alone does not offer high assurance payment transactions. Indeed, without trusted mobile devices, the security of payment applications and data cannot be guaranteed at all.

To illustrate the benefits of our proposed security architecture we use NFC which provides direct connections between mobile devices and merchants (e.g., via points of sale — POS) during a payment process. To illustrate our solution, we propose a secure mobile payment scheme with a primitive architecture to address the security requirements of mobile payment systems. Our scheme focuses on protecting and verifying the payment's platform integrity and as well the user privacy. Specifically, our proposed architecture provides strong platform attestation with the help of Trusted Computing (TC) technologies [11]. With TC enabled, the platform integrity is validated before actual payment application, including e-wallet application and its respective runtime environment. In this way, malicious software can not hide themselves and monitor or participate payment process and thus our scheme provides strong protection of mobile payment. Moreover, all private user information is protected by keys which are sealed into a secure subsystem — Trusted Platform Module (TPM) or Mobile Trusted Module(MTM). Only when a payment platform is in a good state, the user information including the user's account information can be unsealed from the TPM to the respective continue payment transaction. Therefore, our solution provides a very strong protection of user privacy. In order to improve the attestation performance, we also propose two enhancement techniques. First, we fully utilize the merits of mobile computing infrastructure to improve the flexibility of mobile payment. We use identity-based signature schemes (IBS) to verify a mobile platform's integrity by using the respective mobile phone number as the digital identity of the corresponding platform. This approach eliminates the attestation identity key (AIK) certificate management from standard TC technologies and improves the efficiency of our mobile payment scheme. Our second enhancement technique proposes the use of an attestation cache to reduce the attestation overhead and improve the payment performance without sacrificing critical security requirements.

The remainder of this paper is organized as follows. An overview of mobile payments and TC technology is presented within the next section. Our overall secure payment architecture is then first illustrated in Section 3 while Section 4 proposes just a basic secure payment scheme. Later Section 5 provides our two enhanced payment schemes for better efficiency and performance. We describe our implemented prototype and a respective performance study in Section 6. Eventually, Section 7 and 8 presents related work and concludes this paper, respectively.

2 Background

2.1 Mobile Payment

A typical mobile payment scheme involves three parties: a mobile device (representing a payment customer/user), a merchant, and a financial service provider (e.g., a bank or credit card service provider). To secure a payment transaction, a trusted third party (TTP) is involved to authenticate and authorize users. In most use cases, the TTP is part of the financial service provider, e.g., for the easy deployment purpose. There are two types of e-payment applications: check-like and cash-like payments. Check-like payments require a certain amount of *virtual money* which is taken away from the customer before a payment is made, and the customer spends virtual money through a local area network or a micro-money (m-money) supplier [12]. On the other hand, cash-like payments require that a customer's account is involved in each payment transaction. Typical credit card based payments fall into this type. As cash-like payments are much more popular nowadays, we focus on this type of payment in this paper, though our solution can also be used to secure check-like payments.

Figure 1 shows related parties in cash-like mobile payment schemes. As a prerequisite condition, the customer needs to register his account information with the TTP or a financial service provider, and store respective payment information on her device, such as the corresponding bank account and other billing information. At first, the customer initiates a payment request to the merchant, by running a payment application on the device which transfers payment information to the merchant via a communication channel (e.g., NFC). The merchant then forwards the request to the financial service provider (e.g., a bank), and then the bank authenticates and authorizes the customer with the help of a TTP. If the customer is successfully authenticated and authorized, the account is debited. At last, a notification of payment (e.g., a printed receipt or e-receipt) is presented to the customer and the merchant, and the complete payment

transaction process finishes. Here a mobile user can also manage his account online on the financial service provider side.

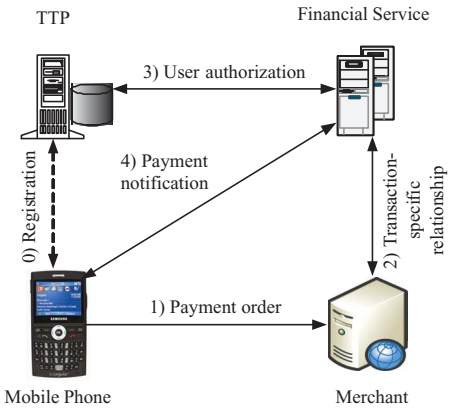

Figure 1. Mobile payment system.

General security requirements of mobile payments have been well studied in the literature [12] — such as user authentication and authorization, and integrity and confidentiality of communication channels. However, as we have mentioned in Section 1, with the increasing use of open and general-purpose mobile computing platforms, mobile phones face the intrusion of different malwares, which seriously threat the integrity of mobile platforms. For example, a malware can install itself via vulnerabilities of running applications on the phone and steal a payment account, or compromise the payment application and thus make payments to a remote server on behalf of the user. Existing security mechanisms in mobile payment fail to resolve these problems. In contrast, our secure payment scheme leverages the attestation mechanism of TC technologies to provide integrity verification for mobile platforms and there achieves a high assurance for mobile payment users, merchants, and financial service providers.

2.2 Trusted Computing

The Trusted Computing Group (TCG) [11] has defined a set of hardware and software specifications for TC technologies. The key of the TCG architecture is the Trusted Platform Module (TPM), a discrete chip which performs certain cryptographic functions and provides "secure storage". A TPM chip itself is identified by an Endorsement Key (EK) credential which is issued by the TPM manufacturer. Each user (or application) of the platform is identified with an Attestation Identity Key (AIK), whose private key is protected by the TPM, whereas the public key is certified by a trusted third party. The platform state is also identified by hashing the system-state into a set of 160-bit hash values based upon its hardware and software configurations to represent whether it is in a known "good state". The respective hash values are stored in so called Platform Configuration Registers (PCRs) of the TPM.

Figure 2 illustrates the logical architecture of a platform with the TPM chip. The kernel and critical services and processes are measured by the measurement agent and the measurement values are reported into the PCRs. As the number of components in a platform is more than the available PCRs, many measurements are extended into one PCR by deriving a hash concatenation of the new measurement with the existing PCR value. The measurement sequences are also saved within an Measurement List (ML) outside of the TPM for the later attestation purposes.

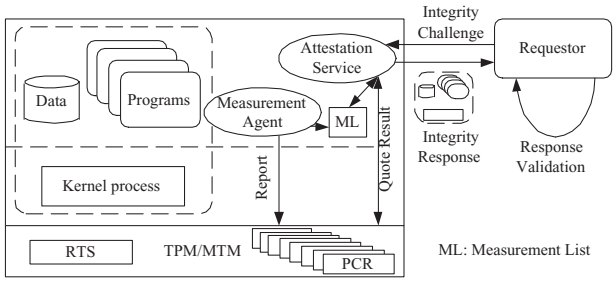

Figure 2. TPM-based platform architecture and attestation mechanism.

As a key mechanism defined by the TCG, attestation is used to report the measured PCR values to a requestor who needs to know the runtime-state of a platform. Figure 2 shows a typical attestation process. Besides the hardware TPM, there are three major components involved in this process:

- *System components* include kernel and user space processes, whose hash values are used to validate whether the runtime environment of the platform is in a good state.
- *Measurement agent* measures the state of the runtime environment of the platform and reports measured hash values to PCRs. At the same time, the measurement agent needs to maintain the measurement list for later attestation purposes.
- *Attestation service* interacts with the TPM and provides the platform integrity metrics which are digitally signed by an AIK.

During an attestation process, the attestation service collects the requested PCR values along with their indices,

signs them using the AIK of the responder, and reports the results to the requestor along with the ML. Using the ML values, the requestor recomputes the expected hash values of the platform components and compares them with the PCRs. As the PCR values stand for the state of the components of the platform, the requestor can be assured that the platform is in a known good state if all the PCR values are valid by comparing them with known good values. The integrity validation is evaluated by the requestor or a TTP on behalf of the requestor. Apart from platform attestation, the TPM also performs secure storage functions like wrapping and sealing which are also needed for secure m-payment transactions in our scheme.

Typically, a trusted boot mechanism is also required for a trusted platform, e.g., with the help of a core root of trust for measurement (CRTM) and the TPM itself. Only after all booting components are measured and reported to some PCRs, the measurement agent can take charge of measuring the applications and hereafter the attestation service can respond to corresponding requesters. Integrity measurements from the booting sequence can also be included in an attestation response, according to the requirements of the requester. However, this paper does not consider a particular trusted boot mechanism, but just assumes that such a mechanism is in place. For simplicity, we simply ignore the details of trusted boot and corresponding integrity measurements in the rest of this paper.

In order to provide strong protection and security mechanisms for mobile platforms, the Trusted Computing Group (TCG) has published specifications of a Mobile Trusted Module (MTM) [4] — a modified version of the TPM — tailored for constrained mobile platforms. Here, typically, a mobile phone is owned by multiple stakeholders, including the: device manufacturer, network operator, third party service providers, and the user (customer) itself. One owner cannot turn off or damage services of another owner. A MTM can be owned by local and remote stakeholders, each of which needs to have their own basic trusted computing mechanisms such as secure boot, secure storage, integrity measurement and verification, and remote attestation, which itself requires the deployment of a separated MTM for each stakeholder.

The MTM will support complete isolation and controlled communication and resource sharing between the different stakeholders. Since this paper does not discuss the different requirements of these stakeholders and the MTM performs similar functions as the TPM in our schemes, we will use the general term *TPM* referring to TPM or MTM within the latter sections of this paper.

3 Overview of Secure Payment Scheme

As we have mentioned in Section 2.1, a typical mobile payment scheme involves three parties: a mobile phone, a merchant, and a financial service provider (bank). A trusted third party (TTP) is usually involved to authenticate and authorize users, and acts most of the time for the financial service provider, e.g., for the ease of deployment purpose. In the following section, we will propose a platform integrity protection solution for the whole secure mobile payment process, including the downloading process of a secure payment application such as an e-wallet. Therefore, a software provider appears as a party of our payment architecture. As illustrated in Figure 3, our architecture consists of five major parties for a complete secure m-payment solution:

- *Mobile phone*: A trusted mobile device consists of a TPM and trusted services which provide the integrity evidences of the platform. In this paper, we assume that an m-payment application directly runs within a Trusted Java [10], and the TPM and the Java Virtual Machine (JVM) compose the Trusted Computing Base (TCB) of the underlying mobile platform.

- *Software provider*: A software provider provides payment applications in a secure way, such as e-wallet. Actually, in our scheme, there are at least two types of software providers, one provides the runtime environment software (e.g., the JVM) and the other provides the payment applications (e.g., e-wallet). Since software downloading and installation share a similar process, for simplicity, we do not differentiate these two types of providers.

- *Merchant*: Merchants not only need to provide the commodities that customers demand, but also the Point of Sales (POS) devices to authorize customers and guarantee that the payment information is indeed forwarded to the financial service providers in a secure way.

- *Financial service provider*: A financial service provider provides user accounts for m-payments and validates the user payment information during the payment transaction processes. Before checking a user's payment information, a financial service provider needs to check the system integrity of the participating mobile phone with the help of the TC service provide, i.e., to verify that the user's mobile phone-system is in a good or trustworthy system state.

- *TC service provider*: The TC service provider acts as a trusted third party (TTP) to validate whether a measurement list is non-tampered and the system integrity

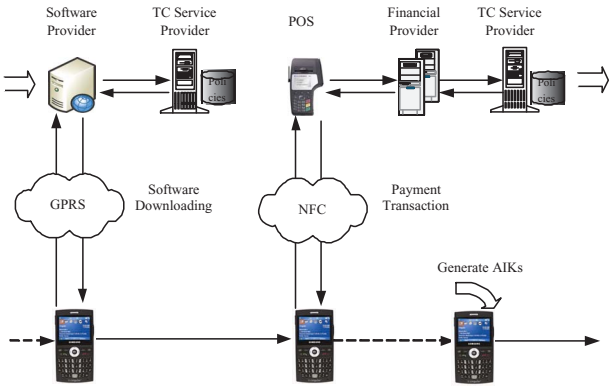

Figure 3. Overview of secure mobile payment procedures.

is in a trusted state. In addition, it also acts as the privacy Certificate Authority (CA) issuer, or as a Private Key Generator (PKG) or a Direct Anonymous Attestation (DAA) issuer, c.f. [11].

Figure 3 shows how different parties jointly perform a secure m-payment transaction. Before any transaction, the mobile user needs to download the e-wallet application from a trusted software provider. After that, the user can perform his m-payment transactions. Different wireless network accesses can be used for communications, such as WiFi, GPRS/UMTS, or NFC. Actually, as aforementioned, our architecture can be used to secure different m-payment solutions and works as well with different types of wireless technologies. Since NFC [5] is expected to be a high-potential technology for mobile services, particularly for mobile payments, we will discuss the secure m-payment with NFC which directly connects a mobile phones to a POS terminal. A POS connects to a financial service provider either with a public network (e.g., Internet via VPN) or a dedicated banking network. The network connections (such as GPRS) for software downloading and other communications are not explicitly specified in our paper. The following Figure 3 gives an overview of our solution.

4 Secure Payment Transactions

In this section, we discuss the secure m-payment process with an *AIK*. An alternative option is to use *direct anonymous attestation (DAA)* for privacy and anonymity purposes [11], or the use for an *identity-based signature (IBS)* for better efficiency and performance purposes, which will be discussed later in Section 5.

Apart from the fact that the connection between a mobile phone and a POS during a payment transaction is provided by NFC, we also assume that other mechanisms are in place to secure the respective connections. Those are needed to fulfill the strong authentication and confidentiality requirements. Security protocols, such as TLS/WTLS [15] in which session keys are dynamically negotiated for every connection can be used for that purpose. But we do not prescribe the actual type of such wireless techniques.

4.1 Components in Mobile Payment

As we have discussed, a secure payment scheme provided by secure hardware within a trusted mobile system should detect any malicious modifications and intrusions of a payment application and/or the corresponding data during their life cycle. Our scheme includes three stages to achieve this goal. First, the specific payment software should be downloaded and installed in a secure manner before any m-payment transaction. Second, a secure e-wallet initiation process is required to generate and protect the secrets (keys and certificates) of the application. Eventually, an m-payment transaction process is performed upon successful attestation of the mobile platform.

- *Secure software downloading* For a secure payment scheme, especially in payment solutions with NFC, e-wallet applications[1] are essential for m-payment transactions. In this context an e-wallet runtime environment is also important. I.e., we need to download this software packages from trusted software providers. Otherwise a malicious software might compromise the platform integrity and the user's privacy. On the other side, as an e-wallet software maintains user payment data such as credit card accounts and billing information, it is typically provided by a financial service provider. For security reason, the service provider typically needs to evaluate the integrity status of the mobile phone before granting download followed by the final installation. In general, this software can be downloaded through different wireless technologies, such as GPRS or WiFi.
- *Secure e-wallet initialization* In order to secure payment transaction processes itself, we also need to secure the e-wallet initialization process. For instance, the private key and the user account information should be generated in a secure environment and then stored inside the secure storage of the target device. In our scheme, we need to validate also the integrity of the mobile phone itself through the help of

[1] M-payment applications or e-wallet applications are referred to as payment software which is installed inside mobile devices. This software maintains the user payment information and provides security related mechanisms. We do not differentiate them in this paper.

a TTP before actually storing the private key with the user account information. The public key generated by an e-wallet application encrypts the user account information and the private key is protected by the TPM. The user's public information should be sent to the TC service provider through GPRS or other wireless technologies while the user registers to the mobile payment service.

- *Secure payment transaction* Similarly, we need to evaluate and validate the integrity of the whole mobile phone before an actual payment process. Prior to starting a payment application, the integrity of the mobile phone is measured and the corresponding measurement values are reported to the PCRs. If the PCR values match the expected values, the sealed private key of the e-wallet can be unsealed from the TPM. Also, the private key decrypts the user account information. Eventually, the user sends out the account information with its signature after the integrity of the mobile phone is validated by the TTP. Typically, in our scheme, all the information including the measurement values is sent out via NFC which provides at this stage enough security protection. As noted, we assume that the POS terminals at merchants have a direct and secure connection to financial service providers, and merchants are not directly involved in payment transaction processes.

Before we introduce our secure m-payment scheme, we need to assume that the *AIK* of the payment application and the public key of the software provider are present inside the mobile phone in advance. In order to use trusted m-payment protocols, the key pairs of an *AIK* should be generated inside the TPM of the mobile phone and the *AIK* credential should be signed and retrieved from a privacy CA. In our scheme, the TC service provider also acts as a privacy CA, and a payment platform receives an *AIK* when the user registers to the mobile payment service or updates the *AIK* pair.

4.2 Secure Software Downloading

Figure 4 shows how an application software is downloaded from a trusted software provider. This process consists of two stages, the first stage is integrity measurement and the second is software downloading. A measurement request is generated by the application manager of the mobile device which takes control of downloading software, e.g., the user invokes the application manager to download the runtime environment software or the m-payment application, and the measurement service initiates the respective measurement operation. During the measurement process, a measurement list (ML) is updated by the service and the measurement results are reported to the TPM.

The integrity challenge in the downloading process is triggered by the application manager (step 1 in Figure 4). Upon receiving an attestation request, the attesting service retrieves the signed aggregate from the TPM (step 2 and 3) and as well the ML from the kernel (step 4 and 5). Both are then returned to the application manager within step 6. The application manager sends then this attestation information to the remote software provider — together with the software request (step 7). In case the software provider cannot validate the status of the mobile system, it forwards the measurement information to the TC service provider in step 8, which itself validates the integrity of the mobile phone. After receiving the attestation result from the TC service provider (step 9), the software provider responds to the request if the system status is in a trusted state (step 10). Eventually, the application manager verifies the signature of the software provider and finally installs the software on the phone in step 11.

Figure 5 depicts the secure software downloading protocol. In step 1, the application manager creates a non-predictable 160 bit random *nonce* and sends it in a challenge request message to the attestation service. In step 2, the attestation service loads the *AIK* into the TPM. This *AIK* is encrypted with the storage root key (SRK), a key known only to the TPM. Then, the attestation service requests a *Quote* from the TPM that now signs the selected *PCR*s and the *nonce*. After achieving the complete measurement list (ML) from the measurement agent, the attestation service responds with the response message to the application manager in step 4. The respective response message is then forwarded to the TC service provider by the software provider in step 5 and 6.

In step 6a, the TC service provider first verifies the *AIK* certificate which binds the verification key of the *Quote*. This credential must be verified to be valid, and then the signature is verified by the TC service provider in step 6b. In step 6c, the TC service provider verifies the *Quote* and the *PCR*s to check the system integrity of the mobile system after validating the unique and not predictable *nonce*. If the software provider retrieves a positive result of the system state from the TC service provider, it sends in step 8 the signed software to the application manager. In step 8a and 8b, the signatures of the software provider and the software integrity are verified, and then the software is installed in the mobile phone in step 8c.

As aforementioned, there are two processes of software downloading, runtime environment downloading and e-wallet application downloading. These two processes have similar procedures.

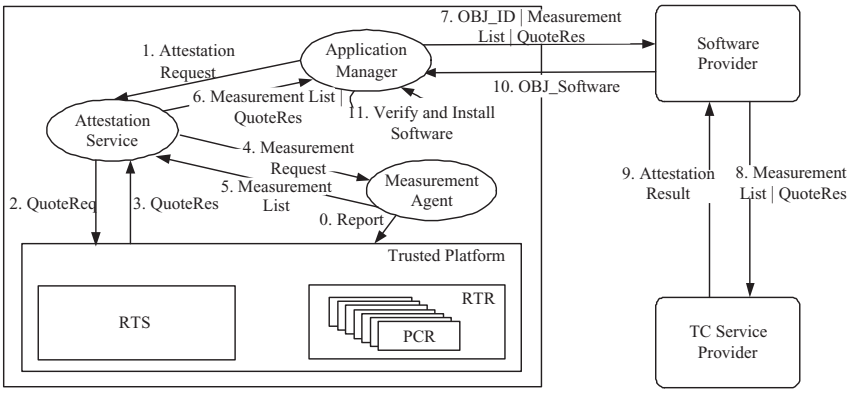

Figure 4. Secure software downloading.

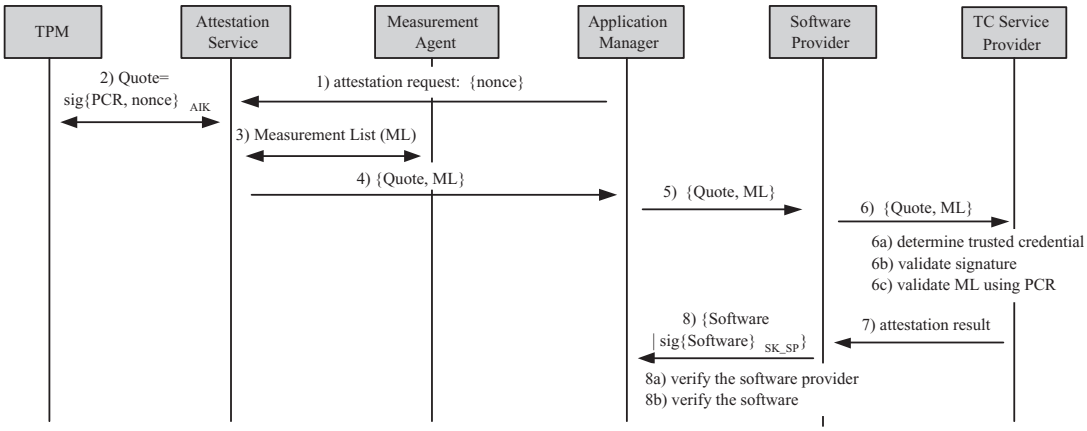

Figure 5. Software downloading protocol.

4.3 E-wallet Initialization

The e-wallet initialization aims to generate a public/private key pair and securely stores the private part for the the m-payment application, which is later used for authentications in payment transactions. Integrity measurement is also performed in this stage before the e-wallet application initialization. The respective attestation procedure is similar to that of the software downloading. Before the applications runs, the system state needs to be reported to the TPM (step 0 in Figure 6). The software application is measured in step 2 and 3 and the integrity value is then reported to the TPM. Hereafter, the measurement list is updated in step 3 and the trusted platform-state of the measurement result informs the initialization of the m-payment application in step 4. Then, the m-payment application starts for the first time and generates an RSA key pair (step 5 in Figure 6). The private key is sealed into the TPM with a key managed by the TPM (step 6), and the public key encrypts the user account information inside the e-wallet. Also, the m-payment application sends the public key to the financial service provider with its signature (step 7). The public key and the signature are forwarded to the TC service provider (step 8). The TC service provider verifies the signature and stores the public key for future verifications, and the verification result is responded to the m-payment application through the software provider (step 9 and 10).

Figure 7 demonstrates the m-payment application initialization protocol. In the first two steps (step 1a and 1b), m-payment related applications are measured and the measurement list is updated. In step 2, *TPM_Extend* is used to extend the PCRs indicating the m-payment application status, and then the m-payment application runs on the mobile system in step 3. After the m-payment application is launched, an *RSA* key pair is generated by the m-payment application. The private key is essential for the secure mobile payment, and is sealed into the TPM by *TPM_Seal*. Then, the *TPM_CertifyKey* is used to certify the public key

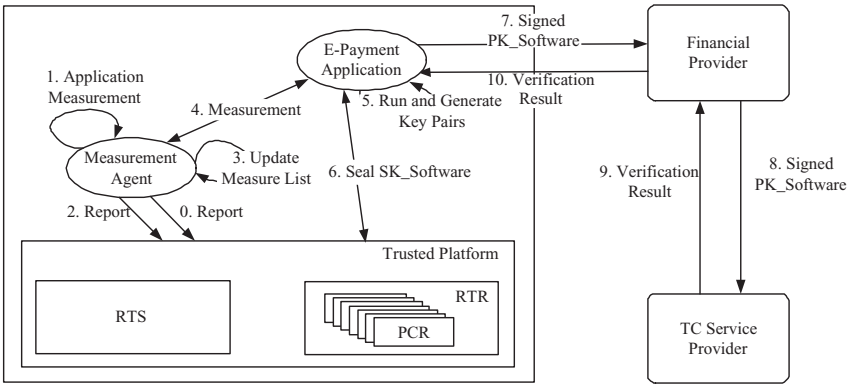

Figure 6. Secure e-wallet initialization.

of the software and the result is sent to the financial service provider in step 6. After receiving the certified public key *cert*, the financial service provider forwards it to the TC service provider. Finally, the TC service provider validates the freshness of the *QUOTE*, verifies the certificate, and returns the verification result to the m-payment application.

4.4 Secure Payment Transaction

Similar to the above two processes, the integrity measurement mechanism is also invoked in the process of secure payment transaction. Before the m-payment application is actually launched, the m-payment application is measured and reported to the *TPM* (step 0 in Figure 8). Then, the m-payment application sends out an attestation request to the attestation service in step 2, and receives the attestation information and measurement list in step 7. Since the attestation process in step 3$\sim$7 is executed in the same way as previously described, we do not repeat it here. After receiving the attestation information and the measurement list, the m-payment application unseals the private key of the m-payment application (step 8 in Figure 8). If the private key successfully decrypts the user account information protected in the e-wallet, both the attestation information and the user payment information with the corresponding signature (signed by the private key of the m-payment application) are then send to the financial service provider. The financial service provider forwards this information to the TC service provider within step 10 and gets the attestation results in step 11. Finally, the financial service provider checks the user account and continues the transaction if it receives a positive attestation result, and the m-payment application eventually obtains the transaction result from the financial service provider in step 12.

The respective payment transaction protocol is shown in Figure 5. Step 1 of the the m-payment application creates a non-predictable 160bit *nonce* and sends it to the attestation service. The attestation service requests a *Quote* from the TPM in step 2 and retrieves the ordered measurement list (*ML*) of the measurements from step 3. In step 4, the *Quote* and the *ML* are retrieved by the m-payment application. The m-payment application uses *TPM_Unseal* to unseal the protected software private key stored within the TPM in step 5. After successfully receiving the private key, in step 6, the m-payment application sends the signed payment information including the user account, the *ML*, and the *Quote* to the financial service provider. This information is forwarded to the TC service provider once the financial service provider receives it in step 7. In step 7a, the TC service provider checks the validity of the received credentials and signatures. In step 7b, the TC service provider validates the freshness of the *Quote* and also the status of the mobile system by checking the quoted *ML*. If the mobile system status is trustworthy, the financial service provider continues the payment transaction in step 7c and deducts the claimed money from the user account. Finally, the transaction result is returned to the m-payment application along with a payment transaction e-receipt, and the whole transaction process completes.

5 Enhanced Payment Schemes

In Section 4, a basic mobile payment scheme was presented, which used standard TC technology to secure mobile payments and provided strong platform protection for secure payments. Although this scheme meets our security requirements, it does not address efficiency and scalability issues at all, which greatly influences mobile payment performance. First, in the above scheme, each mobile payment application needs an *AIK*, which introduces management cost to the overall mobile computing infrastructure, such as certificate management and revocation. Second, in the above scheme the TC service provider is involved in the

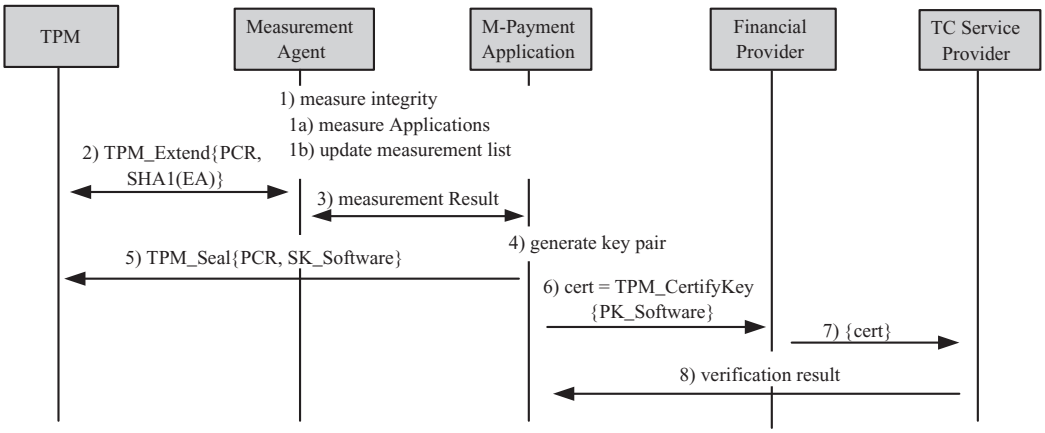

Figure 7. Secure e-wallet initialization protocol.

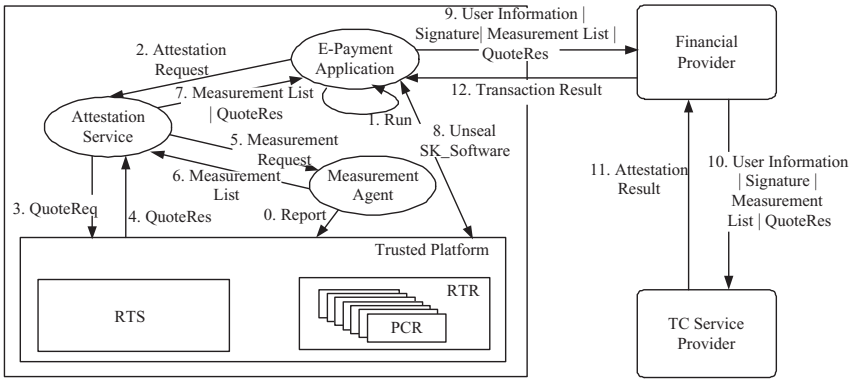

Figure 8. Secure payment transaction.

attestations of every payment transaction. Therefore, this could be a serious performance bottleneck of our payment scheme and as well a single point of failure of the whole system.

In order to address the above issues and improve the overall performance and scalability of our mobile payment scheme, we propose two enhanced mobile payment solutions for different optimization requirements. In the first solution, we leverage the phone number as the device identity to resolve the credential management problem. As analyzed above, the *AIK* introduces additional management cost for mobile users and financial service providers. However, the mobile phone infrastructure provides already a trusted credential itself — the phone number associated with each device, and we simply adopt it to secure the payment scheme using an identity-based signature (IBS) scheme. Through employing IBS scheme, we can remove the cost to manage the *AIK*. Second, for further optimization, we also reduce the TC service provider related attestation steps during payment transactions, and therefore effectively remove the aforementioned performance bottleneck. With our solution we can provide an attestation cache for a much better payment performance which in turn means that the TC service provider will only provide payment attestations for every new user. Note that these two new enhancements of our scheme are independent of each other which means that they can be jointly used in a real system.

5.1 Identity-based Signature for Attestation

In a traditional public key infrastructure, each entity has a public key certificate as its identity from a certificate authority (CA). When there are huge number of users and devices in mobile applications, the authority becomes the bottleneck of security and performance. Beside acting as TTP, a CA also needs to manage certificates and their revocation information. In an IBS system, the public key may be

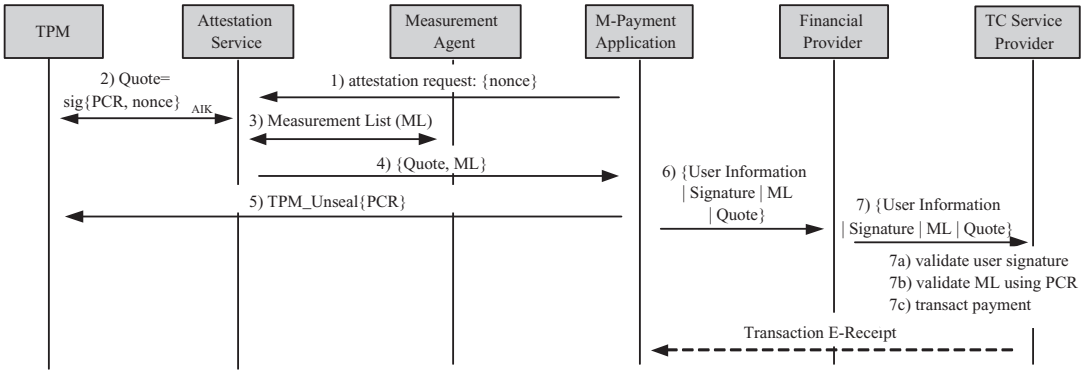

Figure 9. Secure payment transaction protocol.

the mobile phone number, which is provided by a mobile network provider and typically can be trusted by other service providers. For example, in a GSM cellular network, each mobile phone is equipped with a Subscriber Identity Module (SIM), which is a smart card containing user's subscription information and phone book. User information is verified by the network provider when the device joins the network (e.g., whenever the device is booted) and typically is trusted by other service providers. Our scheme leverages this information to improve efficiency and flexibility for mobile payments.

In a typical IBS system, there are four basic algorithms: *setup* algorithm, *extract* algorithm, *sign* algorithm and *verify* algorithm. The functions of these four algorithms are briefly explained as follows.

- *Setup*: Given a security parameter, the algorithm generates a set of system parameters, some of which are made public, and also a master secret, which is kept private.
- *Extract*: This algorithm extracts the private key corresponding to a given public key. It takes the system parameter, the master secret and an identifier ID, which is a public key string such as an email address or phone number, as input, and returns a private key.
- *Sign*: Using the system parameters, a private key and a message as inputs, this algorithm returns the signature of the given message.
- *Verify*: This algorithm uses the system parameters and an ID to check whether the signature equals to what is received. The signature is accepted as valid if and only if this equality holds.

As an example, suppose that Alice wants to send a message to Bob who wants to verify the message signature. First, Alice needs to retrieve the system parameters of the IBS and her private key, where the private key is generated by the setup and extract algorithms along with the system parameters, the master secret and her ID from a Private Key Generator (PKG). Bob also needs to retrieve the system parameters of the IBS from the PKG. After receiving a message from Alice, Bob does not need to retrieve Alice's public key, which usually takes place in a conventional credential-based public key signature. Instead, Bob simply verifies the message with Alice's ID and checks whether the signature is valid.

In our scheme, we assume the PKG is co-located with the TC service provider which executes the setup and the extract algorithms, while the sign and the verify algorithms are carried out by the mobile phone and the TC service provider, respectively. The PKG acts as a trusted third party similar to the CA within our basic secure payment scheme in Section 4, and the phone number is used as the user ID for the IBS scheme. Since most of the procedures before or during mobile payments are similar to those discussed in Section 4, we do not repeat the detailed protocols one by one but simply present the payment transaction protocol here. Figure 10 shows the skeleton of payment protocol using an IBS scheme.

In the setup stage, the TC service provider (TC in Figure 10) acting as a PKG creates a set of public parameters $params$ and distributes those to each user. Once a user gets the private key, the application manager of the mobile device takes the responsibility to seal the key into the TPM. These three steps can be conducted when the user registers the mobile payment service. Thus, a mobile user does not need to retrieve the private key anymore. In the transaction stage, step 4 indicates a message sign operation, where the attestation service located on the mobile phone signs the PCRs using its secret key sQ_{ID}, and sends it to the TC service provider via a POS terminal or other some service providers in step 5. The TC service provider uses the sender's identifier (i.e., the phone number) ID and his

public parameter $params$ to derive the string Q_t and then verifies the message's signature ϱ within step 6. Similar to the attestation process from Section 4, a payment request from a platform which fails in the attestation process (either due to an invalid signature or invalid PCRs values) will be denied by the TC service provider.

Attestation Algorithm

Notation:
ID: Mobile Phone number
MK: The master key of TC
Q_t: A string generated and kept in TC
$params$: The Parameters known to all mobile phones
sQ_{ID}: The private key corresponding to ID
$SK(m)$: The ciphertext of message m
$H(m)$: Hash function
IBS Setup:
1) TC: $(Q_t, sQ_{ID}) = Keygen(ID(number), MK, params)$
2) TC $\rightarrow$ Application Manager: $SK(sQ_{ID}, params)$
3) Application Manager $\rightarrow TPM$: $TPM\_Seal(sQ_{ID})$
Transaction:
4) Attestation Service: $\varrho = Sign_{\{params, sQ_{ID}\}}(PCRs|H(PCRs))$
5) Attestation Service $\rightarrow$ TC: $(PCRs|\varrho)$
6) TC: $Verify_{\{ID, params\}}(\varrho)$

Figure 10. Attestation algorithm with IBS.

In this scheme, we fully utilize the mobile phone infrastructure and replace in the transaction processes the *AIK*-based public key signature with an IBS algorithm. Thus, we can effectively eliminate the credential distribution and reduce the message size during transactions. The enhanced payment scheme improves the mobile payment's efficiency. Since we only replace the signature algorithm and do not change the underlying payment protocols, the enhanced payment schemes achieves the same security goals.

Note that a user's subscription information — such as his phone number — is not bound to a specific mobile phone, e.g., a user can change his network provider by switching to a different SIM card. Thus, sealing a user's private key on a device may not work when a user changes his SIM card on that device, or uses his SIM card on different devices. In a real implementation, the private key of a user can be stored inside a SIM card, which is typically protected by the operating system of the device itself and only authorized processes are allowed to access it. Furthermore, the private key is securely protected in the user's phone, and merchants cannot illegally acquire the private key or the phone number. Thus, no adversary can steal any information of the owner of the corresponding mobile phone.

5.2 Extended *AIK* Certificate for Attestation

In the payment scheme from Section 4, the financial service provider needs to interact with the TC service provider within every payment transaction. In a real world scenario this might be a potential performance bottleneck. Thus, in this section we will propose an enhanced payment scheme with an attestation cache for better performance and flexibility.

There are two stages for downloading and initializing a secure payment runtime environment for the application software. Before these two stages, we need to obtain an *AIK* for every mobile phone registered with a TC service provider. In contrast to the original scheme, prior to the first payment transaction with a mobile phone, the device needs to update the *AIK* credential. The credential will include the platform measurement values, e.g., within the X.509 certificate extension part. To achieve this, a mobile phone generates a credential request which includes the corresponding PCR values and submits this to the TC service provider. The TC service provider issues the *AIK* certificate only if the platform is in valid state, i.e., the measurement values equal those that the TC service provider re-computes based upon the specified *ML* and a set of known trusted values, and the issued *AIK* certificate will include those expected integrity values. Before attesting the mobile phone during a payment, the first six steps are similar to that of the original scheme to measure the mobile phone and sending the measurement values. In the next step, the POS terminal[2] or the financial service provider attests the mobile platform by comparing the measurement values with that of the *AIK* certificate — after verifying the *AIK* and the user's signature. If the *AIK* is valid and the measurement values equal those of the *AIK*, the mobile phone is considered to be in a valid state and the financial service provider can directly deduct money from the user's account and respond to the user with the payment result.

In this scheme, the integrity of the mobile phone is validated when the TC service provider issues an *AIK* certificate, and the expected integrity values are included within the certificate. The POS terminal or the financial service provider attests for each payment transaction the mobile phone on behalf of the TC service provider. Thus, the attestation verification by the TC service provider for every payment transaction is removed and this can significantly improve the performance. Moreover, all the security goals are also achieved. In summary, compared to the original one, several benefits are achieved by this new scheme:

- *Flexibility:* In this scheme, a financial service provider or a POS terminal can directly attest a mobile phone on behalf of a TC service provider, no matter whether it is available during a payment or not. Therefore, it is very useful for mobile payment transactions. However, it has the shortcoming that the *AIK* needs to be updated once the payment application related software is

[2] Although most current POS terminals do not have a signature verification function, it is clear that this is not difficult to implement within modern POS terminals.

updated, e.g., the runtime environment or the e-wallet software is patched.

- *Security:* Although we provide an off-line TC service provider, the integrity of the mobile platform is in any case validated by comparing the claimed measurement values to those embedded inside the *AIK* certificate. If the mobile platform is maliciously changed, the validation fails. Thus, this scheme achieves the same security goals as the previously proposed one.

- *Performance:* Since a TC service provider is not involved in every payment transaction and a financial service provider can directly attest a mobile phone, the performance is significantly improved by removing the communication between the financial service provider and the TC service provider.

6 Prototype Implementation and Evaluation

To evaluate the feasibility of our proposed scheme, which is more applicable to the current m-payment infrastructure, we have implemented a simulation prototype, but we did not simulate the applications on a real mobile phone.

6.1 Prototype Overview

In our prototype, the platform integrity storage is realized by a software TPM [8]. Specifically, Trusted Java [10] is used to provide the TCG Software Stack (TSS). The TSS employs cryptographic methods of the TPM when "establishing trust", and provides functions that can be used by operating systems and applications. Different platforms were developed to act as a mobile device, a financial service provider, a POS terminal, and a TC service provider, respectively.

The financial service provider server and the POS emulator were built on a Windows XP machine with a Pentium III running at 2.8 GHZ CPU and using 512MB of memory. The TC service provider were built on a machine with a Pentium D 933 MHZ CPU and 1 GB of memory. The mobile phone were emulated on a Windows XP machine with a Pentium D 2.8 GHz CPU and 1 GB of memory. All these applications were built on a VMWare Workstation 5.5.2 on Ubuntu-7.04. The software TPM were built with a TPM emulator 0.5 [8], and the TSS were built with Trusted Java [10] including jTSS 0.1 [10] and Trousers 0.2.9.1 [6] together. All network connections were simulated by LAN with TLS protection. The payment application on the mobile phone were developed with Qtopia [9] and the POS emulator and the TC service provider were developed using JDK1.5. Figure 11 illustrates the experimental setup of the prototype system, and Figure 12 shows the GUI of our payment application including a snapshot of the POS emulator during a payment transaction.

We also implemented in the above emulated environment the enhanced IBS scheme and used the MIRACL cryptographic library 5.3.2 from Shamus Software [7]. This library includes Boneh and Franklin's bilinear-paring based IBE scheme [14]. Currently, we only replaced the TPM_Quote operation on the mobile phone and the verification operation of the TC service provider, and directly evaluated the resulting attestation performance. Specifically, we used 11-digit phone numbers as IDs for the IBS scheme with a key size of 512 bits.

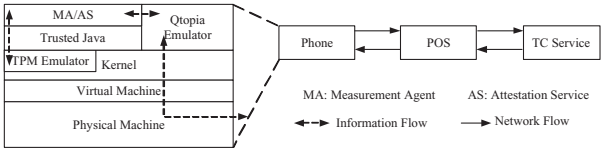

Figure 11. The prototype architecture.

Figure 12. The GUI of mobile phone and POS.

6.2 Performance Evaluation

In our schemes, software downloading and initialization are one-time operations, which do not introduce any runtime performance overhead. Therefore, in our experiment, we only measure the performance of the actual payment transactions. A typical transaction includes user authentication, platform attestation, payment authorization, and operations with a user's account (e.g., money reduction, etc.). First of all, the overhead of authentication, authorization, and account operations of our prototype are the same as those in traditional payment schemes. Therefore, we only evaluated the performance of payment transactions including the integrity attestation operations enforced by our TC service provider.

Figure 13 shows the results of the attestation performance study between a POS terminal and a TC service provider using different concurrent payment requests. We simulated this by generating multiple payment transactions from the same mobile phone. To compare the performance of our scheme to real-life payment applications, we measured the attestation performance with and without TLS protection. The measured time includes the time of the TPM operations such as TPM_unseal and TPM_Quote, the measurement time, the verification time and the overhead introduced by the communication between our three machines. With the increase of concurrent payments, the attestation time varies between 0.04s and 1.60s without TLS protection, and varies between 0.40s and 9.30s with TLS protection. Since the concurrent payment requests are generated from one physical machine, the TLS negotiation is the main performance bottleneck of the resulting attestation performance. In a real-world setting, a mobile phone communicates with a POS terminal via NFC, which has built in security mechanisms to ensure a secure channel [5]. Therefore, the performance without TLS in Figure 13 reflects the real performance of our scheme. If we consider the processing time inside a mobile phone and the back-end database operation, it additionally adds around 1.10s. Thus, a whole payment transaction may cost only 2.70s — even with 100 concurrent transactions to the same financial service server. Compared to traditional credit card payments which typically take around 2 to 4 seconds [1], the performance of our secure payment scheme is very attractive.

We similarly evaluated our enhanced payment scheme using an IBS. Figure 14 shows the attestation time during payment transactions. In this experiment, the measured time includes the time for TPM_unseal operations, the measurement time, the signing and the verification time using the IBS and the overhead introduced by the communication between our three machines. The resulting time varies between 0.03s and 1.57s without TLS protection, and varies between 0.36s and 9.27s with TLS protection. This result demonstrates that this scheme does not degrade the payment performance at all. Figure 15 shows the digital signing and the verifying operations for the attestation itself, which introduces a significant overhead to the overall performance. The signing operation approximately costs 40ms and 25ms using a *TCG AIK* based and an identity-based scheme, respectively. The verifying operation costs about 15ms and 5ms, respectively. This demonstrates that our IBS-based scheme greatly improves the attestation performance compared to that one achieved by using standard TCG technologies.

As an alternative approach, direct anonymous attestation (DAA), as proposed within the TPM 1.2 [11] introduces another attestation mechanism — as opposed to the above privacy CA-based approach. Operations in DAA are typically regarded as time-consuming, e.g., in [2] a $Tspi\_DAA\_Join$ costs 57 seconds and a $Tspi\_DAA\_Sign$ costs 38 seconds. We expect that the overhead of payment schemes relying on DAA is much more than that of our proposed schemes. Thus, we have not implemented DAA in our prototype at all.

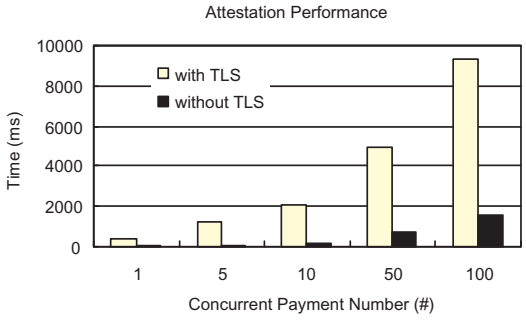

Figure 13. Attestation with CA

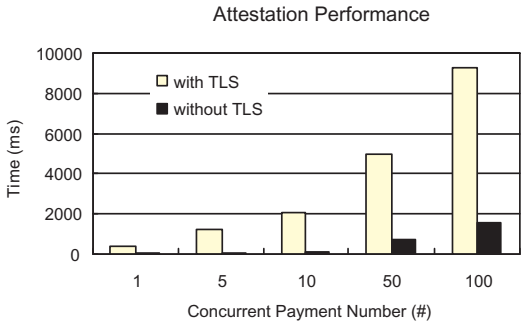

Figure 14. Attestation with IBS

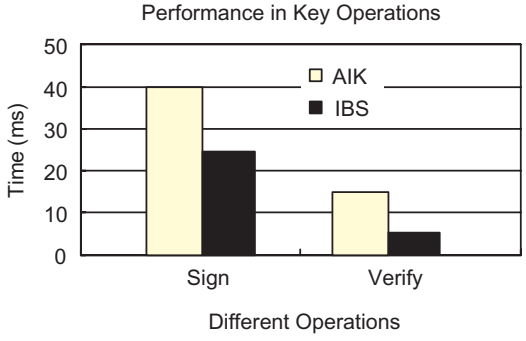

Figure 15. Attestation operations

7 Related Work

M-payment security has been studied extensively in the literature [19, 29, 18, 20, 21]. Herzberg analyzes the

security requirements of mobile payments and proposes some examples of different types of mobile payments, such as secure transaction request generation and location-based mobile payments [19]. Zhang *et al.* construct a biometric-enabled payment system to ensure fair-exchange, user anonymity, and privacy protection [29]. Hashemi and Soroush propose a secure payment protocol considering the restrictions of mobile networks in developing countries [18]. Kungpisdan *et al.* propose a simple and powerful credit-card payment protocol for wireless networks, which implements a secure cryptographic technique that works well under their protocol [20]. Also, Kungpisdan *et al.* propose a secure account-based payment protocol which is suitable for wireless networks, and employs symmetric-key operations which require lower computations for all engaging parties, c.f., [21]. Similar to our approach, E-EMV [13] demonstrates how Trusted Computing technology can be used to emulate EMV for use in Internet-based Card Not Present(CNP) transactions. However, this scheme only focuses on the authentication and the key negotiation process using cryptographic functions provided by the *TPM*.

Our presented architectures and schemes are orthogonal to all of these schemes and could work very well alomg with these approaches.

Another line of work focuses on securing e-wallets. In [23], Mjolsnes and Rong propose a generalization of electronic wallets to enable account-based payments. Users achieve payment mobility and independence from both terminals and payment service providers while maintaining secure access to their payment authorization credentials. Ebringer *et al.* propose a parasitic authentication, thus offering security for handheld computers [16]. With parasitic authentication, users can temporarily delegate their responsibility for authorizing a transaction to a small, portable secondary device. In this work, four systems are proposed by considering different levels of computational power provided by different mobile devices. Leung *et al.* propose a simple approach to determine fraud and to resolve disputes without the use of digital signatures. Towards this, the concept of an atomic transaction for an e-wallet account is realized to install a fraud detection module [22].

Molar *et al.* provide a secure RFID solution with remote attestation [24]. They fully use TC technologies to secure RFID. The key-issuing authority can demand such a proof before releasing shared secrets to the reader. In addition, sealed storage is used to protect secrets even if the reader is compromised. Through sealing the configuration of RFID machines, such as privacy policies which define the RFID reader's permissions, the privacy goal is achieved in a similar way employed by our basic payment scheme. However, they do not consider the overhead which may be introduced by their architecture, such as *AIK* management and transmission. In our improved payment schemes, we fully utilize the mobile network infrastructure to improve the attestation performance without sacrificing security achievements.

Moreover, Sailer *et al* present a platform integrity solution for measurement and attestation [25]. Gallery and Mitchell propose some secure mobile applications using TC, such as SIM-Lock and software downloading [17]. Yang *et al.* provide a secure inference control model by the TC paradigm in [28] and propose also privacy-preserving credentials using functions provided by TC, c.f., [27]. Most of these solutions with TC provide a complete attestation of whole system. Many researchers have argued that attestation of a complete platform including a general-purpose operating system is not practical. However, in phone devices, the system is relatively stable, e.g., a user has only partial permissions to run simple untrusted applications, and most system software is not user configurable. Therefore, for simplicity, we only include in our solution the attestation of critical platform and mobile payment-related components. This of course only attests the mobile payment application and the respective runtime environment (e.g., the JVM) of our prototype.

8 Conclusion and Future Work

In this paper we proposed a secure mobile payment scheme using trusted computing (TC) technology. In our proposed architecture we presented a platform integrity protection solution for mobile payment via NFC. Our scheme addresses the unresolved security challenges of mobile payment, including platform integrity verification and user privacy protection. Our secure payment scheme includes secure software downloading, secure payment application initialization, and secure payment transaction. In order to improve the efficiency, flexibility and performance of our payment scheme, we proposed two enhanced payment schemes, utilizing an IBS scheme and an attestation cache. We have implemented a prototype system, studied the feasibility and performance of our scheme and compared the performance in the different payment schemes. The experimental results show that our scheme is efficient and effective to achieve the security target. A simpler m-payment security model with different levels of security requirements will be studied in the future. Furthermore, since our scheme is very general for mobile payment applications, we will investigate to enhance the security of existing payment schemes with our solution.

References

[1] Credit card processing with microsoft dynamics rms, http://www.microsoft.com/dynamics/rms/product/merchantservices.mspx.

[2] Experience implementing DAA, http://sourceforge.net/mailarchive/forum.php?thread_name =da7b3ce3070716222215bc30306q2347604bf02b2e98.

[3] F-secure, http://www.f-secure.com/weblog/archives/archive-042006.html.

[4] Mobile trusted module, https://www.trustedcomputinggroup.org/groups/mobile.

[5] NFC forum, http://www.nfc-forum.org.

[6] The open-source TCG software stack, http://sourceforge.net/projects/trousers.

[7] Shamus software ltd. miracl. http://www.shamus.ie/.

[8] Software-based TPM emulator, http://developer.berlios.de/projects/tpm-emulator.

[9] Trolltech, http://trolltech.com.

[10] Trusted computing for the java platform, http://trustedjava.sourceforge.net/.

[11] Trusted computing group, https://www.trustedcomputinggroup.org.

[12] N. Asokan, P. Janson, M. Steiner, and M. Waider. The state of the art in electronic payment systems. *Computer*, 30(9):28–35, 1997.

[13] S. Balfe and K. Paterson. E-EMV: Emulating EMV for internet payments using trusted computing technology. *Technical Report RHUL-MA-2006-10, Department of Mathematics, Royal Holloway, University of London*, 2006.

[14] D. Boneh and M. Franklin. Identity based encryption from the weil pairing. *SIAM J. of Computing*, 32(3):586–615, 2003.

[15] T. Dierks and C. Allen. The TLS protocol version 1.0. *RFC 2246*, 1999.

[16] T. Ebringer, P. Thorne, and Y. Zheng. Parasitic authentication to protect your e-wallet. *Computer*, 33(10):54–60, 2000.

[17] E. Gallery and C. Mitchell. Trusted mobile platforms. In *Proceedings of Foundations of Security Analysis and Design IV (FOSAD)*, 2006.

[18] M. Hashemi and E. Soroush. A secure m-payment protocol for mobile devices. In *Proceedings of Canadian Conference on Electrical and Computer Engineering*, 2006.

[19] A. Herzberg. Payments and banking with mobile personal devices. *Communications of the ACM*, 46(5):53–58, 2003.

[20] S. Kungpisdan, B. Srinivasan, and P. Le. Lightweight mobile credit-card payment protocol. In *Proceedings of Lightweight Mobile Credit-Card Payment Protocol*, 2003.

[21] S. Kungpisdan, B. Srinivasan, and P. Le. A secure account-based mobile payment protocol. In *Proceedings of International Conference on Information Technology: Coding and Computing*, 2004.

[22] A. Leung, Z. Yan, and S. Fong. On designing a flexible e-payment system with fraud detection capability. In *Proceedings of the IEEE International Conference on E-Commerce Technology*, 2004.

[23] S. Mjolsnes and C. Rong. On-line e-wallet system with decentralized credential keepers. *Mobile Networks and Applications*, 8(1):87–99, 2003.

[24] D. Molnar, A. Soppera, and D. Wagner. Privacy for RFID through trusted computing. In *Proceedings of ACM workshop on Privacy in the electronic society*, 2005.

[25] R. Sailer, X. Zhang, T. Jaeger, and L. Doorn. Design and implementation of a tcg-based integrity measurement architecture. In *Proceedings of USENIX Security Symposium*, 2004.

[26] U. Varshney. Mobile payments. *Computer*, 35(12):120–121, 2002.

[27] Y. Yang, R. Deng, and F. Bao. Privacy-preserving credentials upon trusted computing augmented servers. In *Proceedings of Third International Conference on Information Security Practice and Experience (ISPEC)*, 2007.

[28] Y. Yang, Y. Li, and R. Deng. New paradigm of inference control with trusted computing. In *Proceedings of Data and Applications Security (DBSec)*, 2007.

[29] Q. Zhang, K. Mayes, and K. Markantonakis. A user-centric m-payment solution. In *Proceedings of the Second International Conference on Mobile Technology, Applications and Systems*, 2005.

[30] A. Zmijewska. Evaluating wireless technologies in mobile payments - a customer centric. In *Proceedings of International Conference on Electronic Commerce*, 2005.