Modelling, Design, and Analysis of Secure Mobile Payment Systems

Supakorn Kungpisdan

A Thesis Submitted for Fulfillment

of the Requirements for the Degree of

Doctor of Philosophy

Faculty of Information Technology
Monash University

Abstract

Mobile payment allows users to perform payment transactions through their mobile devices. However, it brings up many emerging issues regarding security and performance of mobile payment systems that can be classified into at least two main problems. The first problem comes from the limitations of wireless environments that are primarily from mobile devices which have limited system resources and from wireless networks which have high connection cost, low bandwidth, and low reliability. In particular, a mobile user may not be able to efficiently performing highly secure transactions, which require high computational cryptographic operations, over the wireless network with the above characteristics. The second problem is the lack of sufficient security of existing mobile payment systems, mainly due to improper protocol design and the deployment of lightweight cryptographic operations which lead to the lack of important transaction security properties. Such problems have motivated the research conducted in this thesis.

The purpose of this thesis is to propose methods to enable practical and secure mobile payment. The results obtained from this thesis may serve as a basis for protocol designers and system implementers to design and implement secure mobile payment systems and to analyze their existing mobile payment systems. The research conducted in this thesis focuses on three different levels of reasoning and securing mobile payment: *formal model*, *framework*, and *protocol*.

We first propose a formal model for a practical and secure mobile payment system. In this model, we formalize interactions among engaging parties and properties to be satisfied by the system including goals and requirements for payment transactions, transaction security properties, and trust relationships among parties. We generalize transaction performance and define the transaction performance which is acceptable by engaging parties. The proposed model can be seen as a guideline for designing and implementing practical and secure mobile payment frameworks and protocols for both account-based and token-based payment.

At the framework level, we investigate the problems of existing mobile payment frameworks. Then, we introduce a framework that not only overcomes the limitations of wireless environments, but also solves the problems of the existing frameworks. Particularly, a traditional fixed-network payment protocol is well operated in our framework, even more efficiently if a payment protocol specifically designed for wireless environments is applied. In addition, we show that the proposed framework can be captured by the proposed formal model.

At the protocol level, we propose a lightweight, yet secure cryptographic technique. This technique not only reduces the computation at engaging parties, especially at mobile users, but also satisfies the transaction security properties including the trust relationships among engaging parties stated in the proposed formal model. We then introduce two account-based mobile payment protocols which deploy the proposed technique. We develop a prototype of one of the proposed protocols to demonstrate its practicability as a real world application. The results from the implementation show that the implemented protocol itself operates well in wireless environments, yet has better transaction performance if the proposed mobile payment framework is applied to it. We also demonstrate that both of the proposed protocols have better transaction performance than existing protocols.

To show that the proposed framework and protocols satisfy the formal model, we develop a formal logic for analyzing them and successfully prove that they satisfy the goals and requirements for payment transactions and the transaction security properties, stated in the formal model. Combining with the above analysis results, it can be concluded that either a payment system based on the proposed framework deploying an existing payment protocol or a payment system based on the proposed protocol operating on an existing framework is considered as a practical and secure mobile payment system because it satisfies all the required properties stated in the model. In addition, we show that the proposed logic is general in that it is able to analyze any kinds of electronic commerce protocols including mobile payment protocols.

To enhance the security of the proposed protocols, we introduce a limited-use key generation technique which eliminates the need of long-term shared key distribution among engaging parties prior to each transaction. We then apply the proposed key generation technique to the proposed protocols and discuss its potential applications to other kinds of Internet applications.

Finally, to emphasize the generality of our mobile payment model, we propose a (token-based) micropayment protocol for wireless environments that satisfies the proposed model. The protocol deploys the proposed lightweight cryptographic technique to enhance its transaction security. The proposed protocol is prepaid-based, yet extensible to postpaid-based micropayment. This results in a general framework for wireless micropayment. We then demonstrate that our micropayment protocol is more secure and has better transaction performance compared to existing micropayment protocols.

Acknowledgements

The story has begun when I started to study Master of Computer Engineering at King Mongkut's University of Technology Thonburi, Thailand. At the time I did not have any interest in computer science study in particular. Surprisingly, the only area of study that fascinated me was mathematical logic. I found the logic as such a great tool to train my brain in logical reasoning. It was again the logic that led my interest to computer security which has become the main focus of my PhD study.

I would like to first and foremost thank my supervisors, Professor Bala Srinivasan and Dr. Phu Dung Le, for their guidance and support for my PhD study, from School of Network Computing to School of Computer Science and Software Engineering. I am very grateful for their care of both my study and personal life. They gave me a very warm welcome since the first steps at Monash University. Without their supervision, I would not be able to achieve my aspiration.

I thank Sunny Toh for his contribution on the implementation presented in chapter 6. Also, I thank Thalerngsak (Guy) Kijthaweesinpoon for being my brother and best friend in Melbourne. He helped me get through so many things. I specially thank Guy and Akamon Kunkongkapun for having lunch with me almost every day at Monash University, Caulfield campus. What we called "*same*" will be remembered.

I express my thanks to everyone at School of Computer Science and Software Engineering and School of Network Computing for support for everything including the scholarships provided during my PhD candidature.

Gratefully, I thank my parents for their continuous support, care, and encouragement. They have always been supporting me on everything, whatever I decide to do, whenever I needed help, their suggestion and guidance always helped me to sort things out easily. I dedicate each and every of my success to them. I thank my brother, Kan, and sister, Pamila, for taking care of Mom and Dad while I am away from them.

I would like to give my final thank to my lovely girlfriend, Kaewthida Kaewamporn for her genuine love and companionship. She is the first and only person in addition to my parents who dedicates life and future for me. She has always been by my side, being my support. I'm deeply grateful for that.

Declaration

In accordance with Monash University Doctorate Regulation 17 / Doctor of Philosophy and Master of Philosophy (MPhil) regulations the following declarations are made:

I hereby declare that this thesis contains no material which has been accepted for the award of any other degree or diploma at any university or equivalent institution and that, to the best of my knowledge and belief, this thesis contains no material previously published or written by another person, except where due reference is made in the text of the thesis.

Supakorn Kungpisdan

February 16, 2005

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Mobile Commerce

Internet technology offers extensive ranges of services such as electronic mails, file transfers, etc., and one of the most popular services offered on the Internet is *"Electronic Commerce"* (*or e-commerce*). E-commerce is becoming bigger technological wave that has changed the way by which business is being conducted.

Two main areas in which e-commerce grew significantly in recent years are Internet banking and conducting business on the Internet [Gho98]. With Internet banking, the way customers make use of banking services has changed. They do not have to go to ATM (Automatic Teller Machine) terminals or stay in-line at a bank branch to withdraw or transfer money between accounts, but simply log on to a bank's website which provides Internet banking services including withdrawing money from the customers' accounts. Although the customers cannot get physical cash in their hands, they are able to transfer money to electronic cards and bring them to purchase goods or services at stores. Moreover, the customers are able to pay bills or schedule monthly bill payments by using the Internet banking services.

Regarding conducting business on the Internet, basically, e-commerce simulates and enhances traditional ways that people conduct business or communicate to one another into electronic manners. For example, electronic mails (e-mails) replace mailing services in that people do not have to wait overnight for delivery of letters, but only in minutes electronically. Most of the features of e-mails are similar to paper mails such as (digital) signatures of senders, timestamps, or returned mails in case the mails cannot be received by the recipients. In addition to the time reduction, e-mails reduce the cost for document and delivery.

According to business transactions, many e-commerce websites enable their customers to browse for goods and services offered in their virtual stores remotely from the customers' personal computers. Not only physical goods, such as books (e.g. `www.amazon.com`) or laptop computers (e.g. `www.dell.com`), are offered, but electronic goods, such as music, digital images, video clips, or electronic novels, are also available. Customers simply select desired products or services and pay for them by credit-cards or electronic cash cards. More importantly, these virtual stores are open 24 hours a day, 7 days a week.

Recently, e-commerce transactions can be performed on the move. The emergence of wireless communication technology offers the ability to access the Internet in order to perform e-commerce transactions through mobile devices e.g. cellular phones, PDAs (Personal Digital Assistants), or laptop computers. Such mobile devices are connected to the Internet via modems or wireless network adapters. This greatly offers convenience to users to perform e-commerce transactions from distance at any time. Performing e-commerce transactions where at least one engaging participant in an e-commerce system is a mobile user is called *"Mobile Commerce"* (or *m-commerce*). Recently, m-commerce has been receiving attention considerably and has high growth rate. A recent study from IDC, a market intelligence and advisory firm, forecasts that the US m-commerce market will reach US$58.4 billion by 2007 compared to US$127 million in 2001 [IDC].

Compared to conducting e-commerce over fixed-networks, m-commerce offers many advantages [SGK02] which include:

- Users in a m-commerce system can access the Internet to perform e-commerce transactions through their mobile devices in real time at any place.
- Mobile devices have smaller size and lower weight than personal computers (PCs). Thus, they can be easily brought with users to anywhere.
- Mobile devices are personally used so that they can be personalized to fit the needs of particular users.

1.2 Mobile Payment

Although e-commerce is not all about fund transfer, electronic payment (or e-payment), such as credit-card payment over the Internet, is still one of the most popular e-commerce applications. In other words, e-payment is one of the crucial parts of an e-commerce transaction in that the e-commerce transaction cannot complete without it. For example, an online book store which provides both electronic and physical books to its customers must have a supporting payment system available for its customers to transfer money to it. Therefore, each customer can complete the purchase which includes goods delivery (or commitment of goods delivery) and payment with the store in one transaction. Without the payment system provided, the customers are required to perform two sessions separately: one for the goods purchase and the other for the payment transaction. In particular, the payment transaction has to be performed by transferring money to the store's bank account directly.

E-payment is an alternative to traditional payment which has been generally performed physically by presenting a credit card to a merchant and signing on a payment slip as evidence of the payment transaction made to the merchant. As briefly described in the previous section that there are several kinds of e-payment methods, the most primitive payment method is fund transfer between bank accounts. According to this method, an e-commerce website sends an invoice to a customer via an e-mail and requests the customer to transfer the money as the payment for the requested goods or services to its bank account. After receiving the payment, the website sends the corresponding payment receipt via the customer's e-mail and delivers the goods to the customer. It can be seen that the customer commits the purchase to the website online, but the payment is performed off-line. Later on, e-payment has been developed to fully online transactions such as credit cards, electronic checks, electronic cash, and micropayment.

The most popular type of e-payment methods is credit-card payment. According to this method, after selecting preferred goods from an online store, a customer can make a payment via a supporting credit-card payment system provided by the store by simply filling in her credit-card number and relevant information, such as date of birth or billing address, for authentication and payment authorization purposes. Such information is transferred to the customer's credit-card company to check for credit availability. If the request has been approved, the goods (in case of electronic goods) and corresponding receipt of payment are transferred to the customer shortly. Mostly, this kind of payment systems is operated using 128-bit SSL (Secure Socket Layer) protocol [FKK96].

Credit-card payment systems such as SET (Secure Electronic Transaction) [Mas97] provide more secure payment transactions than SSL-based payment schemes [FKK96]. In SET, in addition to holding a valid credit card, a customer needs to install SET payment software (called SET wallet) on her computer. After browsing for goods or services from a SET supporting online store, the SET wallet installed on the customer's computer is activated. After filling necessary payment information, the SET wallet performs highly secure cryptographic operations e.g. public-key cryptographic operations, to generate a purchase request. This request is transferred to the store and the customer's credit card company for payment authorization. After getting an approval, the requested amount is transferred from the customer's account to the store's account and the customer then receives the requested goods and the corresponding payment receipt at the end of the transaction.

Credit-card payment seems to be a simple method to make a payment for goods or services on the Internet because many people have credit cards and regularly use them to purchase goods or services in physical stores. However, the credit-card payment systems have high operational cost, especially at the merchant side. As a result, credit-card payment is not suitable for low-valued payment transactions.

Alternatively, a payment method that is suitable for low-valued transactions is called *"Micropayment"*. Most micropayment systems deploy low computational cryptographic operations and simple message passing in order to reduce operational costs. The examples of micropayment systems are Millicent [GMAG95], PayWord [RS96], and PayFair [Yen01].

Electronic payment in wireless environments introduces the term *"Mobile Payment"* which is defined as interactions among engaging parties in a payment system regarding a payment transaction where at least one engaging party is a mobile user. With mobile payment, obviously, customers can purchase electronic books from an online publisher, that has the system supporting the payment from mobile devices, while they are on the move.

Due to the fact that mobile payment represents e-payment, previously performed in fixed environments, in wireless environments, it offers the same services as that offered by e-payment. However, due to the constraints of wireless environments, low-valued payment methods, such as micropayment, which have lightweight operations and low operational cost are likely to be more suitable for wireless environments than other methods. The constraints of wireless environments will be discussed in details in the next section.

Electronic payment, including mobile payment, plays an important role in e-commerce in that it is relevant to fund transfer among engaging parties after having an agreement to purchase or sell products or services. It must be performed in a secure manner. Moreover, the security of electronic payment system is also one of the most concerns for customers to make online payment with online stores.

1.3 Security and Limitations of Mobile Payment Systems

Generally, two main reasons explain why securing mobile payment systems is not accomplished: limitations of wireless environments and security of the mobile payment systems themselves as shown in sections 1.3.1 and 1.3.2, respectively.

1.3.1 Limitations of Wireless Environments

Performing payment transactions in wireless environments mainly suffers from resource limitations of mobile devices and characteristics of wireless networks [RdS98, KSL03a, WC01].

Resource Limitations of Mobile Devices

Mobile devices have the following limitations:

- Computational capability of their processors is comparatively lower than that of personal computer (PCs).

- They are operated using battery power compared to electric power in PCs. Therefore, they can stay operated for shorter period than PCs.
- They have limited storage which affects available cryptographic algorithms applied to them.

A mobile device with the above limitations is not capable of performing high computational cryptographic operations such as public-key operations which are used in a fixed-network device such as a PC. Due to the low computational capability of mobile devices, completing a payment transaction on a mobile device takes longer period of time than that on a PC which has higher processing capability. Moreover, public key operations are required to have certificate verification processes which require storage on each mobile device to store public-key certificates.

Although recently, mobile devices with high computational capability such as smart phones or powerful, wireless-enabled PDAs, have been launched to the market, they are still unattracted by users. The research conducted by IDC [ITF] states that estimated 650 million mobile phones are expected to be sold in 2004. Among these units, 30.05% of them are WAP-enabled phones (around 195 million phones) [ePa04b]. Whereas the number of high computational capability mobile devices such as smart phones and PDAs to be sold in 2004 are less than WAP-enabled phones; only 17.6 million smart phones are expected to be sold in 2004 [ITF] and 11.367 million PDAs are to be sold and expected to decrease to 11.251 million units in 2005 [Dat04].

Characteristics of Wireless Networks

Wireless networks have the following characteristics:

- Wireless networks have lower bandwidth than fixed networks.

- Network connections over wireless networks are less reliable since packet losses occur more frequently than that of fixed networks. Packets need to be retransmitted which may result in high latency.

- Connection cost of wireless networks is higher compared to that of fixed networks.

- Data transmitted over wireless networks is easily eavesdropped.

From the above limitations, mainly due to poor performance, performing payment transactions over wireless networks is time-consuming. Moreover, performing payment transactions on low computational capability mobile devices will spend longer time to complete each transaction. As the connection cost of the communications over wireless networks is much higher than that over fixed networks, performing payment transactions over wireless networks using such mobile devices will charge users a large amount of money on their bills. In addition, due to the fact that the data transmitted over the wireless networks is easily eavesdropped, this can be prevented by applying highly secure cryptographic techniques such as public-key operations. However, Such operations require high computational capability devices and high-speed wireless networks that may incur high cost for users.

1.3.2 Security vs Transaction Performance of Mobile Payment Systems

Performing electronic payment transactions over wireless networks raises concerns about security of the underlying payment systems. Ideally, both traditional and wireless Internet should serve all applications, including making payment, with the same level of security. Moreover, mobile payment applications should be compatible with existing infrastructure of traditional electronic payment applications so that the existing infrastructure can continue to operate.

However, as discussed in the previous section, performing payment transactions in wireless environments suffers from a number of limitations. A possible solution is to replace high computational cryptographic operations applied to the underlying payment protocol with the lower ones, e.g. replacing public-key cryptographic operations with symmetric-key operations and hash functions, especially the ones operated on a mobile device [MB01, PBD01, SV97]. This can reduce computational and communication load at the mobile users. However, different kinds of cryptographic operations provide different levels of security. Public-key operations provide non-repudiation property whereas symmetric-key operations do not [KSL03a]. Hence, deploying a symmetric-key based payment protocol in the payment system results in the lack of the non-repudiation property. As a result, enabling mobile payment seems to be a tradeoff between its transaction performance and security.

1.4 Motivations of the Thesis

There have been several efforts to secure mobile payment either by migrating existing e-payment protocols for fixed networks to secure payment transactions over wireless ones [RdS98, WLY99, YSWO00] or designing new payment protocols which are specific to wireless environments [ZK02, PBD01, Tha00], ranging from high-valued (thousands of dollars) transactions by credit-card or electronic check protocols to low-valued transactions (a few cents) by micro-payment protocols.

To secure high-valued payment transactions, credit-card payment protocols, such as SET [Mas97] and iKP [BGH$^+$00] protocols, and electronic-check protocols [And98, DK01] were proposed. These protocols are based on public-key infrastructure (PKI) that are not efficiently applied to wireless networks.

To deploy PKI in a wireless environment, a client needs to perform high computational cryptographic operations, and the client's mobile device is required to have storage for a number of public-key certificates. Moreover, during each transaction, the certificate sent to the client has to be verified by a *Certificate Authority* (CA) which is located in a fixed network. Although, the pointer to the location of full certificate, typically a URL, can be used to reduce the size of certificates stored on a mobile device [TR02], PKI-based payment protocols still need to have additional communication passes.

Alternatively, the migration of existing payment protocols for fixed networks to wireless ones was proposed by the deployments of proxy servers [WSZ01, AG03, JRFH02] and mobile agents [RdS98, WLY99, YSWO00]. We name the payment frameworks for such migration approaches as *proxy-based* and *agent-based* frameworks, respectively. In a *proxy-based* payment system, a proxy server located in a fixed network performs transactions on behalf of the clients. The proxy server contains all payment-related information of the clients. Thus, the clients' storage requirement is reduced. However, the clients are required to fully trust the proxy server because their sensitive information, including private keys, is stored on the proxy server.

In an *agent-based* payment system [RdS98, WLY99, YSWO00], a mobile agent act as a client's assistant on performing transactions. Generally, the mobile agent is defined as a software element (program, procedure, object, thread, etc.) owned by a user. It is capable of migrating from one computer to another in order to execute a set of activities on behalf of its owner [RdS98]. The concept of mobile agent has also been applied to mobile payment scenario in order to overcome the limitations of wireless environments, particularly to increase transaction performance [RdS98, YSWO00]. Several agent-based payment systems based on existing fixed-network payment protocols, mainly SET protocol [Mas97], have been proposed [RdS98, WLY99, YSWO00]. These systems employ mobile agents to perform payment transactions on the client's behalf. Thus, each client needs to connect to the Internet for short periods during the entire transaction. This greatly reduces connection cost for the clients. However, such systems are still susceptible to attacks due to the problem of key generation in an insecure environment [WLY99].

To secure low-valued transactions, a number of micropayment protocols have been proposed [GMAG95, RS96, PBD01, Yen01]. They deploy low computational cryptographic operations in order to minimize their operational cost compared to that of credit-card payment protocols. Applying micropayment protocols to wireless environments, most of them do not need mobile agents or proxy servers because their computational and communication load is comparatively low. We classify the payment framework for this kind of payment systems as *non proxy-based* framework.

Several micropayment protocols have been proposed [Yen01, PBD01, RS96] based on various kinds of cryptographic operations. On one hand, PayWord [RS96] deploys public-key operations that have high computational load, especially at mobile users. On the other hand, symmetric-key based micropayment protocols [Yen01, PBD01], which deploy symmetric-key operations and hash functions, require lower client's computation. However, due to the nature symmetric-key operations, such micropayment protocols lack non-repudiation property [KSL03a] which is one of the most important properties for financial transactions [KP02]. Moreover, in order to minimize the party's computation by reducing the number of cryptographic operations applied to the protocol, some private information, which should be known only by the appropriate parties, is transmitted in cleartext during the transaction as in [Yen01] and the assumptions of trust relationships among parties are too-strong. For example, in [PBD01], a payment system provider knows all of its clients' secrets. Either the lack of the non-repudiation property or the exposure of the clients' private information allows the payment system provider to impersonate as its clients to perform payment transactions.

In any payment systems, the following transaction security properties must be satisfied [Ahu96, PBD01]:

- **Party Authentication**: each engaging party in the system must be able to authenticate the party whom she is communicating with.
- **Transaction Privacy**: each engaging party must be able to ensure that the messages are not revealed to any unauthorized parties, but only to the intended recipient of the messages.
- **Transaction Integrity**: each engaging party can ensure that the received messages are not altered during the transmission.
- **Non-repudiation of Transactions**: each engaging party cannot deny the transactions she has performed.

In addition to the fundamental properties described above, a secure payment system requires *"Accountability"* property. The accountability property is defined as the ability to show that the parties who engage in the system are responsible for the transaction related to them [KP02]. Generally, accountability is considered as one of the most important security properties of e-commerce protocols since they are relevant to fund transfer and goods ordering. Each engaging party must be able to prove the association with messages sent in the protocol that she is either the originator or the intended recipient of the message.

The accountability is considered as a *high-level* security property which covers all the above fundamental transaction security properties. To achieve the accountability property, all transaction security properties stated above must be satisfied. Thus, the accountability property is considered as the main security property to be used for the analysis in this thesis.

Several formal logics [Kai96, KN98, KP02] were proposed to analyze e-commerce protocols on the accountability property. Kailar [Kai96] proposed a logic based on belief logic [BAN90]. Kailar's logic provides reasoning about the accountability of signed plain messages e.g. $\{X\}_{K_P^{-1}}$, where X stands for a message and K_P^{-1} stands for the private key of a party P. However, Kailar's logic is inadequate to analyze e-commerce protocols which are composed of complex cryptographic messages e.g. signed encrypted and/or hashed messages e.g. $\{\{X\}_{K_R}\}_{K_P^{-1}}$ or $\{h(X)\}_{K_P^{-1}}$, where X stands for a message, K_R stands for the public key of a party R, K_P^{-1} stands for the private key of a party P, and $h(M)$ stands for the hash value of a message M. Moreover, Kailar's logic does not reason about a verifier who acts as a dispute resolver [KP02]. Note that the reasoning about the verifier is very important because we need to state about the information that a prover can reveal to the verifier as proof evidence. Kessler *et al.* (KN) [KN98] proposed a logic which offers reasoning about the verifier's belief. However, KN's logic does not provide reasoning about proving without revealing secret information to the verifier. Moreover, its reasoning about hash function is too strong in that it allows the verifier to infer the input of a hash function without possessing the input itself [KP02].

Kungpisdan *et al.* (KP) [KP02] proposed a modification of KN's logic for reasoning about the accountability of SET [Mas97] and iKP [BGH$^+$00] protocols. KP's logic can be used to analyze e-commerce protocols which are composed of complex cryptographic messages. Moreover, it provides reasoning about the verifier and captures the provability without revealing secret information to the verifier by using a prover's belief about the verifier's possession of information.

Normally, existing accountability logics [Kai96, KN98, KP02] can provide reasoning only about the accountability of asymmetric cryptographic messages based on the assumption that symmetric cryptographic messages cannot provide the accountability because the secret key for encrypting a message is shared among engaging parties. Thus, it is not able to specify the originator of symmetric-key encrypted message. Recently, several cryptographic protocols that are based on symmetric cryptography, yet are able to identify the originator of the message, have been proposed [Cim02, KSL03a, KSL04c]. Therefore, this makes the existing accountability logics, including KP's logic [KP02], not general in that they are incapable of analyzing the accountability of symmetric cryptographic messages [KSL04a].

As a result, there are several transaction performance and security issues relevant to mobile payment. The major problem still remains unsolved: the availability of practical and secure mobile payment systems.

1.5 Objectives of the Thesis

The objectives of this thesis are presented in three levels of reasoning and securing mobile payment: *formal model*, *framework*, and *protocol*.

- **Formal model:** we aim to develop a formal payment model for a practical and secure mobile payment system. The model defines characteristics and interactions among engaging parties, how the money is transferred, and important properties for a practical and secure mobile payment system.

- **Framework:** we aim to develop a payment framework which is suitable for wireless environments. The framework not only enhances transaction performance of a payment protocol operating on it, but also provides secure transactions. In addition, the proposed framework satisfies the proposed formal model.

- **Protocol:** at protocol level, our objectives are shown as follows:

  - Develop a secure cryptographic technique which not only increases transaction performance while applying it to a mobile payment protocol, but the protocol also satisfies transaction security properties stated in the proposed formal model.

- Design non proxy-based payment protocols for both account-based and token-based payment transactions. The proposed protocols satisfy the proposed mobile payment model by applying the proposed cryptographic technique to them. The proposed protocols offer both secure and practical mobile payment in that:
  - They are secure in that they satisfy transaction security properties including accountability property.
  - They are practical in that they have higher transaction performance than that of existing non proxy-based payment protocols when operating on low computational capability mobile devices.
  - Not only the proposed protocols have higher transaction performance and security than existing non proxy-based mobile payment protocols, but applying them to the proposed framework also offers higher performance and security than existing mobile payment systems.
- Validate the practicability of the proposed mobile payment protocols by implementing a mobile payment system based on the proposed protocols.
- Develop a formal logic for analyzing accountability property of mobile payment protocols that is able to analyze both symmetric and asymmetric cryptographic messages. The logic can be used to analyze accountability of e-commerce protocols.
- Analyze the proposed framework and protocols primarily by using the proposed formal logic to show that a payment system based on the proposed framework or protocols satisfies the proposed model.

1.6 Organization of the Thesis

Chapter 2 provides background of electronic payment systems. We present requirements for a practical and secure mobile payment system which can be used as a basis to evaluate mobile payment systems. We overview existing payment systems for fixed networks and discuss their problems and limitations when applying them to wireless environments. Moreover, we present existing frameworks for enabling mobile payment including their emerging problems.

In chapter 3, we propose a formal model for electronic payment systems which provides reasoning about interactions among engaging parties. We define goals of performing a payment transaction at design stage and the requirements to complete the transaction at implementation stage. Then, we formalize transaction security properties and transaction performance. Finally, we define the notion of a practical and secure mobile payment system and provide a guideline to evaluate a mobile payment system based on the proposed model.

Chapter 4 proposes a framework for mobile payment that is able to solve the security problems of the existing frameworks. We demonstrate the practical usefulness of the proposed framework by applying it to SET protocol. In addition, the analysis of the proposed SET framework regarding its security and performance is provided.

In chapter 5, we propose two non proxy, account-based mobile payment protocols which deploy a secure cryptographic technique which satisfies the proposed mobile payment model. The proposed protocols not only reduce the computation at all engaging parties, especially at the client, but also provide higher security compared to existing account-based payment protocols.

Chapter 6 presents an implementation of KSLv1 protocol, one of the proposed account-based mobile payment protocols, to show that our proposed protocols are applicable to wireless environments. We set up a system where a client, using a PDA, performs transactions to a merchant over a wireless network. The results of the implementation including related issues are provided.

In chapter 7, we examine limitations of the deployment of existing accountability logics for analyzing mobile payment protocols. We then propose a formal logic that is able to analyze accountability property of mobile payment protocols and show that the proposed framework and protocols satisfy transaction security properties and achieve the goals and the party's requirements for payment transactions defined in the proposed mobile payment model. We also show that, the combination of analysis results from the logical analysis and from the previous chapters shows that the payment system based on the proposed framework or protocols satisfies the proposed mobile payment model.

Chapter 8 investigates security issues regarding the applications of shared secrets in Internet transactions. We point out the problems of existing techniques for securing shared secret transfer during transactions. Then, we propose a key generation technique that solves the problems mentioned above. We apply the proposed technique to our proposed protocols to enhance their security. We also discuss its possible use to other kinds of Internet applications.

In chapter 9, we apply the cryptographic technique applied to the proposed account-based payment protocols in chapter 5 to secure token-based mobile payment by introducing a micropayment protocol for wireless networks. The proposed micropayment protocol satisfies the proposed mobile payment model. It is shown that the proposed mobile payment model is general in that it can capture both account-based and token-based payment transactions. The proposed protocol is based on prepaid micropayment, yet easily extended to postpaid one. It therefore results in a general wireless micropayment protocol.

Finally, chapter 10 provides conclusion of the thesis and discusses future research.

1.7 Basic Notations

The following general notations will be used in the thesis:

- $\{K_P, K_P^{-1}\}$: the set of public/private key pair of a party P.
- $Cert_P$: the certificate of a party P which contains the identity and the public key of the party P $\{ID_P, K_P\}$.
- $\{M\}_K$: a message M symmetrically encrypted with a shared key K.
- $\{M\}_{K_P}$: a message M encrypted with the public key of a party P.
- $\{M\}_{K_P^{-1}}$: a message M signed with the private key of a party P.
- $h(X)$: the hash value of a message X.
- $MAC(X, K)$: the Message Authentication Code (MAC) of a message X with a key K.
- $\langle X \rangle_K$: a message X applied with a single-key cryptographic operation with the key K. $\langle X \rangle_K$ can be symmetric-key encryption $\{X\}_K$, MAC, or $h(K)$.

Chapter 2

Overview of Mobile Payment

2.1 Requirements for a Practical and Secure Mobile Payment System

Payment is generally known as interactions among engaging parties regarding fund transfer. A buyer purchases products or services from a seller by exchanging a form of money with the products or services with the seller. This form of money has evolved from physical notes and coins to some other forms e.g. checks or credit cards. Nowadays, the payment process has been developed to be performed over the Internet. Only payment-related information or electronic tokens representing physical money are transferred from a payer to intended payee over the Internet. The recent development of mobile communication technology offering the ability to send data over wireless networks enables mobile users to perform another type of payment transactions called "*Mobile Payment*".

Mobile payment is not just an alternative of electronic payment. The nature of wireless environments leads to a number of emerging issues regarding transaction security and performance to mobile payment. On one hand, the data transmitted over a wireless network is easily eavesdropped. It is necessary to have a secure communication channel established over the wireless network.

This can be achieved primarily by deploying cryptographic operations.

On the other hand, several factors affect transaction performance of the payment system. Low computational capability mobile devices are incapable of performing high computational cryptographic operations such as public-key operations. Completing a transaction with high computational operations on such devices takes a lot of time which incurs high connection cost for mobile users. However, the lack of applying highly secure cryptographic operations results in vulnerability to attacks.

In particular, two main requirements must be accomplished when designing a practical and secure mobile payment system. Firstly, mobile users require a payment system in which they feel secure to perform transactions. Payment-related information should be transferred from a payer to an intended payee in a secure manner without being revealed to any unauthorized parties.

Secondly, mobile users require a payment system that they can perform transactions efficiently in that:

- A payment transaction can be completed within a limited amount of time which is acceptable by users in terms of operational cost and user satisfaction.
- A payment transaction can be performed on limited capability mobile devices.

To enable practical and secure mobile payment, both the migration of existing fixed-network payment systems [WSZ01, JRFH02, RdS98, WLY99] and the payment systems which are specifically designed for wireless networks [ZK02, Pay01, PBD01] have been proposed and implemented. In this chapter, we investigate these payment systems by focusing on their transaction security and performance issues.

This chapter is organized as follows. In section 2.2, a general electronic payment model is presented to describe characteristics and interactions among engaging parties in a payment system. In section 2.3, electronic payment systems are classified into two main categories: account-based and token-based payment systems. We present each of them in details and discuss their applicability to wireless environments. Section 2.4 reviews electronic payment frameworks that have been proposed to enable mobile payment including their classification. Moreover, we discuss their security and performance issues. In section 2.5, we briefly discuss a possible mobile payment solution which offers higher transaction performance and solves the security problems of the existing frameworks. Section 2.6 identifies another important security issue: securing account information transfer during payment transactions. We provide background and discuss problems and limitations of existing frameworks to secure account information transfer. Moreover, we discuss our possible solution in brief. Section 2.7 summarizes the chapter.

2.2 General Electronic Payment Model

In this section, an overview of mobile payment system is outlined in order to understand the characteristics and interactions among engaging parties in electronic payment systems.

IBM Research Division proposed a model for account-based payment systems [APASW98, Her01]. Abad-Peiro *et al.* [APASW98] argued that the model is credit-based but can be applied to other kinds of payment methods. The model is composed of 4 main engaging parties:

- **Client** is a party who requests to purchase products or services from a merchant.
- **Merchant** has products or services offered to the client.
- **Issuer** is the client's financial institution. It has the client's account established. Its task is to manage the client's account including fund

transfer.

- **Acquirer** is the merchant's financial institution. It manages the merchant's account including fund transfer.

An additional party called **Payment Gateway** acts as a medium between the issuer/acquirer at the banking private network side and the client/merchant at the Internet side for payment clearing purpose.

From [AHS98, Her01], the model introduces 3 primitive payment transactions (see Figure 2.1): *Payment*, *Value Subtraction*, and *Value Claim*. These primitive transactions define how the engaging parties communicate to one another regarding fund transfer. The details of each primitive transaction are given as follows:

- **Payment** is the interaction between the client and the merchant regarding payment ordering. *Payment* is considered as a bidirectional action in that the action that client sends a request to make a payment to the merchant is called *Payment Request*, whereas the action that the merchant sends a message as a payment receipt to confirm the goods delivery to the client is called *Payment Response.*

- **Value Subtraction** is the interaction between the client and the payment gateway (on behalf of the issuer). As well as *Payment*, *Value Subtraction* is a bidirectional action in that, the action that the client requests the payment gateway to deduct the money from her account is called *Value-Subtraction Request*, whereas the action that the payment gateway sends an acknowledgement of the client's request that notifies the client whether or not her request has been approved is called *Value-Subtraction Response.*

- **Value Claim** is the interaction between the merchant and the payment gateway (on behalf of the acquirer). *Value Claim* is considered as a bidirectional action in that the action that the merchant requests the payment gateway to deposit the requested amount to her account is called *Value-Claim Request*, whereas the acknowledgement sent from the payment gateway to the merchant regarding the merchant's request is called *Value-Claim Response*.

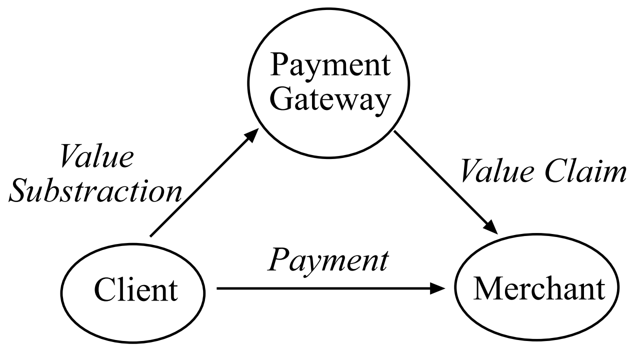


Figure 2.1: Primitive transactions

Figure 2.1 depicts the primitive transactions. Note that each arrow represents the direction to which the money is transferred. It can be seen that at the end of each transaction, the money is transferred from the client's account to the merchant's one. However, the actual money transfer is performed between the issuer and the acquirer over a banking private network. The tasks of both client and merchant are sending the requests/responses to engaging parties including each other. For example, the client does not transfer the requested amount to the merchant's account by herself, but actually sends the issuer the request to deduct the requested amount from her account and to transfer it to the acquirer. The acquirer then transfers the requested amount to the merchant's account.

At this stage, we have provided the necessary background of payment transactions. In the next section, we will present existing electronic payment systems and their classification. Moreover, we will discuss their security and performance issues in details.

2.3 Electronic Payment Systems

Existing payment systems can be categorized by considering how money transfer is organized into two categories: *account-based* and *token-based* electronic payment systems. The details of these systems are presented in sections 2.3.1 and 2.3.2, respectively.

2.3.1 Account-Based Electronic Payment Systems

In an account-based payment system, money is represented by account (or credit) balance in bank accounts and it is transferred among engaging parties' accounts established with their banks. Generally, account-based payment systems can be divided into two categories depending on payment clearing periods: *credit-based* and *debit-based* payment systems.

In a credit-based payment system, a client has credits authorized by her issuer to spend with merchants without being charged any cent from her account during purchases. The credits can be more than the current balance of the client's account depending on the account standing. To make a payment to a merchant, the client is required to have payment authorization from the issuer, and she is billed periodically e.g. monthly. The examples of this kind of payment systems are a credit-card payment scheme over SSL protocol [FKK96], the payment systems based on SET (Secure Electronic Transaction) protocol [Mas97] and iKP (Internet Key Protocol) [BGH$^+$00], VISA's 3D Secure [VIS02], Shamir's web-based payment system using disposable credit-card numbers [Sha02], Mu *et al.*'s system [MV02], Huming *et al.*'s payment system based on bank accounts [GCW01], and various kinds of electronic check payment systems [DK01, And98].

According to a debit-based payment system, a client is allowed to spend the money up to her current balance. The examples of debit-based payment systems are Visa Electron [VIS04] and EFTPOS [Com04, Rea89].

It can be noted that the client in both credit and debit-based payment systems is required to have payment authorization from her issuer in every transaction. This may incur high administrative cost, which is not suitable for low-valued transactions.

It can also be noted that the difference between credit-based and debit-based payment systems is only billing periods. Thus, an account-based payment system can operate both kinds of payment transactions by specifying the period when the money is deducted from the client's account. In particular, one account-based payment system can operate in two modes. That is, to work as a credit-based payment system, the client is billed after specified period and she has to pay for the transactions by the due date, whereas to work as a debit-based payment system, the money in the client's account is deducted immediately after the payment authorization for each transaction has been approved.

In any payment system, the payment protocol plays the most important role. In this thesis, we focus our consideration on the payment protocols and their applicability to wireless environments. In this section, we present SET [Mas97] and iKP [BGH$^+$00] protocols as examples of account-based payment protocols.

SET and iKP Protocols

SET protocol and its ancestor, iKP protocol, are the most well-known credit-card payment protocols. Three main parties are involved in both protocols: client, merchant, and payment gateway. They interact with one another over the Internet side. The client presents her credit card while purchasing goods or services from the merchant. The merchant is a party authorized by the payment gateway to be a merchant in the payment system. In SET and iKP protocols, the payment gateway can be a credit-card company or the client's issuer who offers its own credit-card service. Based on the payment model presented in section 2.2, the actual fund transfer is performed between the issuer and the acquirer over a banking private network.

Both SET and iKP protocols are credit-based in that the client is allowed to perform payment transactions up to credit limit specified by the issuer without paying any cents during the purchases. The client will be billed at the end of specified period. However, the client is required to have payment authorization from the issuer before making each payment. The purpose of the payment authorization is to check credit availability of the client. If the client has sufficient credits, her payment request will be approved.

As the ancestor of SET protocol, most of the structures of iKP protocol are similar to that of SET protocol. In SET protocol, all parties are required to possess their own public-key certificates, whereas iKP protocol consists of three versions depending on the number of certificates possessed by engaging parties: 1KP, 2KP, and 3KP. In 1KP, only the payment gateway is required to have its own certificate. Both the client and the merchant can authenticate themselves to the payment gateway and each other using PINs (Private Identification Numbers). In 2KP, only the client is not required to possess the certificate. Finally, 3KP protocol requires all engaging parties to possess their own certificates. The protocol descriptions of SET protocol are given as follows:

PinitReq:	**C**→**M**:	$InitialRequest$
PinitRes:	**M**→**C**:	$\{TID\}_{K_A^{-1}}, Cert_M, Cert_{PG}$
PReq:	**C**→**M**:	$OI, h(PI), \{h(OI), h(PI)\}_{K_1}, \{K_1\}_{K_C^{-1}},$
		$\{h(OI), PI\}_{K_2}, \{K_2\}_{K_{PG}}, Cert_C$
AuthReq:	**M**→**PG**:	$\{\{TID, Price, Date, h(OI), (OI), \{h(OI), h(PI)\}_{K_1},$
		$\{K_1\}_{K_C^{-1}}, \{h(OI), PI\}_{K_2}, \{K_2\}_{K_{PG}}\}_{K_M^{-1}}\}_{K_{PG}}$
AuthRes:	**PG**→**M**:	$\{\{TID, Price, Date, Yes/No\}_{K_{PG}^{-1}}\}_{K_M}$
PRes:	**M**→**C**:	$\{TID, Date, Yes/No\}_{K_M^{-1}}$

where,

- $\{C, M, PG, I, A\}$: the set of client, merchant, payment gateway, issuer, and acquirer, respectively.
- OI: order information. $OI = \{TID, h(OD, Price)\}$
- PI: payment information which contains credit-card information (CCI). $PI = \{TID, h(OD, Price), ID_M, Price, CCI\}$
- OD: order descriptions which contain goods descriptions.
- $Price$: amount and currency.
- TID: identity of transaction which contains the time and date of transaction $Date$.
- Yes/No: status of the transaction *approved/rejected.*
- $\{K_1, K_2\}$: the set of session keys generated by the client.

In this thesis, we focus on 3KP protocol because it provides the highest security in its family. The descriptions of 3KP protocol are demonstrated as follows.

Initiate: **C→M**: ID_C
Invoice: **M→C**: $ID_M, h(OI), Date$
Payment: **C→M**: $PI, \{PI, h(OI)\}_{K_C^{-1}}$
AuthReq: **M→PG**: $ID_M, h(OI), Date, h(OD), PI, \{h(OI)\}_{K_M^{-1}},$
$\{PI, h(OI)\}_{K_C^{-1}}$
AuthRes: **PG→M**: $AI, \{AI, h(OI)\}_{K_{PG}^{-1}}$
Confirm: **M→C**: $AI, \{h(OI)\}_{K_M^{-1}}, \{AI, h(OI)\}_{K_{PG}^{-1}}$

where,

- $PI = \{Price, h(OI), CCI\}_{K_{PG}}$
- $OI = \{TID, Price, ID_C, ID_M,$ *Date, h(OD)*$\}$
- AI: authorization information which contains Yes/No.

From the SET and iKP protocol descriptions, in **PinitReq** (or **Initiate**) and **PinitRes** (or **Invoice**), the client and the merchant exchange the information to start a payment session. In **PReq** (or **Payment**), the client starts making a payment to the merchant. The content in **PReq** (or **Payment**) consists of two parts for two purposes; one signed with the client's private key K_C^{-1} is considered as *Payment Request* (referred to the payment model described in section 2.2) to purchase goods or services from the merchant, and the other encrypted with the payment gateway's public key K_{PG} is considered as *Value-Subtraction Request* which is a request to the payment gateway to deduct the requested amount from the client's account. Note that, in SET protocol, the latter part is not directly encrypted with K_{PG}. It is first symmetric-encrypted with the session key K_2 generated by the client, and then the key K_2 is encrypted with K_{PG}.

After receiving **PReq** (or **Payment**), the merchant retrieves OI and then the goods descriptions (OD) and the requested amount ($Price$) in OI. The merchant sends **AuthReq** which is a request to the payment gateway to transfer the requested amount to the merchant's account. **AuthReq** is signed and encrypted with the merchant's private key K_M^{-1} and the payment gateway's public key K_{PG}, respectively. Note that **AuthReq** contains not only *Value-Claim Request* which is a request from the merchant to the payment gateway to transfer the requested amount to the merchant's account, but also the forwarded client's *Value-Subtraction Request.*

The payment gateway decrypts **AuthReq**, verifies the merchant's signature, and retrieves PI which contains the client's credit-card information. It consults the issuer and the acquirer about the validity and credit availability of the client's account. After the requested payment has been approved, the payment gateway sends **AuthRes** which represents the commitments to deduct the requested amount from the client's account (*Value-Subtraction Response*) and to transfer the money to the merchant's account (*Value-Claim Response*), respectively, to the merchant. **AuthRes** contains TID, price, and status of transaction *approved/rejected*.

The merchant decrypts the message and retrieves the result of her request. The merchant then sends **PRes** (or **Confirm**) which is a receipt of the payment and the commitment to deliver goods or services to the client. The client then verifies the merchant's signature and retrieves the result of her request.

Note that the analyses done by Herreweghen [Her01] and Kungpisdan *et al.* [KP02] have shown that SET protocol does not guarantee the client about the money deduction. That is, the message sent to the client in **PRes** contains no evidence from the issuer to confirm that the requested amount has been deducted from the client's account, whereas in iKP protocol, **Confirm** contains a part of the message signed with the payment gateway's private key K_{PG}^{-1}. It can be considered as a commitment from the payment gateway (on behalf of the issuer) to deduct the requested amount from the client's account.

SET and iKP protocols offer secure payment transactions. They satisfy all transaction security properties mentioned in section 1.4. However, they are not suitable for wireless environments due to a number of limitations which have been discussed in section 1.3. In particular, they are complex protocols which are based public-key infrastructure (PKI). In order to achieve sufficient security, all engaging parties are required to possess public-key certificates. In wireless environments, the implementation of PKI requires high capability mobile devices. In the case of wireless networks, in addition to higher operational cost compared to the fixed ones, the wireless networks also have limitations on lower bandwidth and reliability.

2.3.2 Token-Based Electronic Payment Systems

In a token-based payment system, electronic money (or payment token) represents physical money. A client exchanges physical money with payment tokens, e.g. electronic coins or electronic cash, with her issuer (by requesting the issuer to deduct the requested amount from her account) and uses them to pay for goods or services. A merchant collects the tokens and sends them to her acquirer to redeem the money (by the means of money transfer to the merchant's account). For simplicity, it is assumed that the issuer and the acquirer are the same party called *bank*. Referred to the payment model presented in section 2.2, the bank is represented by the payment gateway which performs the tasks of both issuer and acquirer.

According to the token-based payment system, the client is not required to have payment authorization from the bank in every transaction. It therefore has lower operational cost compared to that of account-based payment systems. This results in more suitability for low-valued transactions. The examples of the protocols that operate in token-based payment systems are electronic cash [AM00, YL01] and micropayment protocols [HSW96, GMAG95, PBD01, RS96, Yen01, SV97].

Token-based payment systems can be classified into two categories: *prepaid* and *postpaid* payment systems. In a prepaid payment system [Yen01, GMAG95], the client is required to purchase payment tokens by requesting the bank to deduct the money from her account to have the payment tokens in return. The client can spend the payment tokens with merchants for goods or services. The merchants collect the payment tokens and redeem the money from the bank.

In a postpaid payment system [RS96, HSW96, PBD01], the client is allowed by the bank to generate payment tokens by herself and spend them up to the credit limit specified to each merchant. The merchant collects the payment tokens and redeems the money from the bank. After a specified period, the client receives the bill as a result of the micropayment transactions she has performed.

In this section, we outline two existing micropayment protocols: PayWord [RS96] and PayFair [Yen01] as examples of postpaid and prepaid token-based payment protocols, respectively. These protocols deploy different kinds of cryptographic operations in that PayWord deploys public-key operations whereas PayFair is based on symmetric-key operations.

PayWord

PayWord [RS96] is a postpaid micropayment protocol based on public-key operations. Three parties are involved in the system: client, merchant, and bank.

At the beginning of the protocol, where the client and the merchant have originally established accounts with the bank, the client requests the bank to perform a transaction using PayWord. The bank issues the client a *PayWord certificate* which contains an authorized amount CL that the client is allowed to spend to each merchant. Then, the PayWord payment process can be performed as the following:

1. To make a payment to the merchant, the client generates a set of coins $\{c_0, ..., c_n\}$, where $n = CL$, which is specific to the merchant. The set of coins is generated as follows:

   $$c_i = h(c_{i+1}), \qquad \text{where } i = 1, ..., n-1$$

2. In the first payment, the client sends the merchant a *commitment*, which contains the PayWord certificate and c_0, digitally signed by the client.

3. Later on, in each payment, the client sends the coin c_i to the merchant. The merchant can infer the value of the coin i by applying a number of hash operations to c_i as follows: $c_0 = h^i(c_i)$.

4. At the end of a specified period e.g. a day or a month, the merchant sends the highest value of c_i together with the commitment to the bank. The bank then deducts the requested amount from the client's account by calculating the amount from c_0 and the maximum value of c_i, and transfers the charged amount to the merchant's account.

In PayWord [RS96], the computational load at the client is reduced compared to SET [Mas97] and iKP [BGH$^+$00] protocols because the client is not required to have payment authorization from the bank in every transaction. However, PayWord is not suitable for wireless environments because it has high client's computation due to public-key operations. In addition, the payment-related information, c_0 and c_i, is readable by any party who has the client's public key. Therefore, she is able to trace the client's spending behaviour.

PayFair

PayFair [Yen01] is a prepaid micropayment protocol which deploys symmetric-key operations and hash functions. The details of PayFair are given as follows:

Phase A: Prepaid Phase

$$\textbf{C}\rightarrow\textbf{B}: \quad ID_C, O_C, h(O_C, K_C) \qquad (a)$$

$$\textbf{B}\rightarrow\textbf{C}: \quad \{\{N, RN\}_{SK}, RT\}_{K_C}, N, h(\{N, RN\}_{SK}, N, O_C, K_C) \qquad (b)$$

From the above protocol steps, SK is a secret known only to the bank. K_C is shared between the client and the bank. RT is a nonce for replay protection. The client requests the bank by sending order number O_C containing the requested amount. The bank returns the message containing a payment token

$\{N, RN\}_{SK}$, which is later used for generating coins. RN is a random number generated from the serial number N and the secret SK. The client generates a set of coins w_i, where $i = 0, ..., n$ and $w_n = \{N, RN\}_{SK}$, from the process: $w_i = h(w_{i+1})$.

Phase B: Micropayment Phase

C$\rightarrow$**M**: $w_0, N, h(w_0, ID_M, K_C)$ (c)

M$\rightarrow$**B**: $w_0, N, ID_C, R_M, h(w_0, ID_M, K_C)$ (d)

B$\rightarrow$**M**: $w_0, ID_C, ID_M, YES, h(w_0, ID_C, K_M, R_M, YES)$ (e)

The client sends the message (c) containing w_0 to the merchant. The merchant then forwards $h(w_0, ID_M, K_C)$ with relevant information to the bank in (d). After receiving the message, the bank can generate w_n from w_0, N, and its own RN and SK. The bank then transfers the amount n to the merchant's account and sends the response to the merchant in (e). The client can make a payment to the merchant as follows:

C$\rightarrow$**M**: w_i where $i = 1, ..., n$ (f)

It can be seen that with the deployment of symmetric-key operations and hash functions, the computational load at the client of PayFair is reduced compared to that of PayWord [RS96]. However, the problem of revealing payment-related information occurred in PayWord still exists because, in (c) and (f), w_0 and w_i are transmitted in cleartext. In addition, although Yen [Yen01] claimed that the payment token w_n is general-purposed, it is still merchant-specific when used, that is, although the coins are merchant-independently generated, they can still be used with only one specific merchant. Thus, the client needs to request the bank for a new payment token every time she wants to make a payment to a new merchant. Moreover, in (c), the bank can impersonate as the client to perform transactions with the merchant.

2.3.3 Security vs Transaction Cost of Electronic Payment Systems

We have presented electronic payment systems in two categories: account-based and token-based systems. The payment systems in each category deploy various kinds of cryptographic operations to secure the transmission of payment-related information over an insecure channel. Selecting appropriate operations can enhance system performance.

Mainly, selecting cryptographic operations for a protocol is based on the value of the information to be transmitted. That is, in each transaction, there is certain amount of transaction cost occurring from the computation at each engaging party and from the duration to complete the transaction. PKI-based payment protocols that require higher computation and duration to complete each transaction than symmetric-key based protocols therefore incur higher transaction cost.

From the previous sections, on one hand, it can be seen that public-key operations are mostly applied to account-based payment systems which involve higher value transfer than token-based ones. As public-key operations satisfy all transaction security properties mentioned in section 1.4, applying them in the transaction can secure the payment-related information transfer. On the other hand, due to the transaction cost constraint, low-valued transactions deploy symmetric-key operations and hash functions, which satisfy most of the transaction security properties, as the main cryptographic operations based on the idea that the system can accept certain degree of risks from attacks because it may not be worthwhile for an attacker to spend time and effort on attacking the payment system with low-valued transactions.

In addition, communication environment and availability of cryptographic operations must be taken into account when selecting appropriate cryptographic operations. An obvious example is selecting cryptographic operations for fixed and wireless environments. As discussed in chapter 1, most mass-produced mobile devices cannot perform public-key operations efficiently. Even though such operations are performed on the devices equipped with enhanced technology such as smartcard, they are time and battery consuming resulting in high transaction cost which may not be acceptable by users. However, the deployment of simple cryptographic operations tends to be more vulnerable to attacks. Therefore, this tradeoff must be taken into account when designing a practical and secure mobile payment system.

2.4 Enabling Mobile Payment

In the previous section, we have shown that the electronic payment systems originally designed for fixed environments cannot be directly applied to wireless environments due to a number of limitations. To overcome such limitations, several payment frameworks have been proposed to enable mobile payment, mainly by migrating existing fixed-network payment systems to wireless networks and by designing payment systems specifically for wireless networks.

According to the migration of existing fixed-network payment systems, two frameworks have been proposed: *proxy-based* and *agent-based* frameworks. The details of these frameworks will be presented in sections 2.4.1 and 2.4.2, respectively. Whereas, the payment systems specifically designed for wireless environments are classified as *non proxy-based* framework. The details of the framework will be presented in section 2.4.3.

Table 2.1 summarizes existing mobile payment systems by categorizing the payment systems based on the above frameworks into account-based and token-based payment systems. The details of the mobile payment systems in each category will be presented in sections 2.4.1-2.4.3.

Frameworks	Account-based Payment Systems	Token-based Payment Systems
Proxy-based Framework	3D SET [WSZ01] Jin's approach [JRFH02] Antovski's approach [AG03] Dai's approach [DZ03]	

Agent-based Framework	SET/A [RdS98]
	SET/A+ [WLY99]	

Non Proxy-based Framework	Paybox [Pay01] WSET [FAB02]	Horn's approach [HP98] Park's approach [PBD01] Kim's e-cash [KLK$^+$02] Zheng's e-cash [ZK02]

Table 2.1: Summary of existing mobile payment systems

2.4.1 Proxy-Based Framework

A proxy-based mobile payment system allows a client to perform payment transactions by using an existing fixed-network payment protocol through a proxy server located in a fixed network operating on the client's behalf. This is because the fact that, on the migration of a payment protocol to perform transactions in a wireless environment, the engaging parties that already have existing payment infrastructure do not want to change or upgrade their expensive infrastructure to support mobile payment that may incur high cost. A simple solution is to set up a proxy server as a medium between mobile devices and the payment infrastructure. To make a payment to a merchant, the client sends a request over a wireless network to trigger the proxy server located in a fixed network to perform the transaction with the merchant on behalf of the client.

Several systems based on the proxy-based framework have been introduced [WSZ01, JRFH02, AG03, DZ03]. Three Domain (3D) SET for Mobiles [WSZ01] and Jin *et al.*'s approach [JRFH02] were proposed to enable SET payment transactions in wireless environments. Antovski *et al.* proposed an account-based payment protocol for wireless networks [AG03]. Dai *et al.*

[DZ03] proposed an authentication and payment scheme for mobile commerce by assigning a WAP gateway to perform transactions on behalf of the client in a fixed environment.

However, even though the proxy-based solution is compatible with existing fixed-network payment systems, the client is required to fully trust the proxy server in that the client's sensitive information, such as credit-card information and the client's private key, needs to be stored on the proxy server. Therefore, successful attacks on the proxy server will compromise the security of the entire system.

In this section, 3D SET for Mobiles [WSZ01], Antovski *et al.*'s approach [AG03], and Dai *et al.*'s approach [DZ03] are presented in details to give the idea how the proxy-based mobile payment systems work and their security issues are discussed.

Three Domain SET for Mobiles

Three Domain (3D) SET for Mobiles [WSZ01] is a system operating the SET protocol [Mas97] over wireless networks. In this system, an issuer performing transactions on behalf of a mobile client. The following steps illustrate how the system works (see Figure 2.2):

1. After browsing for goods or services on a merchant's site, the client starts a SET payment session by selecting SET as the payment protocol.

2. The merchant notifies the client that the SET payment session is about to start.

3. The client is redirected to the issuer's server. This server contains all the client's payment-related information including credit-card information.

4. The issuer displays a screen with all payment-related information such as product descriptions and price. Then, the client is requested to fill in

authentication information such as password to confirm the payment.

5. The client enters the authentication information. After the authentication is successful, the issuer performs the respective SET payment transaction on behalf of the client.

6. After the completion of the transaction, the client is redirected back to the merchant's site.

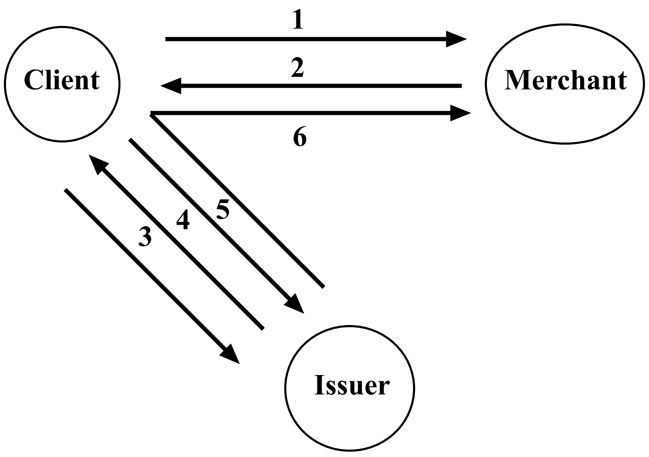


Figure 2.2: 3D SET for Mobiles

It can be seen that 3D SET for Mobiles offers many advantages:

- The client is not required to store any information on her mobile device. This can reduce the problem of limited storage of mobile devices.

- Only a few and small messages are transmitted over wireless networks. Moreover, all high computational operations are performed at the issuer. Thus, the communication load at the client side is low.

- As the system is centralized, all updates can be done only at the issuer including the change of the underlying payment protocol. This offers the flexibility to the system.

However, 3D SET for Mobiles has the following problems and limitations [WSZ01]:

- The client is required to fully trust the issuer that the issuer will not perform payment transactions on the client's behalf without the authorization from the client.
- The client's confidential information needs to be stored on the issuer's server. This information will possibly be lost by successful attacks to the issuer's server.
- The issuer can trace the spending behaviour of its clients and build the detailed client profiles.

Antovski *et al.*'s Approach

Antovski *et al.* [AG03] proposed an account-based payment system for wireless networks. Three parties are involved in this system: client, merchant, and Financial Service Provider (FSP). FSP, acting on behalf of the client, performs transactions to the merchant on the client's behalf. In this system, the client establishes an account with the bank and communicates with the bank through FSP. The payment protocol for this system is shown as follows.

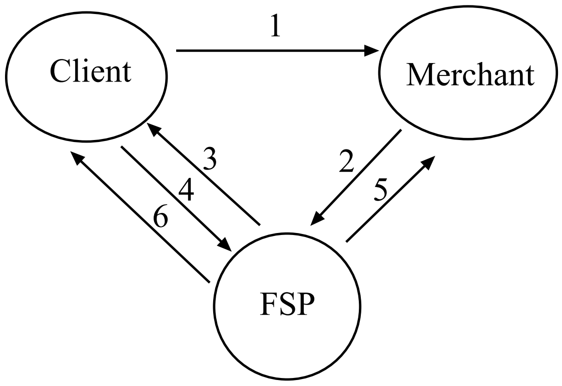


Figure 2.3: Antovski *et al.*'s approach

1 C→M: $InvoiceRequest$

2 M→FSP: $\{Invoice\}_K$

where K is the session key generated from SSL handshake.

3 FSP→C: $\{\{Invoice\}_{K_{FSP}^{-1}}\}_{K_C}$

4 C→FSP: $\{\{DeductionRequest\}_{K_C^{-1}}\}_{K_{FSP}}$

5 FSP→M: $Acknowledgement$

6 FSP→C: $PaymentReceipt$

The details of the above payment protocol are shown as follows:

1. The client sends a request to make a payment to the merchant.

2. The merchant sends an invoice to FSP over a SSL channel.

3. FSP signs the invoice with its own private key K_{FSP}^{-1} and encrypts with the client's public key K_C, and then redirects the message to the client to request for the confirmation of money deduction from the client's account.

4. The client retrieves the message. To confirm the payment, the client enters a password to unlock her private key stored on her mobile device. Then, the client sends FSP a deduction request as the payment confirmation signed with the client's private key K_C^{-1} and encrypted with FSP's public key K_{FSP}.

5. On receiving the message, FSP verifies the client's signature and retrieves the payment confirmation. Then, FSP forwards the confirmation to the bank.

6. FSP sends the notification to the merchant.

7. FSP also sends the payment receipt to the client.

Antovski *et al.* do not provide the details of each message transferred in the protocol. They claimed that the protocol is suitable for wireless environments. However, its computational and communication load are still doubtful because this protocol deploys public-key operations, especially at the client side.

Dai *et al.*'s Approach

Dai *et al.* proposed a payment system that allows clients perform transactions via mobile phones. Three parties are involved in this system: client, WAP gateway, and merchant. The client is assumed to have a WAP-enabled mobile phone, whereas the merchant has an e-commerce WAP site. This system deploys a WAP gateway to perform secure operations on behalf of the client. Dai *et al.* argued that a WAP1.2 phone is capable of generating digital signature according to PKCS#1 standard [RSA93a] which cannot provide signature validation process. Note that the signature validation process is provided by the PKCS#7 standard [RSA93b].

To solve this problem, Dai *et al.* introduced a secure module into the WAP gateway to perform the functions that cannot be done on the WAP phone such as storing certificates, generating signatures according to PKCS#7 standard, and validating the merchant's signature. Dai *et al.* also proposed an authentication and payment scheme based on this system. The client authentication scheme for this system can be simplified in Figure 2.4 as follows:

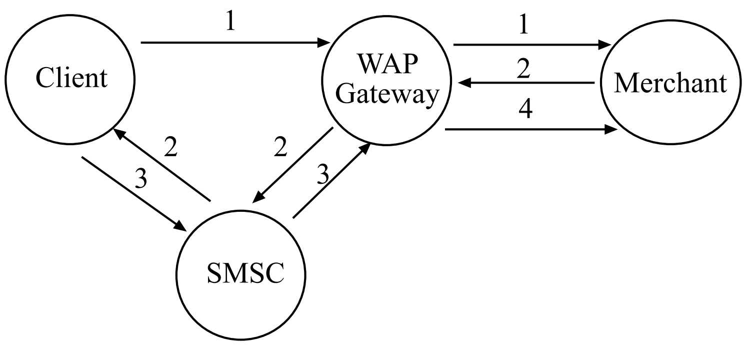


Figure 2.4: The authentication scheme according to Dai *et al.*'s approach

1. The client sends an authentication request to the merchant via the WAP gateway.

2. The merchant generates an authenticate message to the client via the WAP gateway and a SMSC (Short Message Service Center).

3. The client signs the message with PKCS#1 signature and passes it to the WAP gateway via the SMSC.

4. The WAP gateway verifies the signature. It signs the message with PKCS#7 signature and sends to the merchant.

The payment scheme for this system is shown in Figure 2.5 as follows:

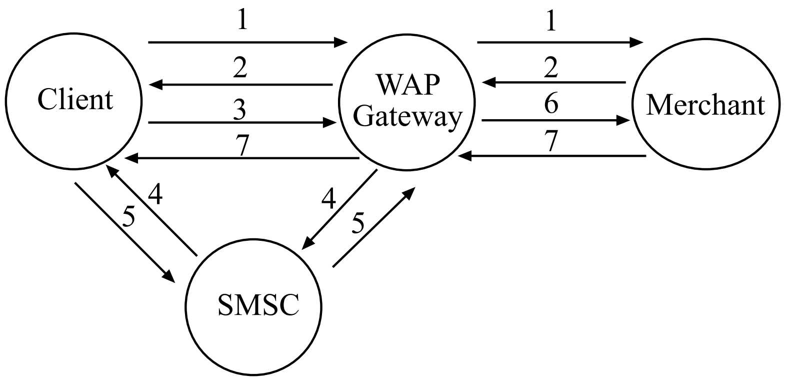


Figure 2.5: The payment scheme according to Dai *et al.*'s approach

1. The client sends a request to the merchant via the WAP gateway.

2. The merchant sends back the payment type, e.g. credit or debit, to the client via the WAP gateway.

3. The client selects the payment type preferred to the WAP gateway.

4. The WAP gateway sends a contract to the client via the SMSC.

5. The client signs the contract with PKCS#1 signature and sends it back to the WAP gateway via SMSC.

6. The WAP gateway verifies the signature and performs the transaction on behalf of the client to the merchant.

7. The merchant sends the receipt to the client via the WAP gateway.

In this system, it is obvious that the security of the client relies on the security and trustworthiness of the WAP gateway. That is, the WAP gateway is able to impersonate as the client because it has the client's private key. Moreover, the use of SMSC is unnecessary because the entire transaction can be performed with the direct communications between the client and the WAP gateway. The use of SMSC also incurs extra cost for the client in addition to being charged for the WAP usage. That is, in each transaction, the client has to be charged for two SMS messages: one in the client authentication phase and the other in the payment phase.

2.4.2 Agent-Based Framework

An agent-based payment system applies mobile agent technology to existing payment systems to enhance the capability of performing transactions in wireless environments. The concept is that a mobile client sends an agent, acting on her behalf, containing payment-related information, to perform a transaction in a merchant's environment over a fixed network. With the use of mobile agent, the client is required to stay connected to the Internet only for short periods of time during the entire transaction. This results in the reduction of connection cost. Moreover, the computation load at the client is reduced because the agent performs the transaction outside the client's environment.

In this section, two agent-based mobile payment systems, SET/A [RdS98] and SET/A+ [WLY99], are presented and discussed in details.

SET/A

SET/A is an agent-based SET payment system proposed by Romao *et al.* [RdS98] in order to solve the problems of implementing SET protocol [Mas97] in wireless environments previously discussed in section 2.3.1. The concept of SET/A is to let a mobile agent perform a transaction on behalf of a client outside the client's environment.

In SET/A, the original SET protocol is operated in asynchronous mode. The operations of SET/A can be depicted in Figure 2.6.

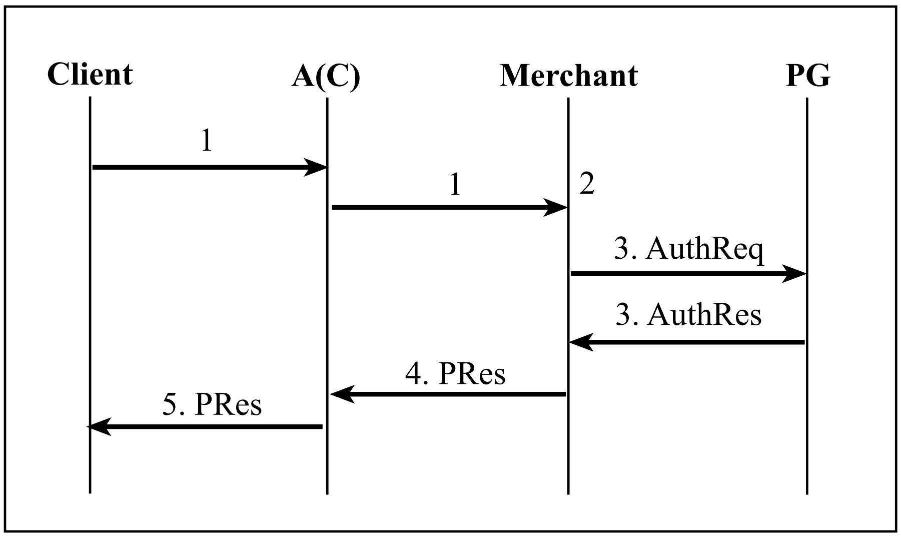


Figure 2.6: SET/A

1. The client starts a SET initialization request by sending an agent containing the client-side software called *SET wallet* together with payment-related information travelling to the merchant's server and staying there in order to perform a transaction on behalf of the client. From Figure 2.6, A(C) represents the agent owned by the client.

2. At the merchant's server, the client's **PReq** is generated and sent to the merchant.

3. The merchant performs the transaction as per the original SET protocol (as presented in section 2.3.1) until receiving **AuthRes** from the payment gateway.
4. After receiving **AuthRes**, the merchant generates **PRes** and passes it to the agent residing in the merchant's server itself.
5. After receiving **PRes**, the agent returns to the client with the result of the transaction.

It can be seen that, with the use of mobile agent, the client stays connected to the Internet only two short periods of time: one for sending the agent containing the client's request to the merchant at the beginning of the transaction and the other for receiving the agent back with the result of the client's request at the end of the transaction. Thus, the cost of connection is reduced. Moreover, the client's computational load is reduced by sending the agent to perform operations at the merchant's server.

However, performing the SET operations at the merchant is vulnerable to attacks because the merchant's server is considered as a hostile environment [WLY99]. In particular, in order to establish **PReq**, the agent is required to randomly generate the session key K_2 for encrypting PI at the merchant's server. The merchant may be able to retrieve K_2 during the key generation. Romao *et al.* [RdS98] suggested that SET/A should be operated in a tamper-proof environment or by using secure co-processors [Yee94], however they are complicated and can incur high cost [WLY99].

SET/A+

Wang *et al.* [WLY99] proposed a modified agent-based SET payment system (called SET/A+) to solve the problems of SET/A mentioned in the previous section. They proposed the SET transaction from the user's point of view that considers the payment process as a stage of Internet trading. This makes the system operate in a larger scenario than that of SET/A.

SET/A+ deploys two mobile agents performing two different tasks: (i) *Information Gathering and Negotiation Agent* (INA) for brokering and negotiation, and (ii) *Payment Agent* (PA) for performing the SET payment transactions. INA is sent from the client to collect information about goods and corresponding merchant and returns to the client. PA performs payment operations on behalf of the client. To make a payment, the client sends PA containing **PReq** to the merchant. PA stays at the merchant's server until receiving the result of the client's request **PRes**, and then returns to the client.

Wang *et al.* [WLY99] argued that SET/A+ has practical advantages over SET/A by considering the payment process together with the brokering and negotiation. This allows the system to operate in a larger framework. Moreover, **PReq** is completely generated at the client. Thus, the problem of session key compromise occurred in SET/A is eliminated.

However, generating **PReq** at the client side results in the problem of the high computational load which is what Romao *et al.* [RdS98] tried to avoid when designing SET/A.

2.4.3 Non Proxy-Based Framework

A non proxy-based mobile payment system requires no proxy server. Therefore, it takes the constraints of wireless environments into account. These constraints mainly come from mobile devices and wireless networks discussed in chapter 1. Thus, various kinds of lightweight cryptographic techniques have been applied to reduce computational and communication load of the systems.

Due to the fact that non proxy-based mobile payment systems do not require any proxy server or mobile agent to increase transaction performance of the systems, their main focus is the payment protocols operated in the systems. Such payment protocols have to be lightweight and provide sufficient security. In this section, we therefore focus on existing non proxy-based payment protocols.

Several payment systems and protocols have been proposed to secure non proxy-based mobile payment [Pay01, FAB02, KLK$^+$02, ZK02, PBD01]. Paybox [Pay01] is a payment system based on mobile telecommunication system. Its security also relies on the security of the mobile telecommunication system. Wireless SET (WSET) protocol [FAB02] was proposed to enable SET transactions in wireless environments by reducing the computation load at mobile users by using WTLS certificates [WAP01] instead of SET certificates. Kim *et al.* [KLK$^+$02] proposed an electronic cash protocol for wireless networks based on hash functions and digital signature. Zheng *et al.* [ZK02] proposed another electronic cash payment protocol based on Brands' restrictive blinding signature to ensure untraceability property. Park *et al.* [PBD01] proposed a micropayment protocol for wireless networks based on tamper-proof resistant devices.

In this section, we outline Park *et al.*'s protocol [PBD01], Paybox [Pay01], Kim *et al.*'s protocol [KLK$^+$02], and WSET protocol [FAB02] to provide the idea how non proxy-based mobile payment systems work and how various kinds of cryptographic techniques have been applied to the payment systems including the discussion about their security issues.

Park *et al.*'s Protocol

Park *et al.* [PBD01] proposed a postpaid micropayment protocol based on tamper-resistant devices. The protocol deploys only hash functions which result in the reduction of engaging parties' computation.

Three parties are involved in the protocol: user (or client) U, value-added service provider (or merchant) V, and service provider (or bank) S. The initial assumptions of Park *et al.*'s protocol are shown as follows:

1. S possesses the keys K_{UV} and K_{US}, where $K_{UV} = h(K_{VS}, ID_U)$ and $K_{US} = h(K_S, ID_U)$. Note that K_S is a secret known only to S.
2. U possesses the keys K_{UV} and K_{US}. These secrets are sent from S.
3. V possesses the keys K_{UV}, where $K_{UV} = h(K_{VS}, ID_U)$. Note that K_{VS} is transferred from S to V in order to generate K_{UV}.

Generally, when U requests to make a payment to V, the requested amount is deducted from U's account established at S and transferred to V's account by S. Following PayWord [RS96] presented in section 2.3.2, Park *et al.*'s protocol is divided into 3 sub-protocols: *Payment Initialization Protocol*, *Payment Protocol*, and *Payment-clearing Protocol*. The details of these protocols are shown as follows.

Payment Initialization Protocol

1. U sends her identity ID_U and a random number r_U to V.

   U→V: ID_U, r_U (1)

2. V responses by sending U a random number r_V together with charging data *chdata* and timestamp TS_V to U. Note that $h(r_U, r_V, K_{UV})$ is used for authentication purpose.

   V→U: $r_V, h(r_U, r_V, K_{UV}), chdata, TS_V$ (2)

3. After receiving the message, U generates a random number c_n, calculates a set of electronic coins c_j, where $j = 1, ..., n$ and $c_j = h(c_{j+1})$, and calculates $commitment_V$ and $commitment_S$. U then sends V the following message:

   U→V: $c_0, commitment_V, commitment_S$ (3)

where $commitment_V = h(ID_U, ID_V, K_{UV}, r_U, r_V, TS_V, chdata, c_0)$ and $commitment_S = h(commitment_V, K_{VS})$. V can verify $commitment_V$ to ensure that U has requested to make a payment to her.

Payment Protocol

1. To start making a payment, U sends V c_j, where j is equal to the value of the requested goods or services.

   U→V: c_j (4)

2. After receiving c_j, V infers the value of the coin by repeatedly applying a number of hash functions to c_j and comparing to c_0.

Payment-clearing Protocol

1. After a specified period, V sends S the message including $c_{j_{max}}$, the highest value of c_j, to redeem the money.

   V→S: $c_0, c_{j_{max}}, ID_U, ID_V, r_U, r_V, chdata, TS_V,$
   $commitment_V, commitment_S$ (5)

2. S verifies $commitment_V$ and $commitment_S$ and compares $c_{j_{max}}$ with c_0. Then, S transfers the amount j_{max} to V.

Park *et al.*'s protocol is claimed to be suitable for wireless environments because it has low computation at all engaging parties [PBD01]. However, it has following problems:

1. As S knows all parties' secret keys, K_{US} and K_{UV}, it can impersonate as U to perform transactions to V.

2. c_0 and c_j used to infer U's requested price are transmitted in cleartext. Thus, it is possible to trace U's spending behavior.

Paybox

Paybox [Pay01] is a simple payment system based on a mobile network system in that both client and merchant are required to have mobile devices with unique phone numbers. Paybox allows the client to perform payment transactions from any kinds of telephone including mobile phone to mobile merchants, e.g. taxi drivers, or to other mobile users.

There are three parties involved in the Paybox system: payer (or client), payee (or merchant), and Paybox server. To perform a Paybox transaction, both client and merchant need to have bank accounts established. Figure 2.7 illustrates how Paybox operates. The details of Paybox transaction are given as the following:

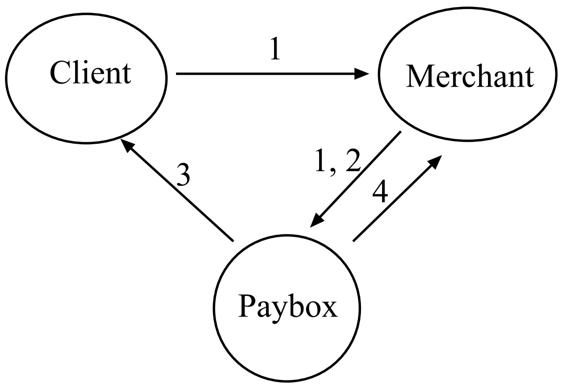


Figure 2.7: Paybox transaction

1. After the client agrees to purchase products or services from the merchant, the merchant contacts the Paybox server via the telephone or data network.

2. The merchant passes the requested amount and the client's mobile phone number to the Paybox server.

3. The client is called back, and she needs to authorize the payment by entering a PIN number.

4. The requested amount is then transferred from the client's account to the merchant's account.

It can be seen that the security of Paybox relies heavily on the security of the telecommunication network. No additional data security is introduced [WSZ01]. Moreover, the client authentication can be achieved only by telling or entering the PIN number to the Paybox system. This offers a possibility of an attacker to tap the conversation or the data transmitted between the client and the Paybox server.

Kim *et al.*'s Electronic Cash

Kim *et al.* [KLK+02] proposed an electronic cash protocol for wireless environments. The protocol reduces party's computation by deploying only hash functions and digital signature. Three parties are involved in the protocol: client, merchant, and bank. The protocol descriptions are shown as follows:

1. After the client and the merchant have agreed on goods or services, the merchant sends the price of the goods *value* and transaction code T to the client.

   $$\textbf{M}\rightarrow\textbf{C:} \quad h(T, value), value \tag{1}$$

2. The client requests for a payment token by sending the following message containing the password *Password* shared between herself and the bank to the bank.

   $$\textbf{C}\rightarrow\textbf{B:} \quad ID_C, Password, h(T, value), value \tag{2}$$

3. The bank verifies the password and issues the payment token to the client.

   $$\textbf{B}\rightarrow\textbf{C:} \quad ID_B, \{h(T, value)\}_{K_B^{-1}} \tag{3}$$

4. $h(T, value)$ is considered as the payment token that the client is allowed to spend with the merchant. To pay for the goods, the client sends the merchant the following message:

   C→M: $T, ID_B, \{h(T, value)\}_{K_B^{-1}}$ (4)

5. After receiving the message, the merchant deposits the amount *value* to the bank by sending the bank the following message:

   M→B: $ID_V, \{h(T, value)\}_{K_B^{-1}}$ (5)

Although Kim *et al.*'s protocol offers low party's computation, it has the following problems:

1. *Password* is transmitted in cleartext over the air interface. Although it is transferred through WAP (Wireless Application Protocol) [WAP02] which deploys WTLS protocol [WAP01], a variant of TLS protocol (formerly known as SSL protocol version 3.0 [FKK96]), it still suffers from the problem of WAP security gap. Note that normally, the mobile client communicates to the fixed-network merchant through a WAP gateway. The client transfers the information encrypted with a shared key between herself and the WAP gateway over the WTLS protocol to the WAP gateway. Then, the information is decrypted and encrypted with another key shared between the WAP gateway and the merchant over TLS protocol. Such information is revealed to the WAP gateway because it lacks of end-to-end security between the client and the merchant.

2. In the message (5), the payment token $\{h(T, value)\}_{K_B^{-1}}$ does not bind with the merchant's identity. Thus, any party who successfully captures $\{h(T, value)\}_{K_B^{-1}}$ can claim the money with the bank.

3. Kim *et al.* did not mention how the client receives T. Also, $h(T, value)$ received in the message (4) does not contain the the client's balance after being transacted.

4. The client should have been able to verify $h(T, value)$ by applying a hash function to T and $value$. However, the client does not have T.

5. The protocol lacks of non-repudiation property in most of the messages, which are the messages (2), (4), and (5).

6. In the message (5), the bank should have been able to retrieve $value$ from $h(T, value)$ but it cannot because the bank does not have T.

WSET Protocol

Fourati *et al.* [FAB02] proposed a wireless variant of SET protocol [Mas97] called WSET. The wireless part of WSET is based on WTLS protocol. A client in WSET performs a payment transaction with a merchant through a WAP gateway and deploys WTLS certificates instead of SET certificates in order to reduce computational load. In addition, using the WTLS certificates results in end-to-end security between the client and the merchant that solves the WAP security gap problem mentioned in the previous section. However, the client still has to perform high computational operations on the SET's **PReq** generation.

2.4.4 Comparison of Existing Frameworks for Mobile Payment

In sections 2.4.1-2.4.3, we have outlined existing frameworks to enable mobile payment and discussed their security and performance issues. In this section, we compare the existing frameworks by considering the following issues:

- **Constraints of Wireless Environments:** the main objective of the existing frameworks is to overcome the constraints of wireless environments. To do so, the amount of data transferred over a wireless network must be minimized. It can be seen that the proxy-based framework seems to be a good candidate because it has least amount of data transmitted across the wireless network. In this framework, a client sends only a request to trigger a proxy server to perform transactions on her behalf. The client therefore requires less storage and computational load compared to the client in other frameworks. However, the client still needs to stay online during the entire transaction. Thus, the performance of payment protocol and the performance of communication networks, both fixed and wireless ones, must be taken into account.

  Moreover, the connection cost must be reduced. This can be achieved by the agent-based framework. In this framework, the connection cost is more concerned than transaction duration in that the agent-based framework tends to focus on the reduction of connection period which is relevant to the connection cost.

  From Table 2.1, it can be seen that the proxy-based and the agent-based frameworks are likely to be applied to secure account-based payment systems that require higher computational and communication load, whereas non proxy-based framework tends to be applied to token-based payment systems which require lower computational and communication load. However, due to the operational cost constraint of token-based payment systems, the proxy-based framework may not be suitable for this kind of payment systems because the setup and operational cost of the proxy server may exceed the value of goods or services.

  From sections 2.4.1 and 2.4.2, the proxy-based and the agent-based frameworks tend to enable payment transactions through existing fixed-network payment protocols in wireless environments. However,the non proxy-based framework tends to design new payment solutions based on existing wireless communication technology such as WAP [WAP02] or short message service (SMS). Although one of the systems in this category, WSET [FAB02], focuses on implementing a fixed-network payment protocol in a wireless environment, the technique applied to it is based on the underlying transport layer security, WTLS protocol [WAP01]. Moreover, non proxy-based mobile payment systems tend to deploy lightweight cryptographic operations, such as symmetric encryptions or hash functions, to reduce the computational load, especially at the client, and the communication load during the transaction.

- **Trust Relationships among Engaging Parties:** the client in the proxy-based framework is required to fully trust the proxy server because her sensitive information is stored on the proxy server. Whereas, the client in the agent-based and the non proxy-based frameworks is not required to trust any other party.

- **Security against Attacks:** the security of data transfer over wireless networks relies on the security technique applied to it. In particular, in all the frameworks stated above, the message transferred from the client to either the proxy server or the merchant must be secured by deploying security services provided by application-layer and transport-layer protocols. For example, in WSET [FAB02], the client performs SET, an application-layer protocol, payment transactions over WTLS/TLS protocols which are transport-layer protocols.

  Additionally, there is a concern about the security of agent-based framework. As discussed in section 2.4.2, SET/A [RdS98] is vulnerable to attacks because the client allows the mobile agent to bring her sensitive information, such as private key or credit-card number, and generate an encrypting key in the merchant's environment.

2.5 Integrated Mobile Payment Framework

According to the discussion in section 2.4.4, it can be seen that each mobile payment framework has its own advantages and disadvantages. From our point of view, we can offer a framework which is more general and offers a better solution than the existing frameworks in that:

1. We deploy a non proxy-based payment protocol to perform payment transactions in the system. Thus, we can achieve lightweight payment transactions.
2. We deploy a mobile agent to perform transactions on behalf of the client to reduce connection cost.
3. We deploy proxy-based framework by setting up a proxy server to provide a trusted environment for the mobile agent to perform high computational operations and to generate the client's request following the structure of the payment protocol.

Figure 2.8 presents the framework that illustrates our approach. Note that $A(C)$ stands for an agent A owned by the client C. Our framework incorporates the main advantages and solves the problems of the existing frameworks. Later in this thesis, chapter 4 will present our framework in details. We demonstrate the practical usefulness of the proposed framework by applying it to SET protocol [Mas97]. Then, in chapter 5, two non proxy-based payment protocols are presented to enable practical and secure mobile payment. Applying either of them to our framework enhances transaction security and performance.

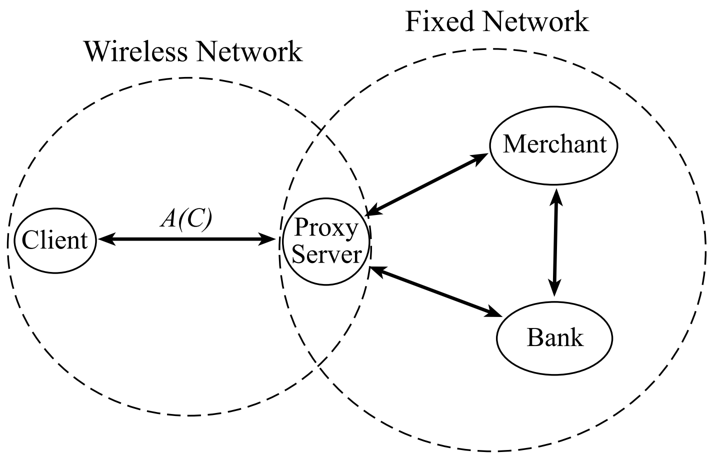


Figure 2.8: Integrated mobile payment framework

2.6 Securing Transfer of Account Information During Payment Transactions

In a shared-key based system, a user and the system (or another user) share secret information (or secret key) that can be used for several purposes:

- **Credential or Authentication Token:** the shared secret can be used to authenticate the user to the system such as access password. For example, the user and the system establish a secure communication channel by running a key exchange protocol. Then, the user supplies her username and password to the system through the secure channel as a request for remote access. The system checks if the supplied username and password match the ones in its database. If they are matched, the user is allowed to access the system resource.

  In some applications such as credit-card payment, a cardholder (or a client) sends her credit-card number, which is the secret shared between herself and her (card) issuer, as an authentication token including payment-related information to a merchant through a secure channel, such as SSL [FKK96], to request a payment to the merchant. The merchant then forwards such information to the issuer to request for payment authorization. As the credit-card number is shared between the client and the issuer, the issuer can verify that the client's request is valid. The issuer then deducts the requested amount from the client's account and transfers it to the merchant's account.

- **Cryptographic Operations:** the shared secret can be used as the key for encrypting or hashing a message sent between parties. For example, Alice sends Bob a message, encrypted with a shared key between herself and Bob, securely over an open network. If Bob did not previously generate this message, he can infer that this message has been originated by Alice. This is because, in addition to Bob, only Alice can generate the message.

Considering Internet payment scenario, the most obvious application which deploys shared secrets is the credit-card payment scheme over SSL [FKK96]. In this scheme, a credit-card number is considered as an authentication token shared between its owner (a client) and her issuer. To make a payment to a merchant, after a SSL connection is established, the client supplies her credit-card number and relevant information, such as date of birth and billing address, to authenticate herself to the issuer. In SET protocol [Mas97], the client's credit-card information as an authentication token is encrypted with the payment gateway's public key and then signed with the client's private key. When it is transferred to the issuer (via the payment gateway), the issuer can infer that this request has been originated by the client and it contains the valid authentication token. In this section, we focus on the security of credit-card information transfer during payment transactions to point out the security issues related to the deployment of shared secrets in payment transactions.

Obviously, the most sensitive information in any payment system is account information which is shared between a client and her issuer. Several security issues related to the exposure of the account information have been reported [RW02, Kra99]. In SSL-based credit-card payment system, although the credit-card information is securely transferred through a SSL channel, it is still revealed to the merchant who is considered as an untrusted party. In SET protocol [Mas97], the encrypted credit-card information is decrypted by the payment gateway and then forwarded to the issuer. This problem may arise if the payment gateway and the issuer are different parties. That is, the payment gateway may be a company that is monitoring the system. It may possibly have a conspiracy with an attacker, or even the merchant, so that the attacker can get the client's credit-card information without any attempt to decrypt any messages.

Moreover, the credit-card number is considered as long-term, reusable, semi-secret information. It is printed on the card which is visible to everyone, and the client's information such as date of birth and billing address is not difficult to find out. Although the credit-card number is replaced by a secret known only between the client and the issuer [LZ04], it still has to be transferred in every transaction. Therefore, it is vulnerable to various kinds of attacks.

Several techniques have been proposed to secure credit-card information transfer over the Internet [Kra99, RW02, LZ04, Sha02]. Credit-card number blinding technique was proposed in [Kra99] by applying HMAC [KBC97] to the credit-card number and a random number. Then, the output of HMAC and the random number are sent to the issuer for verification. Thus, the value of HMAC in each transaction is different. Recently, several concepts of disposable credit-card numbers (DCNs in short) have been proposed in both online [Sha02] and off-line schemes [RW02, LZ04, KSL03a]. These techniques allow each client to perform transactions with a fresh credit-card number in every transaction.

In this thesis, we focus our consideration on off-line DCN generation techniques because they offer advantages over the online techniques. That is, the off-line techniques do not require any communications between the client and the issuer to generate a DCN in every transaction, whereas the online ones do. The off-line techniques therefore do not require any secure channel established between the client and the issuer.

Several off-line DCN generation techniques have been proposed. In Rubin *et al.*'s scheme [RW02], each DCN is generated from the encryption of payment information with a long-term key shared between the client and the issuer. In Li *et al.*'s scheme [LZ04], a new DCN is generated from the hash value of previously used DCN and a long-term key shared between the client and the issuer.

In this section, we outline two existing off-line DCN generation techniques. Section 2.6.1 presents Rubin *et al.*'s scheme. In section 2.6.2, Li *et al.*'s scheme is described in details.

2.6.1 Rubin *et al.*'s Scheme

Rubin *et al.* proposed an off-line DCN generation technique [RW02] which eliminates the need of long-term, reusable credit-card numbers. In this technique, a DCN is generated by the encryption of a set of payment-related information (called *restrictions*), containing payment amount, merchant's identity, billing address, etc., with a long-term shared key between a client and her issuer. For example, Alice wants to purchase a 50-dollar book from Bob's store. She generates a token T as follows:

$$T = \{\textit{fifty-dollars-book-Bob's-store}\}_K$$

where K is the long-term key shared between Alice and the issuer. On receiving this message, the issuer decrypts the message by using the key K. This technique also deploys timestamp for replay and collision protection. Note that the collision may occur when two different payment information is encrypted either with the same key or even with different keys. Although, Rubin *et al.* argued that the system is secure against various kinds of guessing attacks, to some degree, the encryption with the long-term shared key is vulnerable if an attacker has sufficient information and attempts to decrypt the DCN. The system will fail if the long-term key is compromised. Waiting until the fraud is being detected may be unacceptable to the clients whose credit-card information falls into the wrong hands. Moreover, Li *et al.* [LZ04] argued that the encryption may be computationally expensive when there are many users and restrictions.

2.6.2 Li *et al.*'s Scheme

Li *et al.* proposed a technique to generate DCNs based on one-way hash functions for a smartcard-based environment [LZ04]. Initially, a smartcard-based credit card is issued to a client. The information stored on the credit card is composed of the semi-secret, 16-digit credit-card number CCN, the long-term key S, and the initial session DCN T_{init}. CCN also appears on the card. These secrets are known only to the client and the issuer.

In the first transaction, the client sends the issuer T_{init} for payment authorization. In next transactions, the client generates new DCNs T_{new} as follows:

$$T_{new} = h(T_{cur}, S)$$

where T_{cur} stands for the previous-used DCN. The client then sends T_{new} to the issuer. On receiving T_{new}, the issuer calculates T_{new} and compares with the one received from the client in order to verify the client. It can be seen that the security of the system is based on the length of S and T and the security of the hash function. Although it is assumed that the hash function is irreversible, the use of the long-term key S offers an opportunity for an attacker to attempt to compute S from eavesdropping DCNs. Moreover, successful guessing S will compromise the security of the system.

2.6.3 A Possible Solution to Secure Account Information Transfer

In previous sections, we have discussed the importance of securing account information transfer during transactions. The exposure of account information will compromise the security of the entire system. Unfortunately, the existing techniques to secure the account information still rely on the long-term shared key. The compromise of the key results in the exposure of the account information.

In this thesis, we will investigate a technique to secure the transfer of shared secrets during payment transactions. Our technique is based on the generation of limited-use shared secrets that relies on a randomly generated group of shared secrets instead of a single long-term shared secret. The higher number the limited-use shared secrets have been used, the less chance the attacker can compromise the system. Our limited-use shared secrets can be used as single-use credentials, such as DCNs, or the keys for cryptographic operations, such as encryptions or keyed-hash functions. The details of the proposed technique will be presented in chapter 8, whereby the limited-use shared secrets generated by our technique will be used to enhance the security of our non proxy-based mobile payment protocols.

2.7 Summary

In this chapter, we have shown that, on one hand, existing fixed-network payment systems cannot be directly applied to wireless environments because most of them deploy complex and expensive cryptographic operations. In particular, they deploy high computational operations and have high number of communication passes [Mas97, BGH^{+}00, RS96]. Moreover, they have the problems of the lack of important transaction security properties and the exposure of confidential information to unauthorized parties [Yen01, PBD01]. On the other hand, existing mobile payment systems suffer from a number of security problems including the lack of privacy of payment information.

Then, we have discussed the possibility to have a framework which is more suitable for mobile payment than existing frameworks namely proxy-based, agent-based, and non proxy-based frameworks. We also briefly discussed about the proposed framework that solves the problems of the existing frameworks. The details of the framework will be presented in chapter 4. Moreover, in chapter 5, we will introduce two non proxy-based mobile payment protocols which operate well even in non proxy-based environment. With the proposed framework and protocols, the requirements for having practical and secure mobile payment are fulfilled.

We raised the issue regarding the vulnerability of deploying long-term, reusable shared secrets in payment transactions. Such secrets can be represented by account information which needs to be transmitted in every transaction. We then briefly discussed a possible approach to secure account information transfer during each transaction. Its details and applications to mobile payment protocols including the proposed protocols will be presented in chapter 8.

It can also be seen that the existing payment systems fail to satisfy at least one of the requirements presented in section 2.1. As a result, a payment system that satisfies both requirements is required so that the mobile users can have practical and secure payment transactions which potentially increase the acceptability of mobile payment to users.

Chapter 3

Formal Mobile Payment Model

We have demonstrated in chapter 2 that a practical and secure mobile payment system is one of essential goals of conducting electronic commerce transactions over wireless networks. To achieve this goal, on one hand, a protocol designer needs to have a guideline for designing a mobile payment protocol which satisfies the requirements of a practical and secure mobile payment protocol stated in section 2.1. On the other hand, a payment system provider needs to consider whether the chosen payment system provides sufficient transaction security and is practical for its users.

In this chapter, we present a formal mobile payment model which describes characteristics and interactions among engaging parties including important properties that must be satisfied by the system. Section 3.1 models a general electronic payment system. In section 3.2, a formal model for a practical and secure mobile payment system is introduced. Section 3.3 presents a brief guideline to analyze a mobile payment system based on the proposed model. Section 3.4 summarizes the chapter.

3.1 Electronic Payment

3.1.1 Electronic Payment System

In chapter 2, several electronic payment systems have been outlined and discussed. From such systems, we can formally describe an electronic payment system as the following:

Definition 3.1 (Electronic payment system) *A payment system S is defined as unions of the following sets:*

$$S = \{G, CE, PT\} \cup Goals \cup PR \cup TSec \cup Trust \tag{3.1}$$

where,

- G, where $G \neq \phi$, stands for the set of engaging parties in S.
- CE, where $CE \neq \phi$, stands for communication environment which is composed of payment devices and networks.
- PT stands for a payment transaction which represents actions regarding fund transfer performed by engaging parties in G in the communication environment CE. Generally speaking, PT represents a payment protocol in S.
- $Goals$ stands for the set of goals of engaging parties regarding the payment transaction PT.
- PR stands for the set of party's requirements for the payment transaction PT.
- $TSec$ stands for the set of transaction security properties.
- $Trust$ stands for trust relationships among the engaging parties in G.

It can be seen that S must have at least G, CE, and PT as the main elements. In other words, any payment system must be composed of the set of engaging parties G which perform the payment transaction PT to one another in the communication environment CE, e.g. fixed or wireless environment, whereas satisfying the other sets, such as PR, $TSec$, or $Trust$, enhances security of the system. Figure 3.1 illustrates the electronic payment system defined by the proposed formal model diagrammatically.

From figure 3.1, we can see that a payment system is primarily composed of two components: *Operational Semantics* and *Properties*. On one hand, Operational Semantics are the essential elements of the system in that a system will not be considered as a payment system if it lacks of any of these elements. They are composed of engaging parties (G), communication environment (CE), and payment transaction (PT). Properties, on the other hand, fulfill security and practicability of the system. They are mainly composed of Goals and Requirements, Security, and Performance. A payment system should achieve goals of engaging parties ($Goals$) at design stage and satisfies requirements to complete the transaction in engaging parties' points of views (PR) at implementation stage. Moreover, the payment system is considered to be secure if it satisfies Security which consists of transaction security properties ($TSec$) and trust relationships among engaging parties ($Trust$) stated by the model. In addition, the payment system is considered to be practical if it satisfies transaction performance (TP) stated by engaging parties in the system. The details and formalization of these elements will be presented later in this chapter.

3.1.2 Engaging Parties

Definition 3.2 (Engaging parties) *A payment system S consists of a set of engaging parties G, where $G = \{C, M, I, A, PSP\}$.*

- C stands for a client who wants to purchase goods or services from a

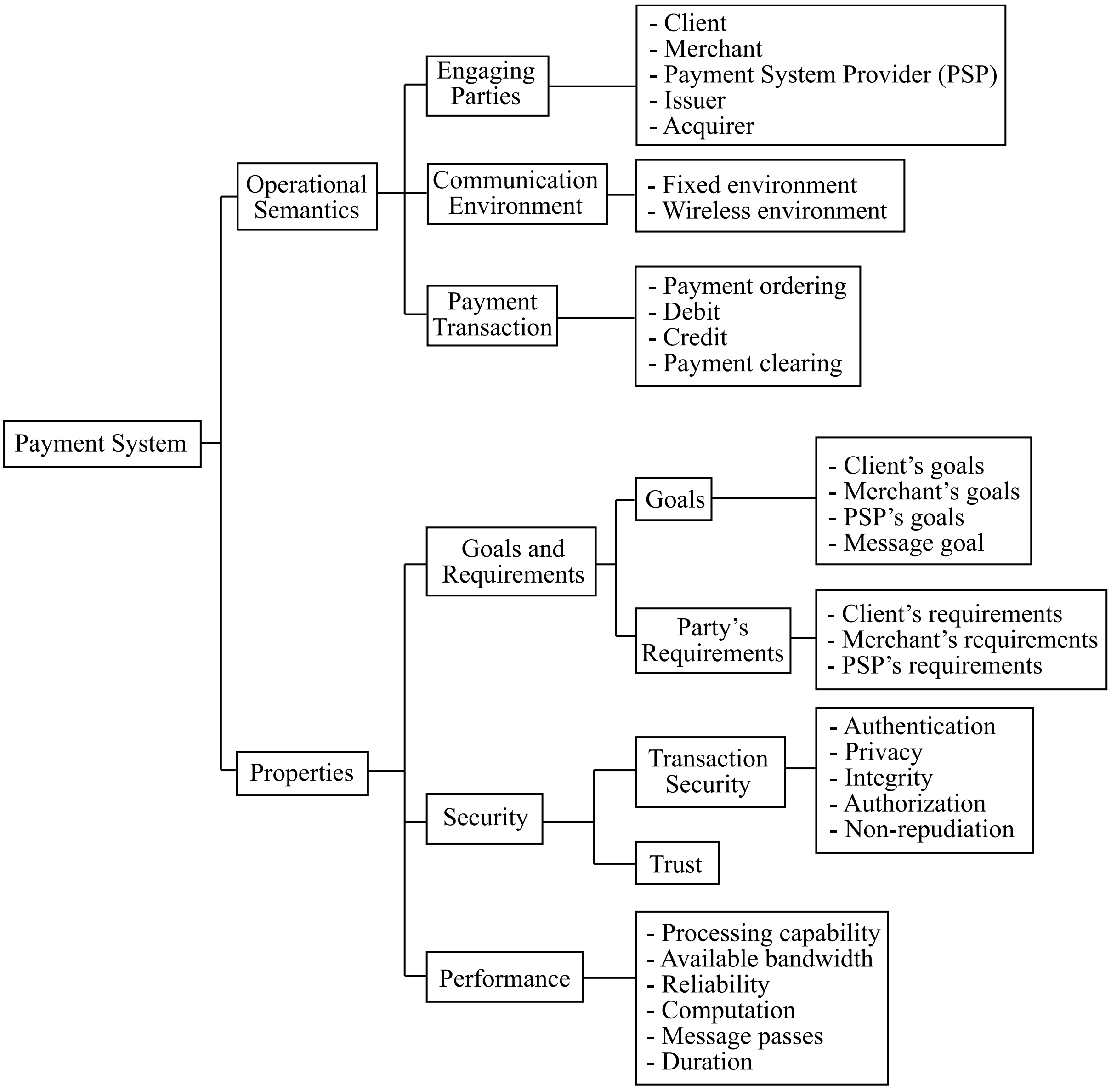


Figure 3.1: Overview of the proposed model

merchant M. C acts as a payer in S. C holds authorized payment information issued from I.

- M stands for the merchant establishing an account with an acquirer (M's financial institution) A. M has authorization from a payment system provider PSP to perform transactions in S. M acts as a payee in S.
- PSP stands for the payment system provider performing payment transactions on behalf of an issuer (C's financial institution) I and the acquirer A on the Internet side and on behalf of C and M on the banking private network side. The function of PSP may be operated by a credit-card company, a mobile operator, or a bank.
- I and A stand for the issuer and acquirer performing payment clearing as a result of the request from PSP. The payment clearing is performed under a banking private network.

3.1.3 Communication Environment

Definition 3.3 (Communication environment) *Communication environment CE is defined as the following set:*

$$CE = \{G, D, N\} \tag{3.2}$$

In CE, the engaging parties in G, using a set of Internet-accessible payment devices D, communicate to one another over a set of communication networks N. Generally, CE can be classified into fixed environment FE and wireless environment WE, where

$$CE = FE \cup WE \tag{3.3}$$

Fixed Environment: the fixed environment FE is defined as the following set:

$$FE = \{G, FD, FN\} \tag{3.4}$$

where $FE \subseteq CE$, $FD \subseteq D$, and $FN \subseteq N$. Note that FD stands for a set of Internet-accessible payment devices operating over a fixed network FN held by engaging parties in G. The examples of FD are personal computers (PCs) and Internet KIOSKs. FN represents a set of communication infrastructure which is composed of a number of wired links among the devices in FD. The examples of FN are Local Area Networks (LANs) and Wide Area Networks (WANs).

Wireless Environment: the wireless environment WE is defined as the following set:

$$WE = \{G, WD, WN\} \tag{3.5}$$

where $WE \subseteq CE$, $WD \subseteq D$, and $WN \subseteq N$. Note that WD stands for a set of Internet-accessible wireless payment devices e.g. cellular phones or Personal Digital Assistants (PDAs), held by engaging parties in G.

From the definitions of fixed and wireless devices described above, we can state the relationship between them as follows:

$$D = FD \cup WD \tag{3.6}$$

WN represents a set of wireless communication infrastructure in which the members in G communicate to one another. WN is composed of a number of wireless links among the devices in WD. The examples of WN are wireless LANs and cellular networks.

From the definitions of fixed and wireless networks described above, we can state the following relationship between them:

$$N = FN \cup WN \tag{3.7}$$

3.1.4 Payment Token

Definition 3.4 (Payment token) *A payment token T is a form of electronic money that is transferred among engaging parties in G in a payment system S. T represents physical money in traditional payment transactions. T has the following properties:*

1. *T is valid if T can be proven by I.*
2. *A party Q accepts T sent from a party R if T is proven valid.*

Referred to various kinds of payment systems outlined in chapter 2, the examples of T are electronic coins in micropayment systems, such as c_i in PayWord [RS96] and w_i in PayFair [Yen01], and $Price$ in SET [Mas97] and iKP [BGH$^+$00] protocols.

3.1.5 Payment Information

Definition 3.5 (Payment information) *Payment information PI refers to the validity of a payment token T to its issuer I. PI has the following properties:*

1. *Any party's PI is known only to her financial institution and herself.*

- *PI_C is known only to I and C.*
- *PI_M is known only to A and M.*

2. *Any party's PI is valid if it can be proven by her financial institution.*

3. *T is valid if PI is proven valid.*

Referred to the payment systems outlined in chapter 2, the examples of PI are *PayWord certificate* in PayWord and PI which contains credit-card information of a client in SET and iKP protocols. An issuer can use this information to identify its clients.

3.1.6 Payment Transaction

Definition 3.6 (Payment transaction) *A payment transaction PT is defined as a set of actions ACT regarding fund transfer performed by engaging parties in G over a set of communication networks N. PT can be represented as the following:*

$$PT = \{G, D, ACT, N\} \tag{3.8}$$

In other words, from the equations 3.2 and 3.8, it can be seen that the payment transaction PT can be represented by a number of actions in ACT regarding payment token transfer in the communication environment CE.

$$PT = \{CE, ACT\} \tag{3.9}$$

ACT is defined as the following set,

$$ACT = \{PO, DB, CD, PC\} \tag{3.10}$$

where, PO stands for *Payment Ordering*, DB stands for *Debit*, CD stands for *Credit*, and PC stands for *Payment Clearing*.

Payment Ordering PO is the interaction between C and M in that C requests to purchase goods or services from M. PO can be represented as follows:

$$PO \models \neg(\textit{payment-order}(C, M, T_C) \leftrightarrow \textit{payment-order}(M, C, T_C)) \tag{3.11}$$

where,

- *payment-order*(C, M, T_C): C requests M to purchase goods or services with the amount T_C, where T_C is the payment token requested by C.
- *payment-order*(M, C, T_C): M responses C's request regarding the order with the amount T_C.

Note that the above function does not contain the information about goods descriptions because we focus only on money transfer. However, it is not hard to add goods descriptions as one of the variables of the above function.

Debit DB is the interaction between I (through PSP) and C regarding the deduction of the payment token T_C requested by C from C's account. T_C will later be paid to the corresponding merchant M. DB can be represented as the following:

$$DB \models \neg(\textit{debit}(C, I, T_C) \leftrightarrow \textit{debit}(I, C, T_C)) \tag{3.12}$$

where,

- *debit*(C, I, T_C): C requests I (through PSP) to deduct the amount T_C from C's account. PSP will later forward the request to I which is located under a banking private network.

- $debit(I, C, T_C)$: I (through PSP) deducts or commits to deduct the amount T_C from C's account.

Note that from the Definition 3.2, I communicates with C through PSP. However, as stated in the Definition 3.5, PI_C must be known only to C and I. Thus, it is possible to replace I with PSP in equation 3.12 with the condition that PI_C must not be revealed to PSP.

$$DB \models \neg(debit(C, PSP, T_C) \leftrightarrow debit(PSP, C, T_C)) \tag{3.13}$$

Note also that the money deduction process varies among payment systems. For example, in prepaid payment system, the value of C's payment token is deducted from C's account before C is eligible to make the payment to M, whereas in postpaid one, C is notified that she will be deducted after the completion of the transaction.

Credit CD is the interaction between A (through PSP) and M in order to transfer the payment token T_M to M's account. CD can be represented as the following:

$$CR \models \neg(credit(M, A, T_M) \leftrightarrow credit(A, M, T_M)) \tag{3.14}$$

where,

- $credit(M, A, T_M)$: M requests A (through PSP) to transfer the amount T_M to M's account. PSP later forwards the request to A.
- $credit(A, M, T_M)$: A (through PSP) transfers or commits to transfer the amount T_M to M's account. It is considered as a confirmation that M will receive the money after the completion of the transaction.

As A communicates with M through PSP (from the Definition 3.2), it is possible to replace A with PSP in the equation 3.14. However, PI_M must not be revealed to PSP according to the Definition 3.5.

$$CR \models \neg(credit(M, PSP, T_M) \leftrightarrow credit(PSP, M, T_M)) \tag{3.15}$$

Payment Clearing PC is the interaction between I and A to transfer the amount requested by C and M between their accounts. Normally, this type of transaction is performed under a banking private network. PC is represented by the following function:

$$\textit{payment-clearing}(I, A, C, M, T_C, T_M)$$

where $T_C \geq T_M$. According to the above function, the amount T_C is transferred from C's account to M's account by I and A. PC can be derived from the following formula:

$$\begin{aligned} & debit(I, C, T_C) \wedge credit(A, M, T_M) \\ & \rightarrow \textit{payment-clearing}(I, A, C, M, T_C, T_M) \end{aligned} \tag{3.16}$$

From the above statement, *Payment Clearing* will complete if I has deducted the payment token T_C from C, and A has transferred the token T_M to M. Moreover, it can be seen that I and A in this statement can be replaced by PSP under the same condition as that of the formulae 3.13 and 3.15:

$$\begin{aligned} & debit(PSP, C, T_C) \wedge credit(PSP, M, T_M) \\ & \rightarrow \textit{payment-clearing}(PSP, C, M, T_C, T_M) \end{aligned} \tag{3.17}$$

Figure 3.2 demonstrates the directions of payment token transfer. *Payment Ordering* represents payment token transfer from C to M, even though, in fact, the actual transfer of payment token is processed from I to A in *Payment Clearing*. *Debit* represents the deduction of the value of the payment token from C to I, although, in fact, this action is performed at I itself because C's account has been established with I. *Credit* represents money deposit from A to M even though, actually, this action is performed at A itself because M has an account established with A.

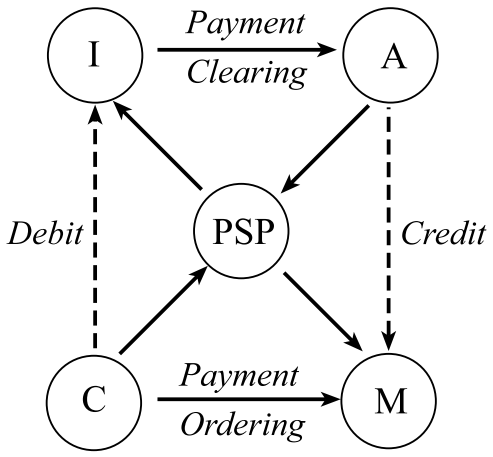


Figure 3.2: Payment transaction

Note that, from Figure 3.2, the arrows represent the directions of actual transactions. As PSP acts as a medium between *C-M* and *I-A*, all actual transactions relevant to *Debit* and *Credit* are performed through PSP. The broken arrows represent the directions of actions performed by originators of the actions to their intended recipients.

3.1.7 Goals of Engaging Parties for Payment Transactions

On designing a payment protocol, we focus on the information related to a payment transaction that should be transferred to engaging parties at the end of the transaction. This information must deliver its sender's intention and purpose regarding the payment transaction to its intended recipient. Such information is considered as goals of engaging parties.

We provide reasoning about the goals of engaging parties by using an accountability logic proposed by Kungpisdan *et al.* (KP) [KP02]. KP's logic not only reasons about provability of payment transactions, performed by engaging parties, to a verifier, but it also infers sending and receiving information relevant to fund transfer. We deploy modal operators in KP's logic to state the goals of engaging parties that contain payment token T, payment information PI, and identities of engaging parties in particular payment transactions.

Based on the notations similar to the ones in [Her01, KP02], the following modal operators are used throughout this thesis:

- *Q authorized X*: a party Q has authorization on performing an action X, where $X \in ACT$.
- *Q CanProve X to R*: a party Q is able to prove to a party R that the statement X is true without revealing any information which is considered to be secret to R.

Referred to the Definition 3.2, PSP acts on behalf of I and A over the Internet side. Thus, in the proposed mobile payment model, we focus on three main parties communicating to one another over the Internet: C, M, and PSP. Based on SET protocol requirements analyzed by Meadows *et al.* [MS98], the following definition presents the goals of engaging parties regarding the payment transaction.

Definition 3.7 (Goals of engaging parties) *Goals of engaging parties regarding a payment transaction* ($Goals$) *are defined as the following set:*

$$Goals = \{CG, MG, PSPG\} \tag{3.18}$$

Client's Goal (CG): C can ensure that M has delivered or committed to deliver the goods or services requested by C.

C CanProve (M authorized payment-order(M, C, T_C)) to V

From the above statement, C must be able to prove to a verifier V, who does not involve in the transaction, that M has authorized the transaction regarding *Payment Ordering* which has been requested by C. Such authorization may be contained in the message sent to C. This message or its parts must be provable that it has been originated by M and it has C as its intended recipient. Moreover, this message must contain authorized amount T_C as a receipt of the payment to C.

Merchant's Goal (MG): M can ensure that A has transferred or committed to transfer the amount equivalent to T_M to M.

M CanProve (PSP authorized credit(PSP, M, T_M)) to V

From the above statement, M must be able to prove to a verifier V that PSP authorized the transaction regarding *Credit* which has been requested by M. In other words, M has to receive the message originated by PSP (on behalf of A) and the message must contain the amount T_M authorized by PSP.

Payment System Provider's Goal ($PSPG$): PSP, on behalf of I and A, has successfully performed *Payment Clearing.*

R CanProve (

PSP authorized payment-clearing(PSP, C, M, T_C, T_M) to V where,

$$\begin{aligned}
&\textit{C CanProve (PSP authorized debit(PSP, C, } T_C\textit{)) to V} \wedge \\
&\textit{M CanProve (PSP authorized credit(PSP, M, } T_M\textit{)) to V} \\
&\rightarrow \textit{P CanProve (PSP authorized payment-clearing(} \\
&\qquad\qquad\qquad\qquad \textit{PSP, C, M, } T_C, T_M\textit{)) to V}
\end{aligned} \qquad (3.19)$$

where P stands for any party. It can be seen that achieving PSP's goal cannot be proven by only PSP itself, but by the cooperation with C and M. To achieve this goal, PSP has to collect the results of two proofs performed by C and M; one is performed by C to prove that PSP has deducted or committed to deduct the amount T_C requested by C and the other is M's goal (MG).

Moreover, based on [KP02], we define the goal of actions which are performed in each protocol message as follows:

Definition 3.8 (Message goal) *Each message transferred in a protocol should deliver its sender's intention regarding an action in ACT regarding a payment transaction PT to its intended recipient.*

$$Q \textit{ CanProve } (\; Q \textit{ authorized act}(Q, R, X) \;) \textit{ to } R$$

where Q and R stand for any party and X stands for the message component containing Q's intention regarding performing the action act, where act $\in ACT$. Note that X may include T_Q and PI_Q.

As a message goal, $PSPG$ can be represented as the following:

$$\begin{aligned}
&\textit{PSP CanProve (PSP authorized debit(PSP, C, } T_C\textit{)) to C} \wedge \\
&\textit{PSP CanProve (PSP authorized credit(PSP, M, } T_M\textit{)) to M} \\
&\rightarrow \textit{PSP CanProve (PSP authorized payment-clearing(} \\
&\qquad\qquad\qquad \textit{PSP, C, M, } T_C\textit{, } T_M\textit{)) to Q}
\end{aligned} \tag{3.20}$$

where R stands for C or M. To achieve this goal, PSP has to prove that the message originated by itself contains necessary information to be used as evidence to prove to C and M regarding its tasks. Such evidence must contain at least the provable identities of both the sender and the intended recipient of the message and authorized amount T_C and T_M. In this case, it can be seen that the verifiers are internal parties who involve in the transaction.

3.1.8 Party's Requirements for Payment Transactions

In the previous section, we can see that the goals of engaging parties and the message goal are required to be achieved at design stage. However, at implementation stage, we cannot avoid disputes which may occur from the malicious behaviors of dishonest parties or attackers e.g. modifying protocol messages. Therefore, we need to define occurrences and properties of the messages transferred in the transaction to reason about the completion of payment transaction in that, at the end of the transaction, all engaging parties should agree with the results of the actions regarding the transaction relevant to them. In particular, a payment transaction is considered to be completed if each engaging party agrees and satisfies the results of the actions relevant to her. Otherwise, the transaction is not completed.

We define a modal operator '*completes*' to reason about the completion of a transaction in which the particular parties agree with the results of the transaction.

$$Q \textit{ completes } X$$

where $X \in ACT$. The statement above states that a party Q agrees with the result of a transaction relevant to the actions in ACT that are related to her. In particular, C completes the transaction after she receives goods or services as a result of sending a payment token T_C to M. M completes the transaction after she has delivered or committed to deliver goods or services as a result of receiving T_M. PSP completes the transaction after it has transferred the payment token from C's account to M's account as a result of the $Debit$ and $Credit$ requests from C and M, respectively.

Definition 3.9 (Party's requirements for payment transactions) *The party's requirements PR are defined as the following set:*

$$PR = \{CReq, MReq, PSPReq\} \tag{3.21}$$

The details of the party's requirements are given as the following:

Client's Requirements ($CReq$): the requirements that must be satisfied in C's point of view regarding a payment transaction can be presented as follows:

$$\begin{aligned}
&\textit{M CanProve (C authorized payment-order(C, M, } T_C \textit{)) to V} \wedge \\
&\textit{PSP CanProve (C authorized debit(C, PSP, } T_C \textit{)) to V} \wedge \\
&\textit{C CanProve (PSP authorized debit(PSP, C, } T_C \textit{)) to V} \wedge \\
&\textit{C CanProve (M authorized payment-order(M, C, } T_C \textit{)) to V} \\
&\rightarrow \textit{C completes payment-ordering(C, M, } T_C \textit{)}
\end{aligned} \tag{3.22}$$

It can be seen that, to complete a transaction in C's point of view, C not only has to receive payment responses regarding *Payment Ordering* and *Debit* from M and PSP (on behalf of I), respectively, but such responses must be the responses of the requests previously made by C. In other words, the *Payment Ordering* response from M must contain the amount T_C requested by C, and the *Debit* response from PSP must contain the amount T_C that C has requested I to deduct from her account.

Note that the above statement does not reason about goods descriptions because the model focuses only on the accountability of price to ensure that the correct amount has been transferred between C and M.

Merchant's Requirements ($MReq$): to complete a transaction in M's point of view, the following requirements must be satisfied:

$$\begin{aligned}
&\textit{M CanProve (C authorized payment-order}(C, M, T_C)) \textit{ to V} \wedge \\
&\textit{PSP CanProve (M authorized credit}(PSP, M, T_M, PI_M)) \textit{ to V} \wedge \\
&\textit{M CanProve (PSP authorized credit}(PSP, M, T_M, PI_M)) \textit{ to V} \wedge \\
&\textit{C CanProve (M authorized payment-order}(M, C, T_C)) \textit{ to V} \\
&\rightarrow \textit{M completes credit}(M, PSP, T_M)
\end{aligned} \tag{3.23}$$

Primarily, M completes a transaction if she has delivered or committed to deliver goods or services to C as a result of receiving *Payment Ordering* request from C. However, before delivering goods or services to C, the payment has to be processed by PSP, and PSP must transfer the amount T_M requested by M to M's account.

Payment System Provider's Requirements ($PSPReq$): to complete a transaction in PSP's point of view, the following requirements must be satisfied:

$$\begin{aligned}
&\textit{PSP CanProve (C authorized debit(C, PSP, } T_C\textit{)) to V} \wedge \\
&\textit{PSP CanProve (M authorized credit(M, PSP, } T_M\textit{)) to V} \wedge \\
&\textit{R CanProve (PSP authorized payment-clearing(} \\
&\qquad\qquad\qquad\qquad PSP, C, M, T_C, T_M\textit{)) to V} \\
&\rightarrow \textit{PSP completes payment-clearing(PSP, C, M, } T_C, T_M\textit{)}
\end{aligned} \tag{3.24}$$

Note that R stands for any party. It can be seen that from the above statement, PSP completes *Payment Clearing* if it has performed *Payment Clearing* as a result of the requests from C and M. In particular, PSP must perform the actions as the responses of *Debit* and *Credit* requests made by both C and M in that PSP transfers or commits to deduct the amount T_C requested by C and transfer the amount T_M requested by M to M's account.

From the statements 3.22 to 3.24, we can establish the relationship between *Goals* and PR as follows:

$$Goals \subset PR$$

In particular,

$$\begin{aligned}&\textit{C completes payment-order(C, M, } T_C\textit{)}\\&\rightarrow \textit{C CanProve (M authorized payment-order(M, C, } T_C\textit{)) to V}\end{aligned} \tag{3.25}$$

$$\begin{aligned}&\textit{M completes credit(PSP, M, } T_M\textit{)}\\&\rightarrow \textit{M CanProve (PSP authorized credit(PSP, M, } T_M\textit{)) to V}\end{aligned} \tag{3.26}$$

$$\begin{aligned}&\textit{PSP completes payment-clearing(PSP, C, M, } T_C, T_M\textit{)}\\&\rightarrow \textit{R CanProve (PSP authorized payment-clearing(}\\&\qquad\qquad \textit{PSP, C, M, } T_C, T_M\textit{)) to V}\end{aligned} \tag{3.27}$$

where R stands for any party.

Satisfying Party's Requirements for Payment transactions

We can see that, on one hand, party's requirements PR capture the whole occurrences, including properties, of actions relevant to engaging parties during a payment transaction. On the other hand, the goals of each party *Goals* capture only the occurrences and properties of the results that each party has been expecting to receive at the end of the transaction as a result of their requests regarding the payment transaction.

For example, at the end of a transaction, a client who receives the response of her *Payment Ordering* request may not consider to complete the transaction if the response does not contain the information that she has been expecting e.g. the amount deducted by the client's issuer is higher than the amount requested by the client. To satisfy the client's requirement, engaging parties are required to provide the information regarding the actions relevant to the client.

3.1.9 Transaction Security

Definition 3.10 (Transaction security) *A payment system S should satisfy the following set of transaction security properties $TSec$:*

$$\begin{aligned} TSec = \{ & Party\ Authentication,\ Transaction\ Privacy, \\ & Transaction\ Integrity,\ Transaction\ Authorization, \\ & Non\text{-}repudiation\ of\ Transactions \end{aligned} \tag{3.28}$$

where,

- **Party Authentication:** any party is able to authenticate herself to other parties.
- **Transaction Privacy:** any message sent in the payment system is readable by only the intended recipient of the message.
- **Transaction Integrity:** any party can ensure that the content of a message will not be altered during a transmission.
- **Transaction Authorization:** no unauthorized payment transaction can be successfully made.
- **Non-repudiation of Transactions:** any party cannot deny the transactions she has performed.

In other words, a payment transaction does not have to satisfy the above transaction security properties. However, the payment system is not considered to be secure if it lacks of any of the above transaction security properties according to the discussion in section 3.1.1.

3.1.10 Trust Relationships among Engaging Parties

Definition 3.11 (Trust relationships) *In any payment system S, trust relationships among engaging parties ($Trust$) are established as the following:*

1. *I and A are trusted by C and M, respectively, that they will not reveal confidential information, including PI, of C and M to other parties.*

2. *G_i, where $G_i \in G$, is partially trusted by G_j, where $G_j \in G$ and $G_j \neq G_i$, in that:*

    - *G_i performs its tasks regarding PT upon the requests from G_j.*
    - *It is possible that PSP impersonates as C and requests for Payment Ordering to M.*
    - *It is possible that M impersonates as C and requests for Debit to PSP.*

3.1.11 Transaction Performance

Definition 3.12 (Transaction performance) *Transaction performance of a payment system S (TP_S) is defined as the following function:*

$$TP_S(processing, bandwidth, reliability, computation, messages, duration)$$

where each variable of TP_S is the following functions:

- *processing*(D, PT): total processing capability at a set of devices D held by a set of engaging parties G on performing a payment transaction PT.
- *bandwidth*(N): available bandwidth of a set of communication networks N during the transaction PT.
- *reliability*(N): reliability of N.
- *compute*(D, PT): total computation at D held by the engaging parties in G on performing PT.
- *messages*(G, PT): the number of message passes among G to complete PT.

- $duration(G, PT)$: duration to complete PT among G.

In this section, we investigate how each of the above functions affects transaction performance of the payment system. To do so, we introduce the following symbols to represent the measurement of each function:

- $func(X) > func(Y)$: executing an input X in a function $func$ can produce higher output than executing Y in $func$.
- $func(X) \leq func(Y)$: executing an input X in a function $func$ can produce no higher output than executing Y in $func$.

Considering two payment systems, S and S', in S, a set of engaging parties G using a set of devices D performs a payment transaction PT over a set of networks N. In S', a set of engaging parties G' using a set of devices D' performs a payment transaction PT', where $PT' = PT$, over a set of networks N'. In other words, the parties in G and G' perform the same payment protocol in S and S', respectively. The comparisons between these two payment systems in terms of the transaction performance are shown as the following:

1. Performing a payment transaction among higher processing capability devices takes less duration than lower ones.

$$\begin{aligned}
&processing(D, PT) > processing(D', PT') \ \wedge \\
&bandwidth(N') \leq bandwidth(N) \ \wedge \\
&reliability(N') \leq reliability(N) \ \wedge \\
&compute(D, PT) \leq compute(D', PT') \ \wedge \\
&messages(G, PT) \leq messages(G', PT') \\
&\rightarrow duration(G', PT') > duration(G, PT)
\end{aligned} \tag{3.29}$$

2. Performing a payment transaction in higher available bandwidth or more reliable network takes less duration than lower one.

$$\begin{aligned}
&bandwidth(N) > bandwidth(N') \ \wedge \\
&processing(D', PT') \leq processing(D, PT) \ \wedge \\
&reliability(N') \leq reliability(N) \ \wedge \\
&compute(D, PT) \leq compute(D', PT') \ \wedge \\
&messages(G, PT) \leq messages(G', PT') \\
&\rightarrow duration(G', PT') > duration(G, PT)
\end{aligned} \tag{3.30}$$

$$\begin{aligned}
&reliability(N) > reliability(N') \ \wedge \\
&bandwidth(N') \leq bandwidth(N) \ \wedge \\
&processing(D', PT') \leq processing(D, PT) \ \wedge \\
&compute(D, PT) \leq compute(D', PT') \ \wedge \\
&messages(G, PT) \leq messages(G', PT') \\
&\rightarrow duration(G', PT') > duration(G, PT)
\end{aligned} \tag{3.31}$$

3. Performing higher computational payment transaction takes more duration than lower one.

$$\begin{aligned}
&compute(D', PT') > compute(D, PT) \ \wedge \\
&reliability(N') \leq reliability(N) \ \wedge \\
&bandwidth(N') \leq bandwidth(N) \ \wedge \\
&processing(D', PT') \leq processing(D, PT) \ \wedge \\
&messages(G, PT) \leq messages(G', PT') \\
&\rightarrow duration(G', PT') > duration(G, PT)
\end{aligned} \tag{3.32}$$

4. Performing a payment transaction with higher number of message passes

takes more duration than the lower ones.

$$\begin{aligned}
&messages(G', PT') > messages(G, PT) \ \wedge \\
&compute(D, PT) \leq compute(D', PT') \ \wedge \\
&reliability(N') \leq reliability(N) \ \wedge \\
&bandwidth(N') \leq bandwidth(N) \ \wedge \\
&processing(D', PT') \leq processing(D', PT) \\
&\rightarrow duration(G', PT') > duration(G, PT) \qquad (3.33)
\end{aligned}$$

5. Less duration of a payment transaction implies higher transaction performance.

$$\begin{aligned}
&duration(G', PT') > duration(G, PT) \\
&\rightarrow TP_S > TP_{S'} \qquad (3.34)
\end{aligned}$$

3.2 Practical and Secure Mobile Payment

In the previous section, we have presented a general electronic payment model. In this section, a formal model for a practical and secure mobile payment system is introduced.

3.2.1 Practical and Secure Mobile Payment System

A payment system S is considered to be practical and secure if, in addition to satisfying basic infrastructure $\{G, CE, PT\}$, S must satisfy essential properties mentioned in the Definition 3.1. Moreover, its transaction performance must be acceptable by engaging parties in G. We first present a practical and secure payment system in the Definition 3.13. Then a practical and secure mobile payment system is defined in the Definition 3.15.

Definition 3.13 (Practical and secure payment system) *A practical and secure payment system PSS is presented as the following set:*

$$PSS = \{G, CE, PT, Goals, PR, TSec, Trust\} \tag{3.35}$$

where $ATP \leq TP_{PSS}$. ATP stands for acceptable transaction performance.

It can be seen that a payment system S may not be considered as PSS if it lacks of any of the members of PSS. Particularly, PSS must satisfy goals $Goals$ and requirements of engaging parties PR, transaction security properties $TSec$ and trust among engaging parties $Trust$, and must have acceptable transaction performance ATP.

Definition 3.14 (Acceptable transaction performance) *Transaction performance of a payment system S (TP_S) is considered to be acceptable by G if $TP_S \geq ATP$, where ATP is acceptable transaction performance specified by the engaging parties in G.*

$$S \textit{ is acceptable iff } TP_S \geq ATP$$

Example: a company wants to set up a payment system S for its electronic commerce department by specifying the following requirements:

1. It is able to operate on the company's existing network N and set of devices D with the processing capability $Processing_{ATP}$.
2. Each transaction must be completed within the duration $Duration_{ATP}$.

Thus, ATP is represented as the following function:

$$ATP(Processing_{ATP}, bandwidth, reliability, compute,$$
$$messages, Duration_{ATP})$$

A payment system S is able to operate on the same set of devices D and network N with the processing capability $Processings_S$, where $Processings_S > Processing_{ATP}$, and it takes the duration $Durations_S$, where $Durations_S < Duration_{ATP}$, to complete each transaction. The transaction performance of S (TP_S) is shown as follows:

$$TP_S(Processings_S, bandwidth, reliability, compute,$$
$$messages, Durations_S)$$

As $Processings_S > Processing_{ATP}$ and $Durations_S < Duration_{ATP}$, we can infer that $TP_S > ATP$. Therefore, the transaction performance of the payment system S is acceptable by the company.

Definition 3.15 (Practical and secure mobile payment system) *A practical and secure mobile payment system $PSMS$, where $PSMS \subseteq PSS$, is defined as follows:*

$$PSMS = \{G, CE', PT, Goals, PR, TSec, Trust\} \tag{3.36}$$

where $CE' = \{WE, FE\}$ and $WE \neq \phi$. $ATP \leq TP_{PSMS}$

In $PSMS$, the communication environment CE' must consist of at least the wireless environment WE. The parties in G may communicate to one another either over only wireless networks or over both fixed and wireless networks.

Note that the above mobile payment system satisfies the requirements for a practical and secure mobile payment system stated in section 2.1 because it satisfies the trust relationships among engaging parties and transaction security properties, and has the transaction performance which is acceptable by users in the system.

3.2.2 Wireless vs Fixed Environments

Wireless environment WE has a number of disadvantages compared to fixed environment FE. Such disadvantages mainly come from the characteristics of wireless device WD and wireless network WN in that,

1. While performing the payment transaction PT, WD has lower processing capability than FD.

$$processing(FD, PT) > processing(WD, PT)$$

2. WN has lower available network bandwidth than FN. Also, WN is less reliable than FN.

$$bandwidth(FN) > bandwidth(WN)$$
$$reliability(FN) > reliability(WN)$$

Example: considering two payment systems, S_1 and S_2, in S_1, the engaging parties in G_{WE} using the set of wireless devices WD perform the transaction PT over a set of wireless networks WN. In S_2, the engaging parties in G_{FE} using the set of fixed devices FD perform the transaction PT', where $PT = PT'$, over a set of fixed networks FN. The transaction performance of the above payment systems can be formalized as follows:

$$processing(FD, PT') > processing(WD, PT)$$
$$bandwidth(FN) > bandwidth(WN)$$
$$reliability(FN) > reliability(WN)$$

From the statement 3.29, we can derive,

$$\begin{aligned}
&processing(FD, PT') > processing(WD, PT) \ \wedge \\
&bandwidth(WN) \leq bandwidth(FN) \ \wedge \\
&reliability(WN) \leq reliability(FN) \ \wedge \\
&compute(WD, PT) \leq compute(FD, PT') \ \wedge \\
&messages(G_{WE}, PT) \leq messages(G_{FE}, PT') \\
&\rightarrow duration(G_{WE}, PT') > duration(G_{FE}, PT) \qquad (3.37)
\end{aligned}$$

Then, from the statement 3.37, we can derive,

$$\begin{aligned}
&duration(G_{WE}, PT) > duration(G_{FE}, PT') \\
&\rightarrow TP_{S_2} > TP_{S_1} \qquad (3.38)
\end{aligned}$$

3.2.3 Cryptographic Operations

In this section, we discuss several kinds of available cryptographic operations in terms of how they affect transaction performance of an electronic payment system.

Considering two payment systems, S and S', in S, a set of engaging parties G using a set of devices D performs a payment transaction PT. In S', another set of engaging parties G' using a set of devices D' performs a payment transaction PT'. We introduce the following functions to represent cryptographic operations applied to payment transactions:

- $asymmetric(PT)$: the payment transaction PT that deploys asymmetric cryptographic operations.
- $symmetric(PT)$: the payment transaction PT that deploys symmetric cryptographic operations.
- $hash(PT)$: the payment transaction PT that deploys hash functions.

Regarding the computation of cryptographic operations, we can see that:

1. Asymmetric operation requires higher computation than symmetric one.

$$compute(D, asymmetric(PT)) > compute(D', symmetric(PT'))$$

2. Symmetric operation requires higher computation than hash function.

$$compute(D, symmetric(PT)) > compute(D', hash(PT'))$$

3. Therefore, we can infer that asymmetric operation requires higher computation than hash function.

$$compute(D, asymmetric(PT)) > compute(D', hash(PT'))$$

From the statements 3.32 and 3.34, we can derive the following conclusions:

$$\begin{aligned}&compute(D, asymmetric(PT)) > compute(D, symmetric(PT'))\\ &\rightarrow TP_{S'} > TP_S\end{aligned} \tag{3.39}$$

$$\begin{aligned}&compute(D, symmetric(PT)) > compute(D', hash(PT'))\\ &\rightarrow TP_{S'} > TP_S\end{aligned} \tag{3.40}$$

$$\begin{aligned}&compute(D, asymmetric(PT)) > compute(D', hash(PT'))\\ &\rightarrow TP_{S'} > TP_S\end{aligned} \tag{3.41}$$

As a result, applying symmetric operations to a payment system results in higher transaction performance than applying asymmetric ones. Moreover, applying hash functions leads to higher performance than both asymmetric and symmetric operations.

3.2.4 Symmetric vs Asymmetric Cryptography for Payment Transactions

Symmetric and asymmetric cryptography are normally used to secure communications among engaging parties. Symmetric cryptography employs secret keys shared among engaging parties, whereas asymmetric cryptography deploys a pair of private/public keys.

Definition 3.16 (Transaction security of symmetric cryptography) *Referred to $TSec$ stated in the Definition 3.10, symmetric cryptography satisfies the following properties:*

1. *Message Confidentiality*
2. *Message Integrity*
3. *Party Authentication*
4. *Transaction Authorization*

Definition 3.17 (Transaction security of asymmetric cryptography) *Based on $TSec$ presented in the Definition 3.10, asymmetric cryptography satisfies the following properties:*

1. *Message Confidentiality*
2. *Message Integrity*
3. *Party Authentication*
4. *Transaction Authorization*
5. *Non-repudiation of Transactions*

According to non-repudiation property, each engaging party must not be able to deny the transaction she has performed. To achieve this property, the recipient of each message must have the ability to identify the originator of the message which can be achieved by using digital signature in asymmetric cryptography. However, in symmetric cryptographic protocols, we cannot prove the originator of encrypted message because the secret key is shared among engaging parties [Kai96].

Non-repudiation property is very important for financial transactions that are relevant to fund transfers and goods ordering. Such activities require non-repudiation services to resolve disputes among parties. Non-repudiation is implicitly satisfied by accountability property [KP02].

3.2.5 Fixed-network Payment Protocols in Wireless Environments

Normally, payment protocols for fixed networks [Mas97, BGH$^+$00, RS96] deploy asymmetric-key operations. We compare applying a fixed-network payment protocol to a wireless network with a wireless-network payment protocol in terms of transaction performance. From the statements 3.39, 3.40, and 3.41, we can infer that,

$$\begin{aligned}
&compute(D, asymmetric(PT)) > compute(D', symmetric(PT')) \ \vee \\
&compute(D, symmetric(PT)) > compute(D', hash(PT')) \ \vee \\
&compute(D, asymmetric(PT)) > compute(D', hash(PT')) \\
&\rightarrow TP_{S'} > TP_S
\end{aligned} \tag{3.42}$$

From the above statement, we can see that the transaction performance increases if we apply symmetric-key operations and hash functions to the payment transaction instead of asymmetric-key operations. However, from the Definition 3.16, symmetric-key operations lack non-repudiation property that is satisfied by asymmetric-key ones.

3.3 Analyzing a Payment System Based on The Proposed Model

In previous sections, we have introduced a formal model for a practical and secure payment system. The proposed model not only focuses on reasoning about mobile payment systems, but it can also be used to analyze fixed-network payment systems. In this section, we briefly demonstrate how the proposed model is used to analyze a payment system. Later in this thesis, our mobile payment framework and payment protocols will be proposed. The detailed analyses of our proposed mobile payment systems based on the proposed formal model will be presented.

Given a system S, from the Definition 3.15, S will be considered as a practical and secure mobile payment system if it satisfies the following parameters: $G, CE, PT, Goals, PR, TSec$, and $Trust$. Moreover, TP_S must be greater than or equal to ATP. From the above conditions, it can be seen that:

1. A payment system is normally composed of $\{G, CE, PT\}$. Generally, S should satisfy G and CE. Then, we consider whether or not the transaction in S is related to fund transfer. If so, it satisfies PT stated by the model.

2. As mentioned in section 1.4, a payment system which satisfies accountability property will satisfy all fundamental security properties stated in $TSec$. We therefore analyze $TSec$ of S by using an accountability logic. The details of the logic will be presented in chapter 7.

3. To analyze $Goals$ and PR, it can be seen that in the Definition 3.7 and Definition 3.9, $Goals$ and PR can be formalized into proof statements of

Kungpisdan *et al.* (KP)'s accountability logic [KP02]. Therefore, if S is successfully analyzed by KP's logic, $Goals$ and PR will be satisfied.

4. To analyze $Trust$, we consider initial trust relationships among engaging parties in S whether they satisfy $Trust$ stated in the Definition 3.11. If they are not mentioned in the system, we consider each protocol step whether there is sensitive information of any party revealed to any unauthorized party.

5. To analyze TP_S, we first state ATP and then compare all related variables of S with those of ATP. The example of this comparison has been illustrated in section 3.2.1.

6. If all the above parameters are satisfied by S, it can be concluded that S is a practical and secure payment system.

3.4 Summary

In this chapter, we have shown that a mobile payment system is a subset of an electronic payment system whereby at least one engaging party performs a payment transaction over a wireless network using a wireless device. We have introduced a formal model for a practical and secure mobile payment system in that:

- It is practical if it provides transaction performance which is acceptable by users. In particular, a mobile user is the party that must be most concerned in the system.
- It is secure if it satisfies all necessary security-related properties.

Such security-related properties can be satisfied by applying cryptographic operations to the information transferred in the system. However, on one hand, applying high computational operations such as asymmetric-key operations may not be applicable to mobile users who hold low computational capability mobile devices. On the other hand, low computational symmetric-key operations lack non-repudiation which is an essential property for financial transaction.

In the next chapters, we will investigate several techniques that can be used to design practical and secure mobile payment systems by using the proposed model as a major guideline for evaluation.

Later in this thesis, from chapters 4 to 6, we will present a mobile payment framework and mobile payment protocols, and analyze primarily their transaction performance and trust relationships among engaging parties, whereas in chapter 7, transaction security, goals of engaging parties, and party's requirements of the proposed framework and protocols will be formally analyzed based on the proposed model.

Note that accountability property plays an important role in the proposed model. That is, it is used as a major analysis tool in our formal model to analyze a payment system on $Goals$, PR, and $TSec$. As previously discussed in section 1.4 that existing accountability logics [Kai96, KN98, KP02] are inadequate to analyze mobile payment protocols, in chapter 7, we will propose an accountability logic for mobile payment protocols and use it to analyze various kinds of mobile payment protocols.

Chapter 4

A Framework for Practical and Secure Mobile Payment

There have been several attempts to devise practical and secure solutions for mobile payment. As discussed in section 2.4.4, several frameworks have been proposed to enable payment transactions in wireless environments. In particular, proxy-based and agent-based frameworks aim to enable mobile payment transactions with the payment protocols originally designed for fixed networks, whereas non proxy-based framework aims to deploy the payment protocols which operate well in wireless environments without any assistance (either from proxy server or mobile agent).

In this chapter, we evaluate the existing frameworks based on the proposed formal mobile payment model presented in chapter 3 to show that the payment systems based on the existing frameworks are not considered as the practical and secure mobile payment systems stated in the formal model. We then present a new framework which satisfies the formal model. Our framework not only incorporates the main features of the existing frameworks, but also solves their problems. As SET protocol [Mas97] has been applied to both proxy-based and agent-based frameworks, we demonstrate the practical usefulness of the proposed framework by applying SET protocol to it and comparing it with the SET payment systems based on the existing frameworks.

This chapter is organized as follows. Section 4.1 discusses the reason why the existing mobile payment frameworks do not satisfy our formal model. Section 4.2 introduces our framework which satisfies the formal model. In section 4.3, we apply the proposed framework to enable mobile SET transactions. Section 4.4 discusses issues related to security and practicability of our framework. Section 4.5 summarizes the chapter.

4.1 Evaluation of Existing Frameworks

In this section, we examine existing mobile payment frameworks namely agent-based, proxy-based, and non proxy-based frameworks whether they satisfy the formal model presented in chapter 3. To provide concrete evaluation, the SET payment systems based on the existing frameworks will be used. Recall that several systems based on agent-based framework [RdS98, WLY99] and proxy-based framework [WSZ01] have been proposed by deploying the SET protocol as their underlying payment protocol. Sections 4.1.1 and 4.1.3 demonstrate the details of the evaluation.

4.1.1 Agent-Based Framework

Recall that SET protocol [Mas97] is one of the most well-known credit-card payment protocols introduced by Mastercard and VISA, two major credit-card companies. Although the SET protocol is successfully implemented in fixed-network environments, it is not easy to implement it in the wireless ones because of the nature of the SET protocol itself and the wireless environments. Note that the problems of SET protocol and the constraints of wireless environments have been previously discussed in sections 2.3.1 and 1.3.1, respectively.

To overcome such constraints, Romao *et al.* [RdS98] proposed an agent-based SET payment system (called SET/A). Note that the details of SET/A have been presented in section 2.4.2. Applying mobile agent technology to

SET/A, a mobile agent containing a SET wallet plays the client's role during the transaction. Thus, the client needs to connect to the Internet for short periods during the entire transaction. However, SET/A is vulnerable to attacks because the agent is required to bring the SET wallet with it to perform operations at a merchant's environment which is considered to be hostile.

Comparing to the formal model, it can be seen that SET/A does not satisfy the model in that it is prone to the lack of transaction privacy stated in the Definition 3.10 because of the key generation in the merchant's environment. Moreover, according to the trust relationships among engaging parties stated in the Definition 3.11 of the model, the merchant is not trusted by the client not to impersonate as the client to perform transactions.

Wang *et al.* [WLY99] proposed another agent-based SET payment system (called SET/A+) to solve the problems of SET/A. SET/A+ operates in a larger scenario than that of SET/A, in that, it includes brokering and negotiation phase, which naturally requires the capability of mobile agent, in SET protocol. A client's **PReq** (refer to the details of SET protocol descriptions presented in section 2.3.1) is completely generated at the client before it is brought with an agent to perform a payment transaction at a merchant. However, performing cryptographic operations, including public-key operations, at the client's mobile device results in the problem of high computational load which is not likely to satisfy acceptable transaction performance stated in the Definition 3.14 of the formal model.

4.1.2 Proxy-Based Framework

As well as agent-based framework, the payment systems based on proxy-based framework, such as 3D SET for Mobiles [WSZ01], was applied to enable payment transactions in wireless environments. Note that the details of 3D SET for Mobiles have been presented in section 2.4.1. In this section, we discuss a proxy-based SET payment system called 3D SET for Mobiles whether it satisfies the formal model. In this system, the function of proxy server located in a fixed network is operated by an issuer. The proxy server contains all payment-related information of its clients. To make a payment, a mobile client sends the issuer a request to perform a SET transaction on her behalf. However, the problem of trustworthiness of the issuer arises because the sensitive information of the client needs to be stored on the issuer's proxy server. The client is required to fully trust the issuer that it will not impersonate as the client to perform transactions with merchants.

It is obvious that this system does not satisfy the trust relationships among engaging parties stated in the Definition 3.11 of the formal model, that is, the partial trust relationship between the client and the issuer should be established. That is, the issuer is not trusted by the client not to impersonate as the client to perform transactions.

4.1.3 Non Proxy-Based Framework

Non proxy-based framework is out of scope of our consideration because the framework itself suffers from the constraints of wireless environments. Security and performance of mobile payment transactions on the non proxy-based framework therefore primarily rely on the underlying payment protocol.

Later in this chapter, we will present a mobile payment framework that solves the problems of the existing frameworks and apply the proposed framework to enable SET transactions in wireless environments.

4.2 The Proposed Framework

Recall that we have outlined existing frameworks to enable payment transactions in wireless environments in section 2.4. Also, we have pointed out advantages and disadvantages offered by each framework. However, there is no best solution among them. Both agent-based and proxy-based frameworks enhance transaction performance, but they have their own concerns about security of the systems. In the previous section, we have shown that the payment systems based on these frameworks do not satisfy the the practical and secure mobile payment system stated in the formal model. Then, in section 2.5, we have discussed the possibility to have a new framework which incorporates the main features of existing frameworks and solves their problems in order to provide a practical solution for mobile payment. In this section, the proposed framework is presented in details.

4.2.1 Details of the Proposed Framework

The proposed framework is composed of three main components:

1. **Payment Protocol:** the payment functionality of our framework can be performed by any kind of payment protocol, or even a fixed-network payment protocol. However, to achieve high transaction performance, a non proxy-based mobile payment protocol is preferred because it is designed to perform lightweight transactions.

2. **Mobile Agent:** the mobile agent in our framework performs the same task as the mobile agent in agent-based framework. That is, it performs transactions on the client's behalf. It contains payment-related information and travels to a foreign environment to generate the client's request for the particular payment protocol.

3. **Proxy Server:** to solve the problem of performing transactions in a hostile environment of agent-based framework, we set up a proxy server providing a trusted environment where the client's mobile agent can use its processing capability to perform high computational tasks. However, following the trust relationships among engaging parties in in section 3.1.10

of the formal model, only *partial* trust relationships among parties need to be established between the client and the proxy server. In particular, The issuer is trusted by the client to store the client's information in order to reduce the storage requirement on the client's device, but the information stored at the issuer must not be sufficient for the issuer to impersonate as the client and to generate the client's request without any authorization from the client.

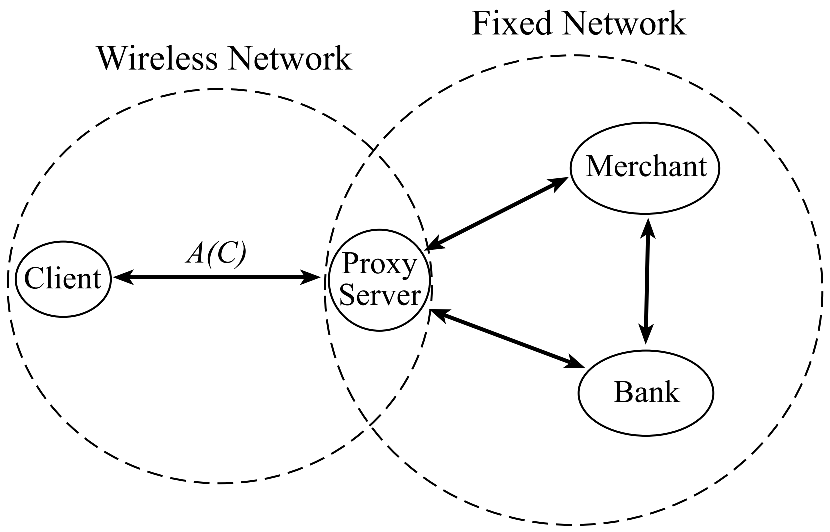


Figure 4.1: The proposed framework

The proposed framework is depicted in Figure 4.1. Note that $A(C)$ stands for the agent A owned by the client C. From Figure 4.1, the client sends the mobile agent $A(C)$, which contains the client's payment information, over a wireless network to the proxy server, where $A(C)$ generates the client's payment request, located in a fixed network. After the request has been generated, $A(C)$ travels away from the issuer to perform the payment transaction following the particular payment protocol. At the end of the transaction, $A(C)$ returns to the client with the result of her request.

4.2.2 Comparing the Proposed Framework to Existing Frameworks

In this section, the general discussion regarding the comparison between our framework and existing frameworks is presented. The detailed discussion will be given after a SET system based on our framework is introduced in the next section. In particular, in section 4.4, we will compare our SET system with the SET systems based on the existing frameworks.

The proposed framework offers several advantages over the existing frameworks. Firstly, the problem of trusted environment occurred in agent-based framework is relieved by having the proxy server. Secondly, we solve the problem of trustworthiness of the proxy server by establishing the partial-trust relationship between the client and the issuer so that not only the client can minimize storage requirement by storing some information on the proxy server, but the issuer also cannot impersonate as the client. Moreover, applying a payment protocol which is designed for a non proxy-based framework to the proposed framework results in higher performance and cost effectiveness than deploying only the protocol in the non proxy-based framework itself. That is, with the property of mobile agent, the connection cost is reduced. In addition, with the proxy server that has powerful computational capability, it takes less time to complete each transaction.

4.3 Applying the Proposed Framework to Enable Mobile SET Transactions

In this section, a SET payment system based on the proposed framework is introduced to illustrate how our framework can overcome the constraints of wireless environments and solve the problems existing mobile payment frameworks [RdS98, WLY99, WSZ01].

We first divide the payment operations in SET protocol [Mas97] into two parts; one is operated at a client's mobile device, and the other is brought with an agent travelling outside the client. At the client side, we minimize the computational load by performing low computational operations. With the agent travelling to the merchant's server, we let it perform high computational tasks.

We deploy the concept of using two mobile agents from SET/A+: *Information gathering and Negotiation Agent* (INA) which travels to merchants' sites to collect information and makes the decision on the suitable merchant, and *Payment Agent* (PA) which performs payment operations by carrying a part of the SET wallet with it.

In the proposed system, all engaging parties except the client are assumed to have powerful computing devices located in a fixed environment. Thus, we do not concern about the computational capability of these parties. The client is assumed to have a low computational capability mobile device which is able to access the Internet. The issuer has the client's credit-card information, and it is assumed to have powerful computational capability, hence we can have a proxy server maintained by the issuer as the environment where PA performs high computational operations to generate a SET's purchase request (**PReq**).

Referred to the SET protocol descriptions presented in section 2.3.1, the main task of the client in SET protocol is to generate **PReq**. Generally, **PReq** consists of order information (OI) and payment information (PI). OI is signed by the client's private key K_C^{-1}. PI is symmetric-encrypted with the session key K_2, and the key K_2 is then encrypted with the payment gateway's public key K_{PG}. Other computational tasks are operated outside the client. Thus, we divide the SET payment operations into two parts: one in the client's mobile device signing OI and the other in PA encrypting PI. The proposed framework is diagrammatically shown in Figure 4.2. The details of the proposed system are given as follows.

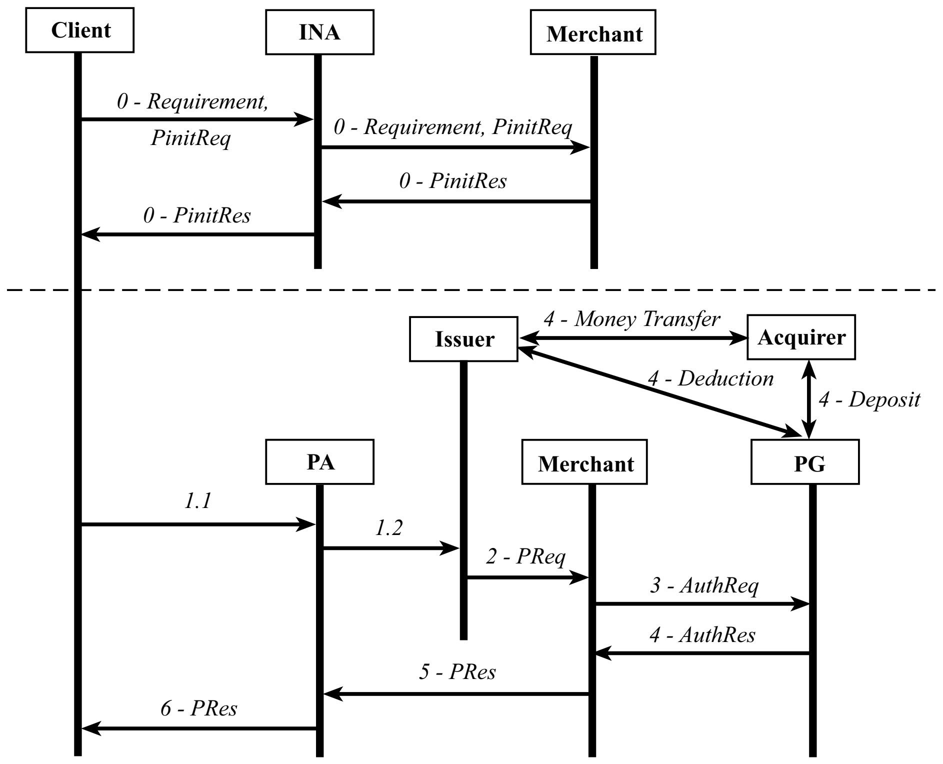


Figure 4.2: The proposed SET payment system

1. Two agents operate in the system: INA and PA. They are generated and owned by a client. The client establishes the requirements regarding the goods to INA. INA then travels to merchants' sites to search for such information. After INA returns to the client with relevant information including the merchant's identity, the client runs an authenticated key exchange (AKE) protocol for wireless networks (will be discussed in section 4.4.9) to establish a secure communication channel between herself and the issuer. After the completion of the AKE protocol, both client and issuer possess a session key K_{ex}.

   In our system, the AKE protocol is used to provide the confidentiality of the content in **PReq** because it contains both the client's public-key certificate and the price of the goods. If they are transmitted in cleartext, an attacker may be able to make the relationship between the client and the price. Then, the payment phase proceeds as follows.

2. As described above, the SET wallet operations are divided into 2 parts; one operates at the client side and the other operates at PA. After getting the key K_{ex}, the SET wallet at the client side constructs a message containing necessary information for establishing **PReq** and then encrypts this message with the key K_{ex}. This information is then passed to PA which then travels to the issuer.

   2.1) **C→PA(C)**: $\{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$
   2.2) **PA(C)→I**: $\{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$

   where,

   - $PReqGen = \{OI, \{h(OI), h(PI)\}_{K_1}, \{K_1\}_{K_C^{-1}}, ID_M, Price,$ $h(CCI), Cert_{PG}, Cert_C\}$, and
   - $PA(C)$ stands for the payment agent PA owned by the client.

3. The issuer decrypts the message with the key K_{ex} shared with the client, verifies the client's signature, and compares the received $h(CCI)$ with the one it has. If they are matched, the issuer issues the pre-image of $h(CCI)$ to PA. The SET wallet at PA then performs the following tasks:

   (i) Generate the session key K_2 (for encrypting PI).
   (ii) Reconstruct PI by collecting $\{TID, h(OD, Price), ID_M, CCI\}$.
   (iii) Decrypt $\{h(OI), h(PI)\}_{K_1}$ to retrieve $h(OI)$, and compare with the hash value of the received OI. If they are matched, the received OI is a valid OI.
   (iv) Encrypt $\{h(OI), PI\}$ with the key K_2.
   (v) Encrypt the key K_2 with K_{PG}.
   (vi) Collect relevant information to construct the SET's **PReq**.

   After constructing **PReq**, PA containing **PReq** travels to the merchant.

4. The merchant verifies the client's signature, retrieves OI, and then sends **AuthReq** to the payment gateway.

5. The payment gateway verifies the merchant's and the client's signatures and then retrieves PI. After receiving the payment information, the payment gateway consults the issuer and the acquirer for payment approval. After getting the approval, the payment gateway sends the merchant **AuthRes**.

6. The merchant verifies the payment gateway's signature and then sends **PRes** back to PA.

7. PA verifies the merchant's signature and then returns to the client with the result of her request.

4.4 Discussions

In this section, we discuss the issues regarding the application of the proposed framework to SET protocol in terms of security and performance and compare our system with the SET payment systems based on existing frameworks.

4.4.1 Issuer

In our SET payment system, the issuer maintains the tasks of the proxy server. In other words, the role of the issuer increases from that of original SET payment system in that it acts as an assistant of the client on performing transactions. This is because the issuer (such as the client's bank) has the database which contains the client's information including credit-card information (CCI). Therefore, the client's credit-card information does not have to be transmitted over the air interface as in SET/A [RdS98]. The client has to send only $h(CCI)$ to compare with the hash value of its pre-image (CCI)

stored in the issuer's database. If they are matched, the issuer issues the valid credit-card information to the payment agent PA in order to establish **PReq**.

Note that deploying SET protocol in the proposed framework does not affect the role of the issuer in our system. Normally, its tasks are the same as those of the original SET payment system. The additional tasks are only acting as the assistant of the client on issuing the credit-card information and the assistant of the payment agent PA on performing high computational operations. The message brought with PA travelling away from the issuer contains the same structure as that of the original SET's **PReq**. Moreover, the issuer cannot impersonate as the client although it has the client's CCI. To generate **PReq**, the issuer needs to have the client's private key. This is a security enhancement of our system compared to 3D SET for Mobiles [WSZ01], a proxy-based SET payment system.

It could also be noted that the issuer would cooperate in this system because, from the business point of view, this would be a payment service which results in the benefit to the issuer in that it is able to increase the number of clients as well as the number of financial transactions.

4.4.2 Trust Relationships among Engaging Parties

In SET protocol, the payment gateway acts on behalf of the issuer and the acquirer at the Internet side. It decrypts the client's payment information (PI) which contains the client's credit-card information, collects relevant information regarding the requests from both client and merchant, and then passes this information to the issuer and the acquirer. It can be seen that the payment gateway must be a party who is trusted by the client in that it will not reveal the client's credit-card information. Thus, the payment gateway must be operated by a credit-card company or a bank that issues its own credit cards. However, the payment gateway is not always a credit-card company, but possibly a company that monitors the SET payment system on behalf of the issuer.

From the above argument, it can be seen that our formal model is not only more general, but more precise than the SET payment model in regards of the trust relationships among engaging parties in the payment system. Note that the Definition 3.11 of the formal model states the trust relationship between the client and the issuer instead of the client and the payment gateway (or PSP in the model). We can see that the proposed model is more general in that the issuer can be considered as the payment gateway communicating with the client/merchant over the Internet side. It is more precise in that the payment gateway is not trusted by the client as the issuer is trusted as the party to which the client has to reveal sensitive information. However, the issuer is not fully trusted by the client in our formal model, that is, only the partial trust relationship between them is established. The issuer must not be able to impersonate as the client. In our system, the issuer cannot impersonate as the client because it does not have the client's private key, only the client's credit-card information is stored at the issuer. As a result, our system satisfies this property, whereas the proxy-based system does not.

Moreover, to make the SET protocol operate in a larger scope, the trust relationship between the client and the payment gateway needs to be reorganized in a way that the payment gateway is not trusted by the client that it will not reveal the client's sensitive information to other parties. From this argument, it can be seen that the payment system based on the original SET protocol does not satisfy the Definition 3.11 of the formal model. To solve this problem, enciphering the client's credit-card information, e.g. using hash function, is suggested.

However, in the proposed payment system, we do not modify the trust relationship between the client and the payment gateway as mentioned above. This is because we primarily concern about the practicability of the system in that existing fixed-network users must be able to perform SET payment transactions without any modification required on their SET payment software. Note that, the above trust relationship will be applied to our account-based mobile payment protocols in chapter 5.

4.4.3 Security of Mobile Agents

While deploying SET protocol in the proposed framework, we focus on the payment agent PA whose tasks are relevant to the SET payment process. In this context, PA acts as a messenger who knows its tasks and purposes. It travels to specific destinations (specific issuer and specific merchant) containing the information regarding the client's **PReq** and returns with the corresponding **PRes**. PA can be defined to have minimum security protection because there is no confidential information, such as the client's private key or credit-card information, to be protected. In addition, the security of the information carried by PA does not rely on the security of the agent itself, but on the cryptographic algorithms applied to it. The client is not necessary to be trusted by the issuer. The issuer can verify the client from the client's signature and the hash value of the client's credit-card information. Thus, the security of the agent is not a major issue in the proposed framework.

4.4.4 Transaction Security Properties

As the security of the original SET protocol has been analyzed in [Her01, KP02], in this section, we focus our analysis on the extension of the original SET protocol which is the message sent from the client to the issuer carried by the payment agent PA(C) shown as follows:

$$\mathbf{C}\rightarrow\mathbf{I}: \quad \{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$$

This message satisfies all following transaction security properties stated in the Definition 3.10 of the formal model:

- **Party Authentication** is ensured by the message with the client's signature. The client can authenticate herself to the issuer from her private key K_C^{-1}.
- **Transaction Privacy** is ensured by the message encrypted with the session key K_{ex} because only the client and the issuer can decrypt the message.
- **Transaction Integrity** is ensured by $h(PReqGen)$.
- **Non-repudiation of Transactions** is ensured by the message signed with the client's private key K_C^{-1} in that any party can prove that this message has been originated by the client because the client is the only party who has K_C^{-1}.
- **Transaction Authorization** is ensured by the message signed by the client. If the client has authorization to perform this transaction, the issuer will accept this client's request. Otherwise, the request will be rejected.

The formal analysis on transaction security properties of the proposed SET system based on our framework will be presented in chapter 7 whereby an accountability logic will be used as a tool to analyze the framework.

It can be noted that in the proposed SET system based on our framework, the client is not required to send her credit-card information over the air interface. Thus, it is not susceptible to attacks. Although K_{ex} may be compromised, the most sensitive information that the attacker can retrieve is only the price of the goods. The attacker cannot extract the goods descriptions since they are in the hashed form. Therefore, the attacker is not able to provide any linkage between the goods descriptions and their prices.

4.4.5 Client Privacy

One of the most important goals of SET protocol [Mas97] is called "*Client Privacy*". The client privacy is concerned with the secrecy of goods descriptions (OD) and payment information (PI). SET protocol achieves the client privacy property if the merchant cannot infer the client's PI, and the payment gateway (including the issuer and the acquirer) cannot infer OD at the completion of transaction.

The SET payment system based on the proposed framework also satisfies the client privacy because there is no modification at both the merchant and the payment gateway. **PReq** brought with the agent PA sent from the issuer has the same structure as that of the original SET protocol message. Furthermore, OD is not revealed to the issuer during the communications between the client and the issuer because it is hashed within OI.

4.4.6 Security of Session Keys

Wang *et al.* [WLY99] pointed out that, in SET/A [RdS98], the session key K_2 (referred to the SET protocol descriptions in section 2.3.1) is risky to be compromised because it is generated by the agent at the merchant's server which is considered as a hostile environment. Because the key K_2 is used to encrypt payment information PI, it should not be revealed to the merchant following the client privacy property described in section 4.4.5.

In the proposed system, the key K_2 is generated in the issuer's environment because the issuer is trusted by the client (referred to the Definition 3.11 of the proposed model), and the issuer's environment can be assumed to be secure against attacks because, if an attacker can access the issuer's credit-card database, the security of the entire system will be compromised.

4.4.7 Computational Load

Computational capability of engaging parties is one of the most important factors to be considered when migrating SET protocol [Mas97] to wireless environments. The computational load at engaging parties, especially at the client, must be minimized. This problem has not yet been solved in SET/A+ [WLY99]. In the proposed system, we minimize the client's computational operations and relocate high computational operations to other parties by the payment agent PA, whereas the security of the entire system is still preserved. As a result, the proposed SET system provides load optimization for the SET transactions in wireless environments. Nevertheless, the client's computational load in our SET system is slightly higher than that of the proxy-based 3D SET for Mobiles [WSZ01]. In 3D SET for Mobiles, it may be assumed that only one symmetric encryption is required to encrypt the client's request to the proxy server to generate the purchase request on behalf of the client. However, from the discussion in section 4.1.2, 3D SET for Mobiles does not satisfy the trust relationships among parties stated in the Definition 3.11 of the proposed model. Therefore, the load optimization and security of the proposed SET system is more likely to be acceptable by users than the systems based on the existing frameworks.

4.4.8 Compatibility with Existing Payment Infrastructure

On migrating a payment protocol which has been implemented in a fixed network to a wireless one, the important issue needed to be concerned is *compatibility*. That is, any mobile SET system should be compatible with existing SET payment infrastructure. To achieve this property, the modification at the merchant's and the payment gateway's infrastructures should be minimized. Both the SET payment operations on mobile devices and on fixed devices should be able to operate on the same payment infrastructure efficiently.

In the proposed SET system, although there are some modifications to the existing SET payment operations on the client's mobile device, the client's **PReq** sent to the merchant has the same structure as that of the original SET's **PReq**. Therefore, it can be implemented on the existing SET infrastructure. Moreover, the proposed system can be operated in the same scenario as that of SET/A+ [WLY99].

4.4.9 Authenticated Key-Exchange Protocols for Wireless Networks

Authenticated key-exchange (AKE) protocol is used to provide authentication and to secure communications among engaging parties. Generally, existing AKE protocols employ public-key cryptography which is not suitable for securing the communications in wireless environments due to the limitations that have been presented in chapter 1. Therefore, it is suggested that, in this thesis, all AKE protocols employed in the proposed payment protocols are the AKE protocols for wireless networks [HP98, BP98, WSZ01, WC01, LCGS03]. In this section, we present overviews of the existing AKE protocols for wireless networks.

Several AKE protocols for wireless networks [HP98, BP98, WSZ01, WC01] have been proposed including their analyses [Her01, HP98]. The protocols proposed by Horn *et al.* [HP98] and Boyd *et al.* [BMN01] employ elliptic-curve cryptosystems to reduce the computation and resource consumption of engaging parties, especially the client. The difference between these protocols is that Horn *et al.*'s protocol [HP98] is PKI-based whereas Boyd *et al.*'s protocol [BMN01] is password-based. Wong *et al.*'s protocol [WC01] is based on challenge-response between parties. Recently, Zhu *et al.* [ZWCY02] proposed a password-based AKE protocol using RSA algorithm. Horn *et al.* [HMM02] proposed an analysis of the existing AKE protocols for wireless networks including Boyd *et al.*'s protocol [BP98] and Horn *et al.*'s protocol [HP98]. They argued that both protocols are suitable for wireless communications although Boyd *et al.*'s protocol [BP98] requires more communication passes. Lam *et al.* [LCGS03] proposed a public-key based AKE protocol which reduces client's computation by applying only public-key encryptions at the client side.

4.5 Summary

In this chapter, we examined existing mobile payment frameworks and showed that the frameworks themselves do not satisfy the formal model introduced in chapter 3 and the payment systems based on the existing frameworks do not satisfy the practical and secure mobile payment system stated in our formal model. We then proposed a mobile payment framework which satisfies the formal model. Our framework not only incorporates the main features of agent-based and proxy-based frameworks in terms of transaction performance, but also solves their security problems.

We have demonstrated the usability of the proposed framework by applying it to SET protocol [Mas97] which has been deployed in the existing frameworks. The results has shown that the SET payment system based on our framework is able to solve the problems of the SET payment systems based on the existing frameworks as described below:

- To deal with the problem of high computational load at the client in SET/A+ [WLY99], our SET payment operations are divided into two parts; one resides at the client's mobile device performing low computational tasks and the other is brought with a payment agent performing high computational tasks at the issuer who acts as the client's assistant.

- We solved the problem of SET/A [RdS98] regarding the session key generation mentioned in [WLY99] with the assistance of the issuer.
- The concern about the trustworthiness of the proxy server previously presented in section 2.4.1 has been relieved by establishing partial-trust relationship between the client and the issuer.
- The client's credit-card information is not required to be transmitted over the air interface. This results in security enhancement of the system.

In addition, we have shown that the proposed SET system satisfies not only all transaction security properties ($TSec$) stated in the Definition 3.10 of the formal model, but also *client privacy* property which is a major goal of the SET transaction. Furthermore, the proposed system has no modification at both the merchant and the payment gateway. It results in full compatibility with the existing SET payment infrastructure.

Note that, comparing to the proposed model presented in chapter 3, the proposed SET system does not satisfy trust relationships among engaging parties ($Trust$) stated in the Definition 3.1.10 because the SET protocol itself does not satisfy this property. We did not modify the SET protocol descriptions since we aim to preserve the compatibility with the existing SET infrastructure. However, the suggestion on modifying the SET protocol to achieve $Trust$ has been provided in section 4.4.2.

In terms of transaction performance, it can be seen that, with the proposed framework which incorporates the features of agent-based and proxy-based frameworks, high computational SET protocol has been successfully operated in the wireless environment. From the discussion in section 4.4.7, it can be seen that the system based on the proposed framework is more likely to satisfy acceptable transaction performance (ATP) than the systems based on the existing frameworks.

To satisfy practical and secure mobile payment system stated in the Definition 3.15 of the proposed model, our system must satisfy $Goals$, PR, $TSec$, $Trust$, and ATP. From the discussion in this chapter, we have shown that $Trust$, ATP, and $TSec$ are satisfied. The analysis of the other properties will be presented in chapter 7.

It could be noted that applying a payment protocol designed for a non proxy-based framework that is more lightweight than SET protocol to our framework therefore results in higher transaction performance. In chapter 5, two lightweight, non proxy-based mobile payment protocols will be proposed. Deploying these protocols in our framework will result in high transaction performance and security that is likely to be acceptable by mobile users.

Chapter 5

Securing Account-Based Mobile Payment

In chapter 4, a new framework for practical and secure mobile payment has been introduced. The framework allows any kind of payment protocols to operate in it with secure and better transaction performance compared to existing mobile payment frameworks. Applying a fixed-network payment protocol to the framework offers better transaction performance than performing it alone in a wireless environment. Moreover, deploying a lightweight, non proxy-based mobile payment protocol in the proposed framework greatly enhances the transaction performance of the payment system.

In this chapter, we propose a family of non-proxy, account-based payment protocols, namely Kungpisdan-Srinivasan-Le (KSL) protocol version 1 (called KSLv1) and KSL protocol version 2 (called KSLv2), which work efficiently in wireless environments. These protocols mainly apply symmetric-key operations and keyed-hash functions at the client's mobile device. In KSLv1, only the client is not required to perform any public-key operations, whereas in KSLv2, none of the parties is required to have public-key certificate.

This chapter is organized as follows. Section 5.1 outlines existing secure symmetric-key cryptographic techniques. Section 5.2 presents the cryptographic technique that works behind the proposed protocols. Section 5.3 introduces notations for the proposed protocols. Sections 5.4 and 5.5 present KSLv1 and KSLv2 protocols in details. In section 5.6, the security and performance of the proposed protocols are discussed. Section 5.7 shows how the proposed protocols can be applied to the proposed framework previously introduced in chapter 4. Section 5.8 summarizes the chapter.

5.1 Enhancing Security of Symmetric Cryptography

In this section, we review existing techniques to enhance security of symmetric cryptography that is the background of the cryptographic technique which works behind KSLv1 and KSLv2 protocols.

In communication scenario, the concept of secret chaining [MB01, Cim02] has been widely implemented. Generally, the secret chaining is the method to generate a number of secrets from a master secret, and each of them is used as a session secret for authentication purpose during a transaction. An obvious example is one-time password proposed by Lamport [Lam81].

The purpose of secret chaining is to solve the problem of transferring a static, long-term shared secret over an insecure network. This can be done by sending a session secret derived from the master secret instead. A good secret chaining scheme should provide forward secrecy property in that it is difficult for an attacker to derive the next session secret from the current session secret.

Several secret chaining techniques have been proposed [MB01, Cim02]. In Marvel *et al.*'s approach [MB01], a chain of secrets was applied to MAC (Message Authentication Code) previously proposed by Tsudik [Tsu92]. The message format in this approach is shown as follows:

$$MAC(Message, K_i), Message$$

where $Message$ stands for a message to be sent to its intended recipient and K_i is a member in the set of secrets which cannot be generated by the recipient of this message. It can be seen that this message can be generated only by the party who knows the secret K_i.

In order to provide message privacy and to identify the originator of the message, the secret chaining technique was applied to Kerberos system based on symmetric-key operations [Cim02]. The message format in this approach is given as follows:

$$\{i, K_i, Message\}_{K_{AB}}$$

where K_{AB} is a key shared between a party A and a party B generated by a Key Distribution Center (KDC), i is an index of K_i, and K_i cannot be generated by the recipient of the message. It can be seen that the privacy of the above message can be achieved by the encryption with the key K_{AB} in that only the parties who have K_{AB} can decrypt the message. Furthermore, we can identify the originator of the message from K_{AB} and K_i in that we can ensure that the party who has both keys has originated this message.

In the next section, we present a symmetric cryptographic technique which enhances transaction security properties of symmetric cryptography (referred to section 3.1.9 of the proposed formal mobile payment model).

5.2 The Proposed Cryptographic Technique

To overcome the limitations of applying asymmetric cryptography to wireless environments stated in chapter 1, we aim to enhance transaction security properties of symmetric cryptography in order to provide the same security properties as that provided by asymmetric cryptography stated in section 3.1.9 of the proposed model. It can be noted that, as mentioned in section 3.1.11, replacing public-key operations with lightweight cryptographic operations that are able to provide the same transaction security properties results in higher transaction performance.

We realize that performing asymmetric cryptographic operations on mobile devices is not applicable [WC01, RdS98, KSL03a]. However, symmetric cryptography still has disadvantages compared to asymmetric one. As described in [Kai96], one of the main advantages of asymmetric cryptography over symmetric one is the ability to identify the originator of a message from its digital signature, whereas this property is not achieved in symmetric cryptography because the encrypting key is shared among the parties who have the shared key. We cannot identify the originator of a symmetric-encrypted message because any party who has the shared key can generate the message.

We apply the techniques proposed by Cimato [Cim02] and Marvel *et al.* [MB01] presented in section 5.1 to mobile payment scenario by deploying symmetric-key operations to secure the communications among engaging parties. The proposed concept is that a party can authenticate herself to other parties by attaching a secret which cannot be generated by those parties to the message. The intended recipient of the message can identify the originator of the message from the attached secret. The proposed technique can be formalized as follows:

$$\langle M_1, \langle M_2 \rangle_Y \rangle_X$$

where $\langle M \rangle_K$ stands for the message M applied with a single-key cryptographic operation with the key K. The above message can be presented in various formats. For example,

- $\{M_1, \{M_2\}_Y\}_X$,
- $\{M_1, h(M_2, Y)\}_X$,
- $h(M_1, h(M_2, Y), X)$, or
- $h(M_1, h(Y), X)$

where $h(M, X)$ can be a hash function whereby its input is the message M concatenated with the key K, a keyed-hash function that applies the key X to the message M, or MAC (Message Authentication Code) of the message M with the key K.

The following example describes how the proposed technique works and its properties. Suppose that X is a secret shared between a client and a merchant. An issuer has given a secret Y, where $Y \neq X$, to the client. From the above message, it can be seen that, if the above secrets (X and Y) are not compromised, the above message has been originated by the client because she holds all secrets (X and Y), whereas the merchant does not have Y. Moreover, the above message cannot be generated by the issuer because it does not have X. When a dispute occurs, with the assistance of the issuer, the client is able to prove to a verifier that she is the originator of this message. In addition, the above message can be considered as non-repudiable evidence in that the client cannot deny that she is the originator of the message. Note that this technique requires no conspiracy between the issuer and the merchant. However, in the payment scenario, considering Y as the client's credit-card information, the issuer would never reveal the secret Y to the merchant.

Referred to the transaction security properties stated in section 3.1.9, the above message satisfies all transaction security properties except transaction integrity. To provide the transaction integrity, the message can be modified by the following:

$$\langle M_1, \langle M_2 \rangle_Y, h(M_1, X_2) \rangle_{X_1}$$

or

$$\langle M_1, \langle M_2 \rangle_Y \rangle_{X_1}, h(M_1, X_2)$$

where X_1 and X_2 are shared between the client and the merchant. We can see that the hash functions applied to each of the above messages guarantee the integrity of the message M_1. Without knowing X_2, the message cannot be modified without being detected by the recipient of the message.

It can be noted that the ability to identify the sender of a message leads to satisfying non-repudiation property. As presented in section 3.1.9, non-repudiation is the property that a party cannot deny the transaction she has performed. Moreover, as stated in [KP02], the message satisfying this property offers the ability to resolve disputes among parties.

We can see that, as described previously, the above symmetric-key based message formats satisfy transaction security properties stated in the Definition 3.10 of the proposed mobile payment model. Such properties can normally be achieved by deploying public-key operations. Later in this chapter, in sections 5.4 and 5.5, we demonstrate the usability of the proposed cryptographic technique by presenting two non proxy-based mobile payment protocols based on the proposed technique. The discussions regarding security and performance of the protocols will be given in section 5.6.

5.3 Notations

The following notations are used throughout this chapter:

- $\{C, M, PG, I, A\}$: the set of client, merchant, payment gateway, issuer, and acquirer, respectively.
- OI: order information. $OI = \{TID, h(OD, Price)\}$.
- PI: payment information which contains credit-card information (CCI).
- OD: order descriptions which contains goods descriptions.
- $Price$: amount and currency.
- TID: identity of transaction which contains time and date of transaction $Date$.
- Yes/No: status of transaction *approved/rejected*.
- K: a session key.
- n: a nonce for challenge-response.
- $TIDReq$: the request for TID.
- $MIDReq$: the request for ID_M.

5.4 KSLv1 Protocol

In this section, we propose KSLv1, a simple and powerful credit-card payment protocol which is suitable for wireless environments. KSLv1 deploys the cryptographic technique presented in section 5.2. Section 5.4.1 presents initial assumptions for KSLv1. Section 5.4.2 presents the session key generation technique for KSLv1 protocol. In section 5.4.3, KSLv1 is presented in details.

5.4.1 Initial Assumptions

The basic initial assumptions of the proposed KSLv1 protocol are given as follows:

1. There are five engaging parties in the protocol: client, merchant, payment gateway, issuer, and acquirer. Referred to the proposed model in chapter 3, the payment gateway performs the task of payment system provider (PSP).

2. The client accesses the Internet by using a mobile device.

3. All engaging parties *except* the client are required to have their own public-key certificates.

4. The issuer issues a shared secret Y to the client. Both the client and the issuer then generate a set of secrets Y_i, where $i = 1, ..., n$, by using the agreed session key generation technique (the details of the key generation technique will be presented in section 5.4.2) and store it on their devices. The secret Y can be distributed by using an authenticated key-exchange (AKE) protocol for wireless networks. The details of existing AKE protocols for wireless networks have been discussed in section 4.4.9.

5. The client's credit-card information (CCI) is a secret known only to the client and her issuer. CCI is different from the semi-secret credit-card number presented on the card. CCI is transferred from the issuer to the client (over a secure channel) once the client first registers for the mobile credit-card payment service to the issuer.

6. The issuer is trusted by the client not to reveal Y and CCI to other parties.

7. The client is not required to trust a payment gateway.

8. The client and the merchant have agreed on the price and goods descriptions.

9. It is easy to compute $h(x)$ from the given x, and it is computationally infeasible to compute x from $h(x)$. Also, the MAC algorithm is assumed to be fast and secure.

5.4.2 Key Generation Techniques

In KSLv1 protocol, two sets of shared keys are generated: X_i (shared between the client and the merchant) and Y_i (shared between the client and her issuer), where $i = 1, ..., n$. To generate the sets of session keys, there are several techniques that have been proposed to secure shared secrets presented in section 2.6.

In this section, we deploy two efficient key generation techniques that have been presented in [KSL03a]; one is used for generating the set of X_i from the given secret X and the other is used for generating the set of Y_i from the given secret Y. The main concept of both techniques is to apply one-way hash function with one-bit cyclic shift (either left shift or right shift) of a master secret at each time when generating a session key. The details of both techniques are given as follows:

Generating X_i

X_1 = *h(1-bit-shift-of-X)*, X_2 = *h(2-bit-shift-of-X),...,* X_n = *h(n-bit-shift-of-X)*

Generating Y_i

Y_1 = *h(1-bit-shift-of-(CCI, Y))*, Y_2 = *h(2-bit-shift-of-(CCI, Y)),...,*
Y_n = *h(n-bit-shift-of-(CCI, Y))*

The technique presented above gives the idea about how the session keys can be generated. Note that it is not restricted to the above techniques to generate the session keys. They are presented as examples of possible key generation techniques. The discussion of the above techniques will be given later in this chapter.

Before performing a transaction, the client needs to register herself to the issuer and the merchant. The registration can be done either by phone or via the issuer's website. After the registration is successful, the client receives client's wallet software (called *KSL Wallet*) by mail or by downloading from the issuer's site. The KSL wallet contains both key generation and payment software. After the KSL wallet is successfully installed, the set of Y_i is generated and stored on the client's mobile device. To generate Y_i, the issuer sends the client the secret Y securely by using an AKE protocol for wireless networks (that has been previously discussed in section 4.4.9). The client then uses Y, together with her own CCI, to generate a set of Y_i using the agreed key generation technique. To generate the set of X_i, the client needs to run *Merchant Registration Protocol* to register herself to the merchant in order to share the secret X with the merchant. The details of KSLv1 protocol will be given in the next section.

5.4.3 Details of KSLv1 Protocol

The design of KSLv1 is based on the proposed formal model presented in chapter 3. KSLv1 is composed of two sub-protocols: *Merchant Registration Protocol* and *Payment Protocol*. In *Merchant Registration Protocol*, the client shares the secret X with the merchant when she newly registers herself to the merchant or she wants to update the secret X. After X has been distributed, the set of session keys X_i is then generated and stored on each party's device.

After the sets of X_i and Y_i are successfully generated, the client can start the *Payment Protocol*. The sets of X_i and Y_i will be used as session keys during transactions.

Merchant Registration Protocol

Before performing a transaction, the client registers herself to the merchant in order to share the secret X with the merchant. The details of the *Merchant Registration Protocol* are shown as follows:

C$\rightarrow$**M**: $\{ID_C, X, n\}_K$

M$\rightarrow$**C**: $\{n\}_K$

First of all, the client generates the secret X which is to be shared with the merchant. The client then sends the merchant her identity ID_C, a nonce n, and the secret X, encrypted with a session key K generated from running an AKE protocol for wireless networks (its details have been previously discussed in section 4.4.9) with the merchant. The merchant then confirms the client's registration by sending the nonce n encrypted with the key K to the client. After the completion of the protocol, Each of them generates a new set of X_i by using the agreed key generation technique (which has been presented in section 5.4.2) and stores it on each party's device.

The objective of this protocol is to distribute and update the secret X between the client and the merchant. Thus, it is not required for every transaction, but only when either the client or the merchant wants to update X.

Payment Protocol

After generating all necessary secrets, the client is able to perform a payment transaction whose steps are:

Step 1 **C→M**: $ID_C, i, TIDReq, MIDReq$

M→C: $\{TID, ID_M\}_{X_i}$

Step 2 **C→M**: $\{OI, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i}$, $MAC[(OI, Price, ID_C, ID_I), X_{i+1}]$

Step 3 **M→PG**: $\{\{MAC[(Price, h(OI), ID_M), Y_i], Price\}_{K_{PG}}, h(OI), i, TID, ID_C, ID_I\}_{K_M^{-1}}$

Step 4 Under private network,

4.1) **PG→I**: $MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price, ID_C, ID_M$

4.2) **PG→A**: $Price, ID_M$

4.3) **I,A→PG**: $Yes/No, \{h(OI), Yes/No\}_{Y_i}$

Step 5 **PG→M**: $\{\{h(OI), Yes/No\}_{Y_i}, \{h(OI), Yes/No\}_{K_M}\}_{K_{PG}^{-1}}$

Step 6 **M→C**: $\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$

Step 1: the client and the merchant exchange necessary information to start the protocol. Note that the index i is used to identify the current session key X_i.

Step 2: the client sends a *Payment Ordering* request (referred to the Definition 3.6 of the formal model to the merchant. The *Payment Ordering* request contains order information (OI) which informs the merchant about the goods and price requested. It also contains $MAC[(Price, h(OI), ID_M), Y_i]$ which represents a *Debit* request that is to be forwarded to the issuer.

It can be noted that, although the merchant has X_i, she cannot generate this message since she does not have Y_i used for constructing $MAC[(Price, h(OI), ID_M), Y_i]$. Consequently, the message has been proven that it has really been originated by the client.

Step 3: the merchant decrypts the message to retrieve OI. As OI contains $h(OD, Price)$, the merchant can compare $h(OD, Price)$ with the hash value of OD and $Price$ that she has got. The merchant then sends a *Credit* request signed with her private key to the payment gateway. The *Credit* request also contains the forwarded *Debit* request.

We can see that the *Debit* request together with $Price$ are encrypted with the payment gateway's public key K_{PG}. This is to ensure that only the payment gateway is the intended recipient of the message. Note that ID_C and ID_I are used to identify the client and the issuer, respectively. The index i is used to identify the current session key Y_i. In addition, the payment gateway cannot generate the *Credit* request by itself since it does not have Y_i.

Step 4: after receiving the message and verifying the merchant's signature, the payment gateway recognizes the issuer from ID_I. Then, it passes the *Debit* request together with relevant information, including the index i, to the issuer. The payment gateway can recognize the merchant from the merchant's digital signature. The payment gateway then sends ID_M and the requested price ($Price$) to inform the acquirer that the merchant is the party to which the requested amount paid by the client will be transferred. After checking the validity of the client's account, either the issuer or the acquirer sends an approval result (Yes/No) to the payment gateway. Note that this step is performed under a banking private network, hence we do not concern about its security issue.

Step 5: after receiving the approval result, the payment gateway sends a *Credit* response to the merchant and a *Debit* response to the client through the merchant. Note that $\{h(OI), Yes/No\}_{K_M}$ represents the *Credit* response, and $\{h(OI), Yes/No\}_{Y_i}$ represents the *Debit* response. The merchant verifies the payment gateway's digital signature and retrieves the response as a result of her *Credit* request. The merchant can check whether the message is really the response of her request by comparing the received $h(OI)$ with the hash value of her own OI. If they are not matched, the merchant rejects the transaction. The merchant then encrypts the *Debit* response with X_{i+1}, and forwards the encrypted message to the client as a *Payment Ordering* response.

Step 6: the client retrieves the result of her request and compares the received $h(OI)$ with the hash value of her own OI. As well as that in **Step 5**, if they are not matched, the client rejects the transaction.

In next transactions, the client does not have to run the *Merchant Registration Protocol* with this merchant again. She can use other values in the set of X_i to perform transactions until being notified to update the secret X.

It can be seen that the *Payment Protocol* is the main sub-protocol of KSLv1 protocol which needs to be performed in every transaction whereas the *Merchant Registration Protocol* is used only to update the secret X. In addition, the issuer can update the secret Y and distribute it to the client by using any AKE protocol for wireless networks. Also, after each X_i and Y_i has been used, they are put into all parties' revocation lists in order to prevent the replay of the secrets from both client and merchant.

5.5 KSLv2 Protocol

In the previous section, we have presented KSLv1, a credit-card mobile payment protocol which reduces the computation at the client who is assumed to perform payment transactions using a low computational capability mobile device. In KSLv1, the client's computation is reduced by deploying symmetric-key operations, whereas the merchant and the payment gateway perform public-key operations as they are assumed to have powerful computing capability devices.

It can be seen that, in KSLv1, the merchant and the payment gateway are restricted to perform transactions using fixed-network devices. However, in some applications, these parties can be mobile users. For example, a taxi driver who provides mobile payment service to passengers. She is required to have a mobile device installed in the taxi performing transactions as a merchant in the payment system.

Moreover, a mobile payment system can be applied to money transfer among mobile users. For example, Alice requests Carol to pay money to Bob on behalf of Alice. In this case, Carol may be Alice's personal agent or an authorized person who performs a transaction on Alice's behalf as a result of the request from Alice. After receiving Alice's request, Carol sends a message as a commitment to notify Bob that she will pay Bob the money on behalf of Alice. The actual payment may be later performed over a fixed network by Carol to Bob. In this case, Carol acts as a payment gateway in the mobile payment system where no actual money is transferred, but only the commitments regarding fund transfer during the transaction are transferred among them.

In this section, we present KSLv2, a credit-card payment protocol for wireless networks that is able to serve the above requirements. KSLv2 satisfies all transaction security properties stated in the Definition 3.10 of the proposed model without any public-key operations required at all engaging parties. Thus, all parties' computation and communication passes are reduced. Moreover, without expensive PKI-based operations, the setup cost for payment infrastructure and the transaction cost are reduced. Therefore, all engaging parties including the merchant and the payment gateway are able to perform transactions on mobile devices.

In section 5.5.1, initial assumptions of the proposed KSLv2 protocol are given. Section 5.5.2 presents the key generation technique for KSLv2. In section 5.5.3, KSLv2 is presented in details.

5.5.1 Initial Assumptions

The initial assumptions of KSLv2 protocol are given as follows:

1. As well as KSLv1, there are five engaging parties in KSLv2: client, merchant, payment gateway, issuer, and acquirer.

2. The client shares her credit-card information (CCI) with the issuer.

3. Once the merchant registers herself to the payment gateway, the payment gateway shares the secret Z with the merchant. Both of them then generate a set of secrets Z_j, where $j = 1, ..., n$, (the details of session key generation technique will be presented in section 5.5.2) and store it on their devices. Also, the issuer shares the secret Y with the client. Both of them then generate a set of secrets Y_i, where $i = 1, ..., n$, and store it on their devices. The secrets Y and Z can be distributed by using an AKE protocol for wireless networks. The details of existing AKE protocols have been presented in section 4.4.9.

4. The issuer is trusted by the client in that it will not reveal the client's CCI and Y to other parties.

5. It is easy to compute $h(x)$ from the given x, and it is computationally infeasible to compute x which $y = h(x)$ from the given y. Moreover, the MAC algorithm is a fast and secure version.

5.5.2 Key Generation Technique

The key generation techniques for X_i and Y_i, where $i = 1, ..., n$, have been presented in section 5.4.2. Based on [KSL04c], to generate a set of Z_j, where $j = 1, ..., n$, we deploy the key generation technique for X_i as follows:

$$Z_1 = h(\textit{1-bit-shift-of-Z}),\ Z_2 = h(\textit{2-bit-shift-of-Z}),...,\ Z_n = h(\textit{n-bit-shift-of-Z})$$

where Z_j is j-bit cyclic shift of the secret Z.

5.5.3 Details of KSLv2 Protocol

In this section, KSLv2 protocol is presented in details. As well as KSLv1, before performing a transaction, the client and the merchant need to run the *Merchant Registration Protocol* to distribute the secret X. As the details of *Merchant Registration Protocol* has been previously presented in section 5.4.3, in this section, we demonstrate only the *Payment Protocol* of KSLv2.

Payment Protocol

Step 1 **C→M**: $ID_C, i, TIDReq, MIDReq$

M→C: $\{TID, ID_M\}_{X_i}$

Step2 **C→M**: $\{OI, h(Y_i), ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i}$, $MAC[(OI, Price, ID_C, ID_I), X_{i+1}]$

Step 3 **M→PG**: $\{MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price, ID_C, ID_I\}_{Z_j}, ID_M, j, MAC[(h(OI), TID, ID_C, ID_I), Z_{j+1}]$

Step 4 Under private network,

4.1) **PG→I**: $MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price, ID_C, ID_M, h(Z_{j+1})$

4.2) **PG→A**: $Price, ID_M$

4.3) **I,A→PG**: $Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}, h(Yes/No, h(OI), h(Y_i))$

Step 5 **PG→M**: $\{Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}, h(Yes/No, h(OI), h(Y_i))\}_{Z_{j+1}}$

Step 6 **M→C**: $\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Yi}\}_{X_{i+1}}$

Step 1: the client and the merchant exchange the information necessary to start the protocol.

Step 2: the client sends the merchant a *Payment Ordering* request (referred to the Definition 3.6 of the proposed model in chapter 3). The *Payment*

Ordering request contains OI used to inform the merchant about the goods and price requested. It also contains $MAC[(Price, h(OI), ID_M), Y_i]$ which is a *Debit* request that is to be forwarded to the issuer. It can be noted that, although the merchant has X_i, she cannot generate this message since she does not have Y_i used to construct $MAC[(Price, h(OI), ID_M), Y_i]$. Thus, we can ensure that this message has really been generated by the client. It can be noted that the content of the message in this step is the same as that of **Step 2** of KSLv1 except the additional $h(Y_i)$. The purpose of $h(Y_i)$ is to prevent the payment gateway from modifying the message in **Step 5** sent from the payment gateway to the merchant. More details will be discussed in **Step 5**.

Step 3: the merchant decrypts the message to retrieve OI. The merchant then sends the payment gateway a *Credit* request encrypted with Z_j. The *Credit* request contains the forwarded *Debit* request. Note that ID_C is used to identify the client, and the indexes i and j are used to identify the current session keys Y_i and Z_j, respectively.

Step 4: after receiving the message, the payment gateway can identify the issuer from ID_I. The payment gateway then passes the *Debit* request to the issuer. Also, the payment gateway sends ID_M and *Price* to notify the acquirer that the merchant is the party to which the requested amount *Price* will be transferred. The issuer checks the validity of the client's account and sends the approval result (Yes/No) to the payment gateway. Note that this step is performed under the banking private network. Thus, we do not need to concern about its security issue.

Step 5: the payment gateway sends a *Credit* response as a result of the *Credit* request to the merchant. Note that the entire message encrypted with Z_{j+1} represents the *Credit* response, whereas $\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}$ represents a *Debit* response which will be later forwarded to the client. The merchant can check whether this message is the response of her request by comparing the received $h(OI)$ with the hash value of her own OI. If they are not matched, the merchant rejects the transaction. Note that the payment gateway cannot fool the merchant by modifying the approval result because it does not have $h(Y_i)$. The merchant can verify *Yes/No* from $h(Yes/No, h(OI), h(Y_i))$ by using OI and $h(Y_i)$ previously received from **Step 2**.

Step 6: the merchant encrypts $\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}$, which represents the *Debit* response, with X_{i+1} and sends the encrypted message to the client as *Payment Ordering* response. The client decrypts the message to retrieve the results of her requests from both responses.

5.6 Analysis and Discussions

5.6.1 Goals of Engaging Parties for Payment Transactions

In this section, we show that the proposed protocols, KSLv1 and KSLv2, satisfy the goals of engaging parties for payment transactions stated in the Definition 3.7 of the proposed mobile payment model. Considering KSLv1, it can be seen that:

- **Client's Goal** is satisfied by **Step 6** of the *Payment Protocol* sent from the merchant to the client as a receipt of payment. It is encrypted with the secret X_{i+1} which can be used to authenticate the merchant. It also contains the result of transaction Yes/No. Moreover, $h(OI)$ contains goods descriptions so that the client can check that she has paid for the correct goods or services.

- **Merchant's Goal** is satisfied by **Step 5** of the *Payment Protocol* in that the payment gateway is identified by the message signed with its private key K_{PG}^{-1}, Yes/No notifies the merchant about the result of transaction, and $h(OI)$ contains the approved price.

- **Payment System Provider's Goal** is satisfied by **Step 5** and **Step 6** of the *Payment Protocol* in that, after the merchant receives the message in **Step 5** and the client receives the message in **Step 6**, it infers that the payment gateway has completed its tasks.

According to KSLv2 protocol, all goals of engaging parties are also satisfied in that:

- **Client's Goal** is satisfied by **Step 6** of the *Payment Protocol.*

- **Merchant's Goal** is satisfied by **Step 5** of the *Payment Protocol* in that the message is encrypted with Z_{j+1} known only by the merchant and the payment gateway, but the merchant is not capable of generating $\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}$, but the payment gateway is. This is because the payment gateway has Z_{j+1}. Therefore, it is clear that the payment gateway (on behalf of the issuer and the acquirer) has originated this message. Moreover, the message contains approval result (Yes/No and $h(OI)$) which contains the approved amount to be transferred to the merchant's account.

- **Payment System Provider's Goal** is satisfied by **Step 5** and **Step 6** of the *Payment Protocol* with the same reason as that of proving the PSP's goal in KSLv1.

As a result, the proposed KSLv1 and KSLv2 protocols satisfy all of the goals of engaging parties regarding payment transactions. As mentioned in section 3.1.7 that the goals of engaging parties ($Goals$) can be formalized by accountability, in chapter 7, the goals of engaging parties of KSLv1 and KSLv2 will be formally analyzed by using an accountability logic to emphasize the results of the above discussion.

5.6.2 Transaction Security Properties

The most concerned security aspect of any payment system is transaction security, in that, each transaction must be performed in a secure manner. In this section, we consider whether or not the proposed KSLv1 and KSLv2 protocols satisfy transaction security properties ($TSec$) stated in the Definition 3.10 of the proposed mobile payment model. From the message format discussed in section 5.2,

$$\{Message, Y, MAC[(Message, Y), X_2]\}_{X_1}$$

After the above message format has been applied to KSLv1 and KSLv2 in sections 5.4 and 5.5, all transaction security properties stated in the formal model are satisfied in that:

- **Party Authentication** is satisfied by the symmetric encryption with the secret X_1 and also satisfied by the secret Y shared between the client and the issuer. The encryption ensures that either the client or the merchant has generated this message, and the secret Y ensures that the client is the originator of the message.
- **Transaction Privacy** is satisfied by the symmetric encryption with the secret X_1 which guarantees that this message is known only by the parties who have X_1 that are the client and the merchant.
- **Transaction Integrity** is satisfied by MAC. With MAC, although the key X_1 is compromised, the attacker still cannot modified the content of the message as it is in the MAC form with the key X_2, where $X_2 \neq X_1$.
- **Transaction Authorization** is satisfied by the encryption with the secret X_1 in that only the party who has X_1 is recognized by the merchant and is allowed to perform the transaction with the merchant.

- **Non-repudiation of Transactions** is satisfied by the secret Y in that the merchant is able to provide non-repudiable evidence to prove that the client has originated this message and has requested to perform the transaction with the merchant. This is because Y cannot be generated by the merchant, but by either the client or the issuer, and the merchant can ensure that the client has originated the message and she is the intended recipient of this message because the message is encrypted with X_1 which is shared between the client and herself.

Note that we will also formally analyze KSLv1 and KSLv2 on accountability in chapter 7 to emphasize that the proposed protocols satisfy transaction security properties stated in the proposed formal model.

5.6.3 Failure Recovery

Compared to fixed environments, network failures occur more frequently in wireless environments due to less network reliability. Moreover, as wireless communication technologies offer mobility to users, in some situation, a user may be out of network coverage while performing a transaction.

We realize that failure recovery is relevant to key synchronization process, in that, after a payment system is recovered from a failure, it should have an efficient key synchronization mechanism. Our protocols are able to deal with the situation when the failure occurs, especially while performing several sessions concurrently. For example, five sessions are performing concurrently. The session keys X_1-X_5 and Y_1-Y_5 (and Z_1-Z_5 in case of KSLv2 protocol) are being used. If the session 3 fails, after its recovery, in KSLv1 protocol, the client can restart the session 3 with the keys X_c and Y_c, where c can be any value in the sets of X_i and Y_i, not restricted to the their next values. The issuer can recognize the new session keys X_c and Y_c from the index c attached to the message sent from the client. As each session key is generated independently, the client can select any value in the set of X_i.

If the same situation occurs in KSLv2 which deploys three sets of session keys X_i, Y_i, and Z_j, the key synchronization process of X_i and Y_i can be done by the same process as that of KSLv1 protocol. According to Z_j, the merchant can restart the session with the key Z_d, where d can be any value in the sets of Z_j. The payment gateway can recognize the new session key Z_d from the index d attached to the message sent from the merchant. As well as X_c and Y_c, Z_d is independently generated. Therefore, the merchant can select any value in the set of Z_j to start a new payment session.

5.6.4 Trust Relationships among Engaging Parties

In any payment system, a party should not trust others unless a valid proof of trustworthiness can be provided. However, in SET [Mas97] and iKP [BGH$^+$00] protocols, the client and the merchant need to trust the payment gateway since some confidential information, particularly the client's credit-card information, has to be revealed to the payment gateway in order to be forwarded to the issuer/acquirer for clearing purpose. Unfortunately, the payment gateway may be a company that is monitoring the system. It may possibly have a conspiracy with an attacker, or even the merchant, so that the attacker can get the client's credit-card information without any attempt to decrypt messages.

The trust relationships among parties in KSLv1 and KSLv2 protocols are based on the Definition 3.11 of the proposed model, in that, we state the trust relationship between the client and the issuer instead of the trust relationship between the client and the payment gateway because the issuer issues a credit card to the client. Therefore, we do not need to concern about the honesty of the payment gateway as that in SET and iKP protocols.

5.6.5 Performance Analysis

In this section, we show that the proposed KSLv1 and KSLv2 protocols have advantages over existing account-based payment protocols in terms of transaction performance, especially when applied to wireless environments. We mainly focus on the computation at engaging parties, particularly the number of cryptographic operations applied to each engaging party in each protocol. Table 5.1 and 5.2 demonstrate the number of cryptographic operations applied to KSLv1, KSLv2, SET [Mas97], 3KP [BGH$^+$00], and Antovski *et al.*'s protocols [AG03] at client, merchant, and payment gateway, respectively. Note that H stands for hash function, KH stands for keyed-hash function (or MAC), and p and q are large prime numbers, the RSA public-key parameters. As the payment gateway performs the same tasks as that of FSP in Antovski *et al.*'s protocol (referred to the protocol descriptions presented in section 2.4.1), we refer FSP as the payment gateway in our comparison. Note also that 3KP protocol is used in our comparison because it provides the same security level as provided by SET protocol whereas 1KP and 2KP cannot achieve that security level. Moreover, in our comparison, all the above protocols are assumed to deploy the same kinds of cryptographic algorithms.

From Table 5.1 and 5.2, it is not hard to see that, in KSLv1 and KSLv2, no public-key operations is required at the client side. Only symmetric-key operations including MAC and hash functions are required at the client. Compared to SET, iKP, and Antovski *et al.*'s protocols, the client is required to perform both signature generations and public-key decryptions which are considered as high computational tasks. In particular, all engaging parties in KSLv2 are not required to perform any public-key operations. Note that in the proposed KSLv1 and KSLv2, the key generation process is required to update secret keys regularly. In Table 5.1, we assume that each session key (X_i, Y_i and Z_j) is generated at the beginning of each transaction. In fact, this process can be done

Cryptographic Operations		KSLv1	KSLv2
1. Public-key encryption	C	-	-
	M	1	-
	PG	1	-
2. Public-key decryption	C	-	-
	M	1	-
	PG	1	-
3. Digital signature	C	-	-
	M	1	-
	PG	1	-
4. Signature verification	C	-	-
	M	1	-
	PG	1	-
5. Symmetric operation	C	4	4
	M	3	5
	PG	-	2
6. Hash function (H)	C	3	2
	M	3	-
	PG	-	-
7. Keyed-hash function (KH)	C	2	2
	M	-	2
	PG	-	1
8. Session key generations	C	$3KH$	$3KH$
	M	$2KH$	$4KH$
	PG	-	$2KH$
9. Communication overhead	C	KH	KH
	M	$2\|p\|+2\|q\|$	KH
	PG	$2\|p\|+2\|q\|$	H

Table 5.1: The number of cryptographic operations of KSLv1 and KSLv2 at client, merchant, and payment gateway, respectively

Cryptographic Operations		SET	iKP	Antovski's protocol
1. Public-key encryption	C	1	1	1
	M	1	-	-
	PG	1	-	1
2. Public-key decryption	C	-	-	1
	M	1	4	-
	PG	2	1	1
3. Digital signature	C	1	1	1
	M	3	1	-
	PG	1	1	1
4. Signature verification	C	2	3	1
	M	2	2	-
	PG	1	2	1
5. Symmetric operation	C	1	-	-
	M	-	-	2
	PG	1	-	-
6. Hash function (H)	C	3	2	-
	M	2	4	-
	PG	-	1	-
7. Keyed-hash function (KH)	C	-	-	-
	M	-	1	-
	PG	-	-	-
8. Key generation	C	-	-	-
	M	-	-	-
	PG	-	-	-
9. Communication overhead	C	$3\|p\|+3\|q\|$	$3\|p\|+3\|q\|$	$2\|p\|+2\|q\|$
	M	$3\|p\|+3\|q\|$	$3\|p\|+3\|q\|$	-
	PG	$3\|p\|+3\|q\|$	$3\|p\|+3\|q\|$	$2\|p\|+2\|q\|$

Table 5.2: The number of cryptographic operations of SET [Mas97], 3KP [BGH$^+$00], and Antovski's protocol [AG03] at client, merchant, and payment gateway, respectively

off-line which results in the reduction of computation and time consumption at engaging parties, especially the client.

In terms of communication overhead, it is clear that the proposed KSLv1 and KSLv2 protocols have less overhead than existing protocols. Particularly, the messages sent to and from the client in KSLv1 and KSLv2 do not have communication overhead compared to other protocols that have the overhead due to RSA-based operations. This results in less amount of data transferred during transactions.

It can be seen that KSLv1 and KSLv2 have advantages over existing account-based payment protocols in terms of the reduction of computation and communication load at the mobile client. As a result, using KSLv1 and KSLv2 greatly increase the transaction performance compared to existing account-based payment protocols, especially Antovski *et al.*'s protocol [AG03] which was claimed to work well in wireless environments. Therefore, the transaction performance of the payment system based on the proposed protocols is more likely to satisfy acceptable transaction performance (ATP) stated in the Definition 3.14 of the proposed model than that of the systems based on the existing protocols.

5.7 Applying KSLv1 and KSLv2 to the Proposed Mobile Payment Framework

The main purpose of designing a non proxy-based payment protocol is to operate efficiently in wireless environments without any assistance from either proxy server or mobile agent. Applying the non proxy-based payment protocol to either proxy-based or agent-based framework will increase transaction performance of the system. In the previous section, we have shown that the proposed KSLv1 and KSLv2 are more suitable for wireless environments than existing account-based payment protocols. In this section, we show that the proposed protocols can be applied well to the proposed mobile payment framework presented in chapter 4 which will enhance transaction performance of the system considerably, yet satisfy transaction security properties and trust relationships among engaging parties specified in the proposed mobile payment model.

To do so, a proxy server is set up as the environment where the mobile agent performs highly computational operations. The proxy server is maintained by the payment system provider (or the payment gateway). Applying KSLv1 and KSLv2 to the proposed framework can be done as follows:

1. The client sends an agent $A_1(C)$ to exchange necessary information to start the payment session in **Step 1** of both KSLv1 and KSLv2.
2. In KSLv1, the client generates $MAC[(Price, h(OI), ID_M), Y_i]$ and sends another agent $A_2(C)$ containing the following message to the proxy server:

   $$A_2(C) = \{MAC[(Price, h(OI), ID_M), Y_i], OI, ID_C, Price, X_i, X_{i+1}\}$$

   In KSLv2, the client generates $MAC[(Price, h(OI), ID_M), Y_i]$ and $h(Y_i)$. Then, $A_2(C)$ brings all of the above information including $h(Y_i)$ to the proxy server.
3. At the proxy server, $A_2(C)$ generates the message in **Step 2** of KSLv1 and KSLv2. Then $A_2(C)$ travels to the merchant. After receiving the message, the merchant performs the transaction as per the protocol steps of KSLv1 and KSLv2. The agent $A_2(C)$ stays at the merchant until receiving the message in **Step 6**. Then, $A_2(C)$ returns to the client with the result of the transaction.

It can be seen that the most concerned information from the above system is X_i and X_{i+1}. Assuming that the mobile agent is not highly secured, these two keys may fall into an attacker's hands. However, the attacker may not be able to take advantages from them because she cannot easily retrieve Y_i from $MAC[(Price, h(OI), ID_M), Y_i]$. In addition, $MAC[(Price, h(OI), ID_M), Y_i]$ specifies the intended recipient of the requested price ($Price$), that is the merchant, with ID_M. As long as the MAC algorithm is not compromised, the key Y_i is still secure.

Note that the payment gateway is chosen to perform the task of the proxy server instead of the issuer as in the proposed SET framework presented in chapter 4 in order to prevent the impersonation by the issuer. It can be seen that the message brought with the agent to the proxy server contains the secret key X_i and X_{i+1}. As the partial trust relationship between the client and the issuer does not mention the trustworthiness of the issuer regarding not to impersonate as the client, if the issuer is assigned to perform the task of the proxy server, X_i and X_{i+1} will be known to the issuer that will offer an opportunity to impersonate as the client. Particularly, the issuer can construct the fake client's *Payment Ordering* request from the compromised X_i and X_{i+1} because the issuer already have Y_i.

5.8 Summary

In this chapter, we have presented two account-based mobile payment protocols, KSLv1 and KSLv2, that are suitable for wireless environments in terms of security and performance. They overcome the constraints of wireless environments due to limited capability of mobile devices by deploying simple cryptographic operations at the client side. The proposed protocols are suitable for different applications in that, KSLv1 is suitable for a system whereby a mobile client communicates with fixed-network merchant and payment gateway, whereas KSLv2 is suitable for a system that all engaging parties are mobile users. The applications of KSLv2 have been presented in section 5.5.

In our protocols, we reduced the frequency of key update process by deploying an efficient off-line session key generation technique. We analyzed the security of KSLv1 and KSLv2 by considering the security of the session keys. The results have shown that the session keys generated by our key generation technique are secure against various kinds of attacks.

According to poor reliability of wireless networks, the structure of the proposed protocols, which are composed of short and simple message passes, offers the ability to secure transactions after the system is recovered from failures.

Compared to existing payment protocols, KSLv1 and KSLv2 offer practical usefulness and advantages over SET [Mas97], iKP [BGH$^+$00], and Antovski *et al.*'s protocols [AG03] as the following:

- The protocol structure of the proposed protocols and the cryptographic technique applied to them are simpler, yet have more powerful capability, than those applied to the existing protocols.

- KSLv1 and KSLv2 satisfy transaction security properties provided by SET and iKP protocols. Especially at the client side, although no public-key operations is exploited, the security properties at the client are still provided at the same level as that of SET and iKP protocols.

- The proposed protocols have lower computation at the client side because they do not deploy any public-key operations at the client side. Thus, no mobile devices with advanced features is required.

- The client in the proposed protocols is not required to trust the payment gateway as required in SET and iKP protocols. Thus, the client can ensure that her confidential information will not be compromised by the payment gateway.

- The client is not required to send her credit-card information during the transaction. The issuer can identify the client and infer the client's credit-card information from the session keys generated from the master secret shared between them. Therefore, the client's credit-card information is not susceptible to attacks.

- Compared to Antovski *et al.*'s protocol which was claimed to work well in wireless environments, our KSLv1 and KSLv2 have lower computational and communication load. This infers that the proposed protocols have better performance than Antovski *et al.*'s protocol when deployed in the same communication environment.

We have shown that the proposed KSLv1 and KSLv2 have better performance than the existing non-proxy, account-based payment protocols. As a result, a payment system that is based on the proposed framework previously presented in chapter 4 and deploys KSLv1 or KSLv2 as the underlying payment protocol offers better transaction performance than the systems based on the existing frameworks. Therefore, mobile users can have efficient and secure payment transactions, and it may gain more acceptability than existing payment protocols.

In addition, we have tried to show that the payment systems based on KSLv1 and KSLv2 protocols satisfy the practical and secure mobile payment system stated in the Definition 3.15 of the proposed model. Recall that, to satisfy the proposed model, a payment system must satisfy $Goals$, PR, $Trust$, $TSec$, and ATP. In this chapter, we have shown that KSLv1 and KSLv2 satisfy $Trust$, $TSec$, and ATP. The other properties will be analyzed and discussed later in chapter 7, whereby an accountability logic will be used as the major analysis tool.

Chapter 6

Validation of KSLv1 Protocol

Several payment protocols have been proposed and discussed in chapter 2. These protocols were claimed to be applicable to wireless environments. However, they were presented as theoretical approaches, whereas only some of them have been implemented in the real world. Their ability to perform mobile payment transactions satisfying all the requirements mentioned in section 2.1 is still doubtful.

In chapter 5, we have presented two account-based payment protocols, KSLv1 and KSLv2. They have been mathematically analyzed that they are capable of performing transactions with better performance than existing payment protocols [BGH$^+$00, Mas97, AG03]. As well as other existing mobile payment protocols, without a concrete implementation, it may not possible to claim the superiority of our protocols and implementable in the real world applications.

In this chapter, we present detailed implementation of KSLv1, one of the proposed account-based mobile payment protocols, to demonstrate its applicability to wireless environments. We set up a system where a client performs transactions through a PDA (Personal Digital Assistant) over a wireless LAN (Local Area Network). The reason that PDA was chosen as a client-side device in our implementation is that PDAs are becoming more popular and having higher sales compared to other kinds of mobile devices. According to the statistics shown in [ePa04a], it is believed that by the year 2005, the global sales on PDAs will be US$3.1 billion, while the sales on mobile phones will only be US$0.3 billion.

We discuss the suitability of various kinds of available cryptographic algorithms to be applied to our system not only to increase transaction performance, but also to achieve reasonable security. We measure the results of our implementation in terms of the time taken to complete each transaction to show that KSLv1 is capable of performing mobile payment transactions with satisfactory performance. Moreover, the relatively small size of the client-side application can also be installed on other kinds of limited storage mobile devices e.g. mobile phones.

This chapter is organized as follows. Section 6.1 presents environment settings of our system. In section 6.2, the system design of KSLv1 protocol is demonstrated. Section 6.3 describes the detailed implementation of KSLv1. In section 6.4, we discuss the results from the implementation. Section 6.5 summarizes our work.

6.1 Environment Settings

6.1.1 Communication Environment

As presented in section 5.4, KSLv1 protocol is composed of 5 parties: client, merchant, payment gateway, issuer, and acquirer. As the operations and the communications between the issuer and the acquirer are performed over a banking private network, they are assumed to be out of the scope of our implementation. The following presents the details of hardware and software used in our implementation.

Hardware

1. Client Hardware

    - Palm Tungsten C, 400 Mhz Intel Processor with XScale Technology, 64 MB RAM (51 MB usable)
    - IP Address: 211.28.118.164 (located in Australia)

2. Merchant Hardware

    - Intel Pentium Desktop 2.4 GHz Pentium 4 Processor, 512 MB RAM, 80 GB Harddisk
    - IP Address: 203.125.78.11 (located in Singapore)

3. Payment Gateway Hardware

    - Intel Pentium Desktop 2.4 GHz Pentium 4 Processor, 512 MB RAM, 160 GB Harddisk
    - IP Address: 210.49.106.251 (located in Australia)

The above machines are set up in different locations to emulate the scenario whereby the client, the merchant, and the payment gateway are situated in different geographical locations.

Software

1. Client

    - Palm OS version 5.2.1
    - IBM's Java Virtual Machine
    - Java 2 Platform, Micro Edition (J2ME)
        - Mobile Information Device Profile (MIDP) version 2.0

    - Connection Limited Device Configuration (CLDC) version 1.1
    - Wireless Toolkit (WTK) version 2.1
  - Proguard version 2.1 for obfuscating

2. Merchant/Payment Gateway
   - Windows XP Professional
   - Sun's Java Virtual Machine
   - Java 2 Platform, Standard Edition (J2SE) version 1.4.2

3. General
   - TextPad version 4 for source code editing
   - Cryptographic APIs from Legion of the Bouncy Castle
   - IEEE 802.11b connection with WEP encryption

6.1.2 Chosen Cryptographic Operations

In the design of our application, suitable algorithms for encryptions and hash functions have to be chosen by considering their security and computational requirements. From [RRP02, PRRJ03], the computational requirement is directly related to energy consumption of operating devices in that the higher computation the algorithm requires, the higher energy it consumes. Referred to section 3.1.11 of the proposed model, a practical and secure mobile payment system should deploy low computational, yet acceptably secure cryptographic algorithms. Tables 6.1 and 6.2 show the comparisons of some popular cryptographic algorithms [RRP02, PRRJ03].

In this section, we present and discuss the cryptographic algorithms that were chosen for our implementation as follows:

- **Symmetric-key algorithm:** from Table 6.1, it can be seen that, compared to 3DES (Triple Data Encryption Standard) algorithm [MvOV97],

Algorithms	3DES	AES
Processing power (MIPS) required at 10Mbps	535.9	206.3
Energy used in encryption/decryption (μJ/byte)	6.04	1.21

Table 6.1: Comparison between 3DES and AES algorithms in terms of computational requirement and energy consumption

Algorithms	MD5	SHA-1
Processing power (MIPS) required at 10Mbps	33.1	115.4
Energy used in hashing (μJ/byte)	0.59	0.76

Table 6.2: Comparison between MD5 and SHA-1 algorithms in terms of computational requirement and energy consumption

AES (Advanced Encryption Standard) algorithm [SASR01] requires not only less energy consumption, but also less computation. This results in more suitability to operate on limited capability mobile devices.

Comparing in terms of security of each of the above algorithms, it is shown in [Cer03] that 3DES with 112-bit key length is able to provide the equivalent security to RSA public-key algorithm with only 2,048-bit key length, while AES with 128-bit length of key is able to provide the security equivalent to RSA public-key algorithm with 3,072-bit length of key.

Due to the fact that AES provides higher security, faster operation, and less energy consumption compared to 3DES, AES was chosen as the symmetric-key encryption algorithm for our implementation. In particular, the AES algorithm with 128-bit key was deployed in our implementation.

Note that, it is believed that the strength of AES algorithm with 128-bit key length is secure enough for many years to come [Cer03]. If higher security is required, the AES algorithm is able to operate with 192-bit and 256-bit key length that deliver the equivalent security to the RSA algorithm with 7,680-bit and 15,360-bit key length [Cer03], respectively.

- **Hash function:** MD5 algorithm [MvOV97] was chosen in our implementation because, from Table 6.2 which shows the comparison between two of the most popular hash algorithms in terms of computational requirement and energy consumption, MD5 requires less computation and energy consumption than SHA-1 (Secure Hash Algorithm version 1) [MvOV97].

  Moreover, as hash function is to be used in session key generation process (which will be discussed later in this chapter) and each session key is used as both the encrypting key and the key for keyed-hash function, MD5 was chosen for our implementation as it is able to produce the equal length of output to the length of an AES key (128 bits).

- **Keyed-hash function:** We have chosen HMAC-MD5 (Hashed Message Authentication Code with Message Digest 5) algorithm [KBC97] to perform keyed-hash operations in our implementation because the key length of HMAC-MD5 is equivalent to the length of each session key. Moreover, HMAC-MD5 is considered the most secure keyed-hash algorithm that is available for wireless networks. Note that in KSLv1 protocol presented in section 5.4, HMAC-MD5 is used to secure the *Debit* request sent from the client to the issuer in the *Payment Protocol.*

- **Public-key algorithm:** RSA algorithm with the default key size of 1,024 bits was chosen to perform public-key operations in our implementation. As the public-key operation between the client and the merchant is performed only once over the wireless interface to allow the client to send the secret key to the merchant, the main use of public-key operations in KSLv1 protocol is to secure the message transfer between the merchant and the payment gateway.

It can be noted that the J2ME platform [Sun05] does not support any cryptographic algorithms (at the time of implementing the system), the cryptographic algorithms needed in the implementation of KSLv1 protocol were brought from a third-party cryptography provider. The Legion of the Bouncy Castle's APIs [Leg04] was selected due to its support for both the J2SE and J2ME platforms.

6.2 System Design

6.2.1 KSL Wallet Design

In KSLv1 protocol, the client needs to have the session keys X_i and Y_i at the beginning of each transaction. Assume that the client has already downloaded the client-side software called *KSL Wallet* and installed it on her device. Moreover, the secrets Y and CCI have been distributed to the client and the set of session keys Y_i, where $i = 1, ..., n$, has been generated. The keys CCI, Y, and Y_i are stored in separate key files on each party's device. Particularly, the key files stored on the client's PDA are password-protected. The client is requested to set up her own password during the KSL wallet installation. Later on, the client is required to enter this password to open the program and to authorize the payment transactions. In particular, the password protects unauthorized users to access all the key files. Figure 6.1 shows the login screen of the KSL wallet.

The KSL wallet performs three functionalities: session key generation, merchant registration, and payment. These functionalities follow the session key generation, *Merchant Registration Protocol*, and *Payment Protocol* described in section 5.4. In the development of the KSL wallet, two midlets were created for each phase of the KSLv1 protocol (the *Merchant Registration Protocol* and the *Payment Protocol*), and another class was created to contain all of the cryptographic algorithms used by the midlets. The class diagram in Figure 6.2

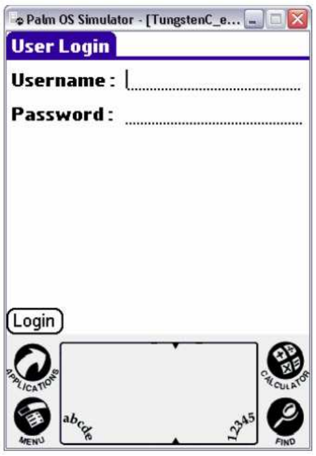


Figure 6.1: Login screen with password authentication of KSL wallet

presents all functionalities of the KSL wallet.

From Figure 6.2, the KSL wallet contains three main classes:

- `cMerchantRegistrationProtocol` generates a session key K and the secret key X and transfers X together with the client's information to the merchant as per the *Merchant Registration Protocol* of KSLv1.
- `cPaymentProtocol` performs payment transactions as per the *Payment Protocol* of KSLv1.
- `cEngine` contains necessary cryptographic functions for all the client's classes.

Moreover, the KSL wallet has session key generation function shown in Figure 6.3. To generate a new session key from the master key, the `GenerateChainSecret` class takes in two values: a master key and a random number i. Upon receiving the master key, the `BigInteger` representation of the master key is created and, using the `shiftLeft()` method to perform left-cyclic shift in the `BigInteger` class, the number of zeros according to the value of i is added to the right of the master key. Then, the new value will be put through a MD5 hash function to produce a new 128-bit session key. The

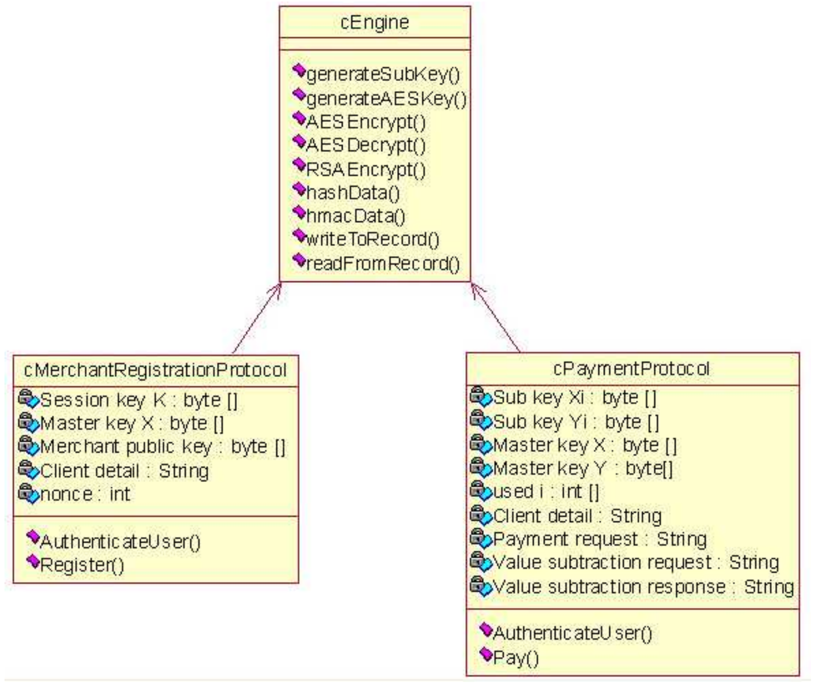


Figure 6.2: Class diagram of KSL wallet

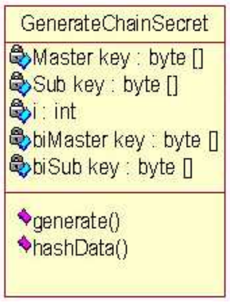


Figure 6.3: Class diagram of session key generation

`GenerateChainSecret` class is also contained in both merchant and payment gateway.

Note that the above technique can be directly applied to generate the set of session keys X_i from the key X, whereas to generate the set of session keys Y_i from the secret Y, the client's credit-card information CCI is treated as the `BigInteger` and added to the value of Y before using the `shiftLeft()` method.

Note also that, in our implementation, each session key is generated at the beginning of each transaction in order to reduce the storage requirement on the client's mobile device. The discussion regarding this issue will be presented in section 6.4.2.

6.2.2 Merchant Design

In the merchant application, only the back-end of the merchant server was implemented. Figure 6.4 shows the class diagram of the merchant application.

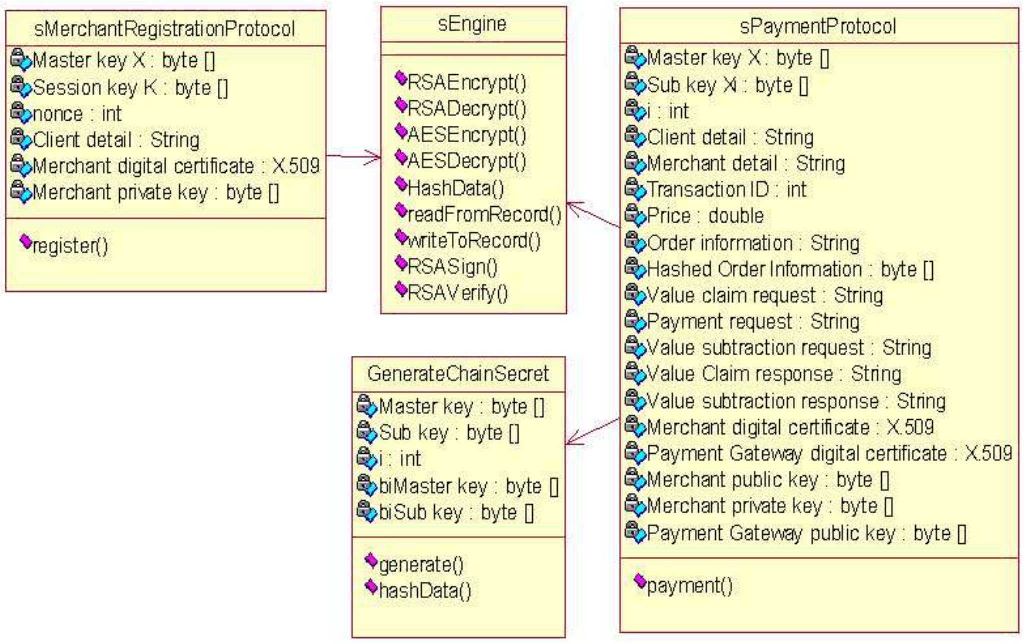


Figure 6.4: Class diagram of merchant

From Figure 6.4, the merchant application contains four classes:

- `sMerchantRegistrationProtocol` interacts with the client regarding the exchange of the key X in the *Merchant Registration Protocol*.
- `GenerateChainSecret` generates the set of session key X_i, where $i = 1, ..., n$, from the key X as a result of completing the *Merchant Registration Protocol*.

- `sPaymentProtocol` receives the session key X_i from the class `GenerateChainSecret` and interacts with the client regarding the payment transaction.

- `sEngine` contains necessary functions for all of the merchant's classes.

6.2.3 Payment Gateway Design

In our implementation, the payment gateway application acts on behalf of the issuer and the acquirer at the Internet side. As the flow of information between the issuer and acquirer exists within a private banking network, we do not implement the issuer and the acquirer. They are assumed that all transactions relevant to them are successful within a limited time. Figure 6.5 depicts the class diagram of the payment gateway used in our implementation.

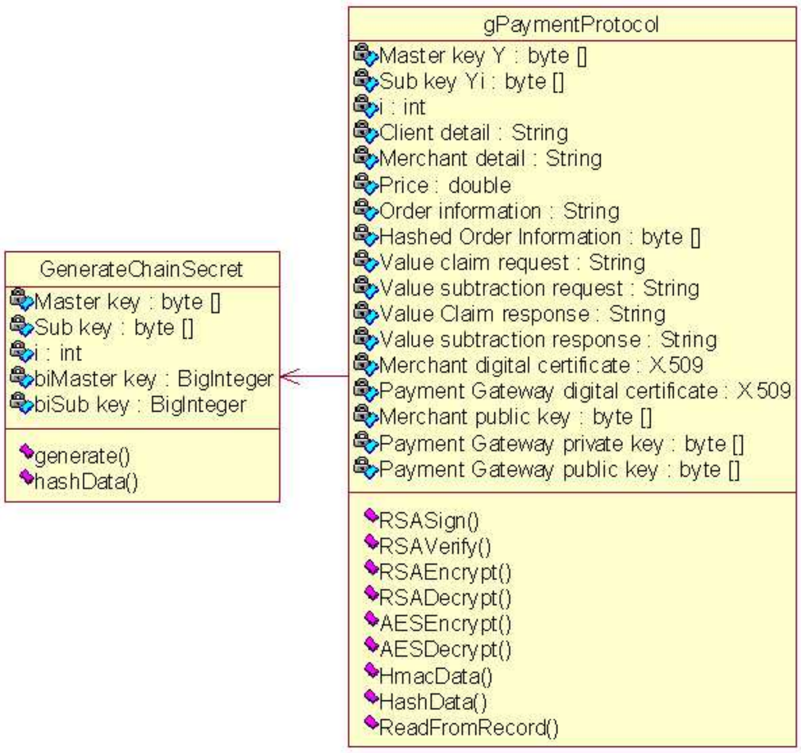

gPaymentProtocol
Master key Y : byte []
Sub key Yi : byte []
i : int
Client detail : String
Merchant detail : String
Price : double
Order information : String
Hashed Order Information : byte []
Value claim request : String
Value subtraction request : String
Value Claim response : String
Value subtraction response : String
Merchant digital certificate : X509
Payment Gateway digital certificate : X509
Merchant public key : byte []
Payment Gateway private key : byte []
Payment Gateway public key : byte []
RSASign()
RSAVerify()
RSAEncrypt()
RSADecrypt()
AESEncrypt()
AESDecrypt()
HmacData()
HashData()
ReadFromRecord()
GenerateChainSecret
Master key : byte []
Sub key : byte []
i : int
biMaster key : BigInteger
biSub key : BigInteger
generate()
hashData()

Figure 6.5: Class diagram of payment gateway

From Figure 6.5, the payment gateway application is composed of the following classes:

- `GenerateChainSecret` generates the set of the session keys Y_i from the secret Y. Note that this function was supposed to be performed by the issuer. However, as the issuer is not implemented in our system, in this implementation, we assume that the issuer and the payment gateway are the same party so that the computational tasks of both the issuer and the payment gateway can be performed on the same server.
- `gPaymentProtocol` interacts with both the client and the merchant regarding the payment transaction.

In the next section, the detailed implementation of KSLv1 protocol will be presented.

6.3 Implementation

6.3.1 Merchant Registration

After the client and the merchant have agreed on the price and goods descriptions and the client has never made any payment to the merchant before, the merchant sends her digital certificate to the client. This certificate contains the merchant's public key. The client runs the `cMerchantRegistrationProtocol` midlet to generate the key X and share X with the merchant. To do so, the `generateAESKey()` method is called twice to generate the session key K and the secret key X. After the keys have been generated, the key K is then encrypted with the merchant's public key using the `RSAEncrypt()` method and then transmitted to the merchant. On receiving the message, the merchant decrypts the message using the `RSADecrypt()` method to retrieve the key K.

Then, the KSL wallet encrypts the client's details, the master key X, and a nonce n with the key K (as per the *Merchant Registration Protocol* of KSLv1 presented in section 5.4.3) by using the `AESEncrypt()` method. The format of the message is given in Table 6.3.

Fields	Client's Data	X	n
Size (bits)	variable	128	128

Table 6.3: Format of the message sent from client in *Merchant Registration Protocol*

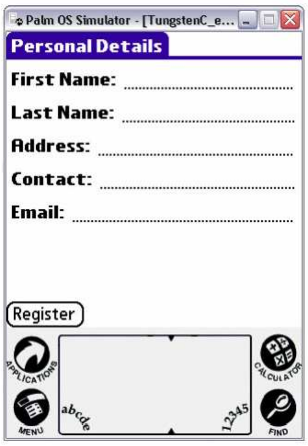
From Table 6.3, it can be seen that the length of the key X is 128 bits because it is generated from the AES engine, whereas n can be any length. To provide satisfactory security, n is set to have 128 bits in length.

On receiving the message, the merchant decrypts the message using the `AESDecrypt()` method to retrieve the client's information including the key X. The merchant then encrypts the nonce n with the key K using the `AESEncrypt()` method and sends the encrypted message to the client as the confirmation of the registration.

At the client side, the client retrieves the confirmation of her registration by decrypting the message using the `AESDecrypt()` method. After the registration is successful, the client's information and the key X are stored at the merchant.

Then, both client and merchant generate the set of session keys X_i, where $i = 1, ..., n$, from X by using the `GenerateChainSecret` class and store X and X_i in separate key files on each party's device. Note that the file which contains the set of X_i is also protected with the same password as that of Y_i. Figure 6.6 illustrates the KSL wallet during the *Merchant Registration Protocol*. On the left screen, the KSL wallet is prompted for the client to enter her details, whereas the right screen shows the successful registration.

After the registration is successful, the client is prompted to enter her


Palm OS Simulator - [TungstenC_e...

Personal Details

First Name:

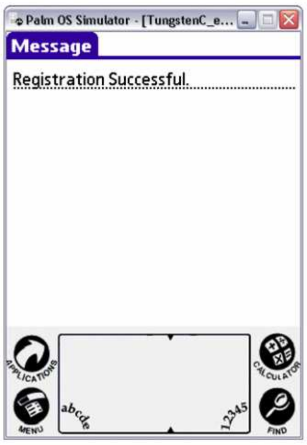
Last Name:

Address:

Contact:

Email:

Register


Palm OS Simulator - [TungstenC_e...

Message

Registration Successful.

Figure 6.6: Snapshots of KSL wallet while performing *Merchant Registration Protocol*

personal information e.g. name and contact information including her semi-secret credit-card number. After filling the information, the client is required to enter a password. In the client's point of view, the password and her credit-card number are used to confirm the payment for further processing. Whereas, in the wallet's point of view, the credit-card number and the password are used to access the key files and append the newly generated Y_i to the file. Note that, later on, after each Y_i has been used, only its index i is stored in the Y_i file instead of the key itself to increase the storage space of the client's PDA. After the client enters the correct password, the payment is then processed as shown in the next section.

6.3.2 Making Payment

In this section, the detailed implementation of the *Payment Protocol* of KSLv1 protocol is presented. Making the payment to the merchant is performed by `cPaymentProtocol` midlet in the KSL wallet. The KSL wallet application generates a random number i. The client generates the session key X_i based on the corresponding i value. After getting the random i value, the client sends i together with $TIDReq$ and $MIDReq$ to the merchant. On receiving the message, the merchant generates X_i based on the i value and encrypts TID and ID_M with X_i. Then the merchant sends the encrypted message to the client.

The client then creates a *Payment Ordering* request as well as a *Debit* request message to be sent to the merchant and the issuer, respectively. Their formats are shown in Tables 6.4 and 6.5, respectively.

Fields	OI	$Price$	Client's Data	Issuer's Data	*Debit* request
Size (bits)	variable				128

Table 6.4: Message format of *Payment Ordering* request of KSLv1

Fields	$Price$	$h(OI)$	Merchant's Data
Size (bits)	variable	128	variable

Table 6.5: Message format of *Debit* request of KSLv1

Note that OI stands for order information and $h(OI)$ stands for the hash value of order information. The *Debit* request is performed a HMAC process with the key Y_i by using the `hmacData()` method. The output of HMAC is encrypted together with the *Payment Ordering* request using the `AESEncrypt()` method with the session key X_i before being sent to the merchant.

On receiving the requests from the client, the merchant, having X_i, decrypts the message using the `AESDecrypt()` method. Then, the merchant encrypts the *Debit* request sent from the client and the amount payable by the client with the payment gateway's public key using the `RSAEncrypt()` method. This encrypted message is combined with necessary information to form the *Credit* request. The format of the *Credit* request is given in Table 6.6.

Fields	*Debit* request	$Price$
Size (bits)	128	variable

Table 6.6: Message format of *Credit* request of KSLv1

Note that, to get the payment gateway's public key, the payment gateway sends the merchant its digital certificate which contains its own public key as a result of the request by the merchant. The digital signature is created by using the `RSASign()` method with the merchant's private key. The encrypted data together with the digital signature is then transmitted to the payment gateway.

On receiving the requests from the merchant, the payment gateway verifies the merchant's signature by using the `RSAVerify()` method with the merchant's public key. Upon verifying the digital signature, the payment gateway must decrypt the data from the merchant by the `RSADecrypt()` method to retrieve the *Debit* request, the *Credit* request, and the random number i. The payment gateway then uses the `generate()` method in the `GenerateChainSecret` class to generate the appropriate session key Y_i, after which, it will decrypt the *Debit* request using the `AESDecrypt()` method with the session key Y_i.

As the issuer and the payment gateway are assumed to be the same party, the payment gateway then creates a *Debit* response encrypted with session key Y_i by using the `AESEncrypt()` method.

To generate a *Credit* response, $h(OI)$ and Yes/No are encrypted with the merchant's public key using the `RSAEncrypt()` method. Then both *Debit* and *Credit* responses are signed with the payment gateway's private key using the `RSASign()` method and then transmitted to the merchant. The format of the *Credit* response and the *Debit* response are given in Tables 6.7 and 6.8, respectively.

Fields	$h(OI)$	Yes/No
Size (bits)	128	variable

Table 6.7: Message format of *Credit* response of KSLv1

Fields	$h(OI)$	Yes/No
Size (bits)	128	variable

Table 6.8: Message format of *Debit* response of KSLv1

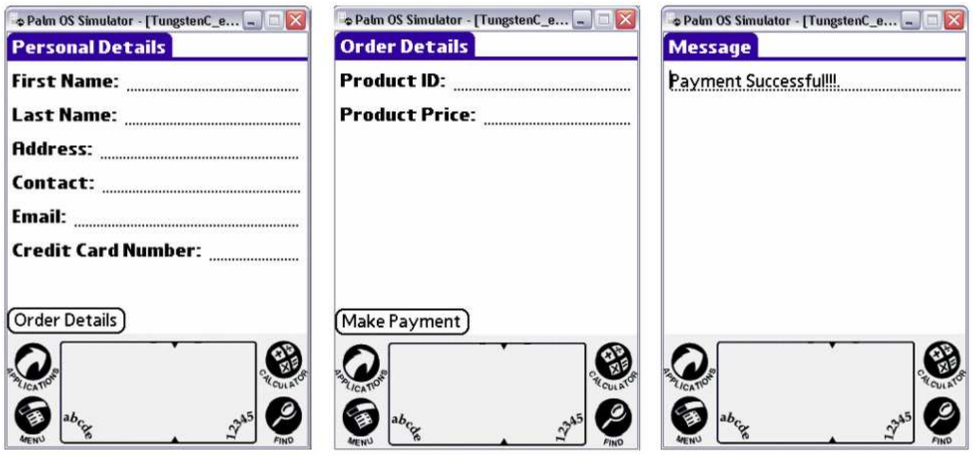
On receiving the responses from the payment gateway, the merchant verifies the payment gateway's signature by using the `RSAVerify()` method. Then, the merchant retrieves the *Credit* response by using the `RSADecrypt()` method. As the *Debit* response is encrypted with Y_i, the merchant generates X_{i+1}, encrypts the *Debit* response using the `AESEncrypt()` method with X_{i+1}, and then sends the client the encrypted message as a *Payment Ordering* response.

On receiving the message, the client retrieves the *Payment Ordering* response by using the `AESDecrypt()` method to decrypt the message. Its message format is given as follows:

Fields	$h(OI)$	Yes/No
Size (bits)	128	variable

Table 6.9: Message format of *Payment Ordering* response of KSLv1

Figure 6.7 illustrates snapshots of the KSL wallet during the *Payment Protocol*. On the left screen, the KSL wallet is prompted for the client to enter her details including the credit-card information. The middle screen shows the product information and its price. The right screen shows the successful payment.


Palm OS Simulator - [TungstenC_e...

Personal Details

First Name:

Last Name:

Address:

Contact:

Email:

Credit Card Number:

Order Details

Palm OS Simulator - [TungstenC_e...

Order Details

Product ID:

Product Price:

Make Payment

Palm OS Simulator - [TungstenC_e...

Message

Payment Successful!!!

Figure 6.7: Snapshots of KSL wallet while performing *Payment Protocol*

6.4 Results and Discussions

6.4.1 Transaction Time

In this section, we demonstrate the results obtained from the implementation regarding the time taken to perform various parts of KSLv1 protocol as well as the overall time taken to complete a payment transaction. We collected this information by performing 10 executions with different sets of data. Tables 6.10 and 6.11 show the average time taken for the execution of the *Merchant Registration Protocol* by the client and the merchant, respectively.

Data	AESGen	RSAEnc	AESEnc	TMRP
Time (ms)	20	250	50	16,840

Table 6.10: Time taken at client on performing *Merchant Registration Protocol*

From Table 6.10,

- AESGen stands for the AES key generation used to generate the key X,
- RSAEnc stands for the RSA encryption during the distribution of the key K from the client to the merchant. In this process, the key K is encrypted with the merchant's public key and sent to the merchant,
- AESEnc stands for the AES encryption of $\{ID_C, X, n\}$ with the key K, and
- TMRP stands for the total time taken to complete the *Merchant Registration Protocol*.

Data	RSADec	AESDec	AESEnc	TMRP
Time (ms)	94	15	< 1	11,981

Table 6.11: Time taken at merchant on performing *Merchant Registration Protocol*

From Table 6.11,

- RSADec stands for the RSA decryption of the key K with the merchant's private key,
- AESDec stands for the AES decryption of $\{ID_C, X, n\}$ with the key K,
- AESEnc stands for the AES encryption of n with the key K, and
- TMRP stands for the total time taken to complete *Merchant Registration Protocol*.

It should be noted that the time taken in the implementation does not include the time taken by the client to input the data. In the implementation, we tested by entering maximum number of characters allowed into each individual field and found that the maximum time taken by the client and the merchant on performing the *Merchant Registration Protocol* was 18,730 milliseconds and 13,590 milliseconds, respectively. These timings are not much different from the average time calculated in Tables 6.10 and 6.11.

Processes	X_i Generation	Y_i Generation	AES Decryption	Total time
Time (ms)	10	10	28	7,159

Table 6.12: Time taken at client on performing *Payment Protocol*

Processes	X_i Generation	Digital Signature Generation	Digital Signature Verification	Total time
Time (ms)	10	62	10	3,119

Table 6.13: Time taken at merchant on performing *Payment Protocol*

Processes	Y_i Generation	Digital Signature Generation	Digital Signature Verification	Total time
Time (ms)	10	60	10	1,922

Table 6.14: Time taken at payment gateway on performing *Payment Protocol*

Tables 6.12 to 6.14 demonstrate the average time taken on performing the *Payment Protocol* by the client, the merchant, and the payment gateway, respectively. The average time is calculated based on the time taken from 10 executions of the *Payment Protocol*. Note that all the timings shown above do not include the time taken by the client to fill in all the necessary information e.g. product information, price, including credit-card information.

We tested by entering the maximum number of characters into each individual field and found that the maximum time taken by the client, the merchant, and the payment gateway to perform the *Payment Protocol* was 10,510 milliseconds, 4,377 milliseconds, and 2,103 milliseconds, respectively.

It can be seen that, on performing the first transaction with the merchant, the average total time that the client spent during a KSLv1 payment transaction (both the *Merchant Registration Protocol* and the *Payment Protocol*) was only 24 seconds ($16.84 + 7.16 = 24$ seconds). In next transactions that the client does not have to do the registration with the merchant, the total average time to complete each transaction will reduce to only 7.16 seconds. With this amount of time to complete a transaction, KSLv1 protocol is potentially able to conduct payment transactions in wireless environments.

6.4.2 Storing Keys

Generating and storing secret keys affect the size of our application. In KSLv1 protocol, the client has two options regarding storing keys; she may either generate the entire sets of the session keys (X_i and Y_i) at a time after sharing X and Y with the merchant and the issuer, respectively, or generate each X_i and Y_i at the beginning of each transaction. If the former is chosen, the client is required to store all session keys on her mobile device. This option may take a lot of space that is not suitable for very limited storage devices such as mobile phones. If the latter is chosen, it does not take a lot of space on the PDA as only a set of index of used session keys needs to be stored on the client's device. However, generating X_i and Y_i before each transaction takes certain amount of time which may reduce transaction performance of the system.

In our implementation, the latter option was chosen. Although the concern about the storage space of the PDA is not critical, we wanted to see if KSLv1 is able to perform well with the higher computational load option.

6.4.3 Application Size

As mobile devices have limited memory space, the size of application stored on mobile devices is an important issue to be taken into consideration during the development of mobile applications. The KSL wallet program has an acceptable file size of 122 kB for a Palm Pilot. It can be seen that the compared to the memory size of the PDA used in our implementation, the KSL wallet requires only 0.24% of the entire memory. Whereas the size of the KSL wallet the size of the executable `jar` file for a mobile phone is 90 kB which is easily installed and fitted to the mobile phone.

6.5 Summary

In this chapter, we have shown that limitations of wireless environments can be overcome by a careful choice of cryptographic algorithms applied to mobile devices and a secure, lightweight payment protocol by presenting an implementation of KSLv1 protocol, one of the proposed account-based mobile payment protocols presented in chapter 5.

From the results obtained from our implementation, it is obvious that payment transactions over a wireless network can be conducted by KSLv1 protocol efficiently. KSLv1 has overcome the limitations of both mobile devices and wireless networks that have been discussed in section 1.3.1 as follows:

- To deal with limited storage of mobile devices, the KSL wallet requires only 150 kB to be stored on a Palm Pilot and 90 kB to be stored on a mobile phone.

- The client's KSL wallet has two options regarding storing keys which cover all kinds of mobile devices. One is generating the entire sets of session keys X_i and Y_i off-line after the keys X and Y have been distributed. Although this option requires a lot of storage space to store all the session keys on each party's device, it increases transaction performance of the system because each party does not have to generate the session keys at the beginning of each transaction. Thus, this option is suitable for mobile devices with large storage space, such as PDAs.

  The other option is that each session key is generated at the beginning of each transaction and removed after being used in the transaction. According to this option, only the index of the used session keys needs to be stored on the client's device instead of the keys themselves. Thus, it requires less storage than the other option. However, the transaction performance of this option is lower compared to the others because the session key generation process is required at the beginning of every transaction. Therefore, this option is suitable for mobile devices with very limited storage space, such as mobile phones.

- To increase transaction performance, we have chosen lightweight, secure cryptographic algorithms that are suitable for our application. Deploying such algorithms in KSLv1 results in the reduction of message passes and computation at the client's mobile device. As we have seen from the implementation, a payment transaction by KSLv1 can be completed within average 10 seconds.

Chapter 7

Towards Formal Analyses of Mobile Payment Protocols

It is obvious to state that ultimate goals of designing an electronic commerce (e-commerce) protocol are not only providing secure transactions among engaging parties, but the protocol should provide assurance on actions performed by each engaging party. The former can be achieved by satisfying transaction security properties stated in section 3.1.9, whereas the latter can be achieved by a property called "*accountability*".

Accountability of e-commerce protocols is concerned with the ability to show that the particular parties who engage in the protocols are responsible for transactions [KP02]. Particularly, each party must be able to prove to a party who acts as a dispute resolver (or a verifier) that she is honest for the transaction relevant to her. For many years, accountability has been used to resolve disputes among engaging parties [Kai96, KN98]. Recently, Kungpisdan *et al.* [KP02] has shown that the accountability property can be used to specify goals of e-commerce protocols.

In this chapter, we aim to analyze our proposed framework and protocols on the accountability property to show that the payment systems based on the proposed framework or protocols satisfy not only goals and requirements of engaging parties, but also transaction security properties stated in the proposed mobile payment model introduced in chapter 3. This is because, it is shown in chapter 3 that such properties stated in the model can be formalized and represented as statements of Kungpisdan *et al.* (KP)'s logic. However, we found that existing accountability logics including KP's logic are inadequate to analyze mobile payment protocols. We then propose an extension of KP's logic and use it to analyze our proposed framework and protocols.

This chapter is organized as follows. Section 7.1 reviews the accountability property and its applications to e-commerce protocols, and existing formal analyses on the accountability property. Section 7.2 presents overviews of Kungpisdan *et al.*'s accountability logic [KP02]. Section 7.3 discusses the applications of the accountability property to cryptographic protocols for wireless networks. Section 7.4 introduces the proposed logic which is suitable for analyzing the accountability of mobile payment protocols. In section 7.5, we analyze the proposed framework by using the proposed logic. In section 7.6, the analyses of the proposed KSLv1 and KSLv2 protocols on the accountability property are presented. Section 7.7 shows that that party's requirements for payment transactions of iKP protocol [BGH$^+$00] not only can be reasoned by our proposed logic, but they can also be formalized into the party's requirements stated in the proposed formal model previously introduced in chapter 3. Section 7.8 summarizes the chapter.

7.1 Accountability vs E-commerce Protocols

The main goal of developing an e-commerce protocol is to ensure that at the completion of a transaction, all parties who engage in the transaction are convinced that they have authorized messages regarding the transactions related to them [KP02]. For example, a merchant is ensured that a client has requested to purchase goods or services and has paid to her, or the client is ensured that she will receive goods after making the payment to the merchant.

For example, as mentioned in section 4.4.5, one of the goals of SET protocol [Mas97] is "*Client Privacy*". Recall that the client privacy is concerned with client's provability of her authorized payment transaction without revealing goods descriptions and payment information (e.g. credit card number) to a payment gateway (on behalf of an issuer or an acquirer) and a merchant, respectively. SET protocol achieves the client privacy because, at the completion of a transaction, the merchant cannot infer the client's payment information since it is encrypted with the payment gateway's public key, and the payment gateway cannot infer the goods descriptions because the goods descriptions are in hashed form.

It can be seen that the property described above is very important for payment transactions because it can provide guarantees of both the occurrences and the completion of transactions related to all engaging parties. Such a property can be reasoned by the accountability property.

Accountability is very important for financial transactions since it is crucial that each party must be guaranteed on the actions related to the transaction she has performed. For example, a client must be guaranteed that she will receive the goods or services that she has previously requested after the payment has been made to the corresponding merchant.

We consider the accountability as a *high-level* security property which covers basic transaction security properties stated in the Definition 3.10 of the proposed formal model: *party authentication*, *privacy of transactions*, *transaction integrity*, *transaction authorization*, and *non-repudiation of transactions*. To achieve the accountability property in any transaction, all the above transaction security properties must be satisfied, whereas satisfying all the above transaction security properties in a transaction does not guarantee the accountability property. This is because the accountability property also concerns with the transaction-related information. For example, in SET protocol [Mas97], price and goods descriptions are crucial information for the client and the merchant, whereas payment information (PI) must be delivered to the payment gateway. Even though SET protocol satisfies the above basic transaction security properties, the lack of payment-related information provided to engaging parties results in the failure to achieve the accountability property. In particular, it is found that the **PRes** of SET protocol (referred to the SET protocol description in section 2.3.1) does not contain $Price$ to guarantee the client the requested price. Moreover, only the merchant's signature does not guarantee the successful money deduction by the issuer.

From [KP02], the accountability is not only used to resolve disputes among engaging parties, but also to specify goals of e-commerce protocols. Thus, it is worth to consider the accountability as the main security property of the proposed protocol.

Several definitions of accountability [Kai96, KN98, KP02] have been proposed. In Kailar's logic [Kai96], the accountability property is concerned with the ability to prove the association of an originator of a message with some actions to a third party without revealing any private information. However, Kailar's logic can provide reasoning only about the accountability of a signed plain message. It is inadequate to analyze a complex cryptographic message e.g. signed encrypted and/or hashed messages. Moreover, it does not reason about the verifier. Note that we need to reason about the verifier because we need to state about the information that the prover can send to the verifier as proof evidence.

In Kessler *et al.* (KN)'s logic [KN98], to prove the accountability, a prover must be able to provide proof evidence to a verifier, and after the verifier receives the evidence, she is convinced on what the prover wants to prove. However, KN's logic lacks of reasoning about revealing secrets to the verifier. Thus, some information which is considered to be secret may be revealed to the verifier during the transaction.

Recently, Kungpisdan *et al.* (KP) [KP02] proposed a modification of KN's logic [KeNe98]. They analyzed SET and iKP protocols by focusing on authorization of primitive transactions that are previously presented in [Her01]. Kungpisdan *et al.* formalized the accountability by using the modal operator '*CanProve*' which was firstly introduced by Kailar [Kai96]. Consider the following statement:

$$P \textit{ CanProve } \phi \textit{ to } V$$

where P and V stand for a prover and a verifier, respectively, and ϕ stands for a statement regarding some actions. The above statement states that P can prove to V that ϕ is true. Referred to [KP02], this statement can be derived by axiom **P2** that represents the main concept of accountability. The axiom **P2** is shown as follows:

P2: $[\ (V\textit{-is-external-party}) \land (P \textit{ has } X) \land$
$(V \textit{ sees } X \rightarrow V \textit{ believes } \phi)\ ]$
$\rightarrow P \textit{ CanProve } \phi \textit{ to } V$

The axiom **P2** states that if a verifier V is an external party, a party P has a message X, and if V receives X, V believes that a statement ϕ is true, it arrives the conclusion that P can prove to V that ϕ is true. In other words, P initially has evidence X to prove that ϕ is true. P then sends X to V. After V receives X, V is convinced that ϕ, what P wants to prove, is true.

We can see that the axiom **P2** provides reasoning about both prover and verifier. Particularly, the axiom **P2** reasons about what the verifier receives and how she derives a new belief after receiving some information. Another axiom **P3** represents the accountability of a signed message as follows:

P3: $P\ has\ \{X\}_{K_M^{-1}} \wedge P\ CanProve\ (\xrightarrow{K_Q} Q)\ to\ V$
$\rightarrow P\ CanProve\ (Q\ says\ X)\ to\ V$

The axiom **P3** states that if a party P has a message X signed with the private key of a party Q, the public key of Q, and X, and P can prove to a verifier V that the public key of Q refers to Q, then P can prove to V that Q has sent X. We can see that the axiom **P3** can be used to specify the originator of the message in that only the party who possesses K_Q^{-1} can sign the message. Note that the syntax of KP's logic will be described in details in the next section.

7.2 Kungpisdan *et al.*'s Logic

Kungpisdan *et al.* (KP) [KP02] proposed a logic for analyzing accountability property of e-commerce protocols. The details of KP's logic are given as follows

Syntax

Terms

- $\{P, Q, R, V\}$: the set of parties that communicate to one another in a protocol.
- $\{X, Y\}$: the set of messages or message components in a protocol.
- $\{\phi, \psi\}$: the statements derived from protocol messages.
- $\{K_P, K_P^{-1}\}$: the set of the public key and private key of a party P.
- $\{X\}_{K_P}$: the message X encrypted with the public key of a party P.
- $\{X\}_K$: the message X symmetrically encrypted with a shared key K.

- $h(X)$: the hash value of the message X.
- $\xrightarrow{K_Q} Q$: the key K can be used to refer to the party Q.
- $P \overset{K}{\longleftrightarrow} Q$: the key K is a shared key between the parties P and Q.
- *X-is-fingerprint-of-Y* : the message X can be used as a representative (fingerprint) of Y (for example, X may be the hashed form of Y).
- *K-is-decrypting-key-for-*$\{X\}_K$: the key K can be used to decrypt the message $\{X\}_K$.
- $Cert_P$: the certificate of the party P which contains the identity and the public key of P $\{ID_P, K_P\}$.
- $\{M\}_{K_P^{-1}}$: the message M signed with the private key of the party P.
- $MAC(X, K)$: Message Authentication Code (MAC) of the message X with the key K.
- $\langle X\rangle_K$: the message X applied with a single-key cryptographic operation with the key K. $\langle X\rangle_K$ can be symmetric-key encryption $\{X\}_K$, Message Authentication Code $MAC(X, K)$, or hash function $h(K)$.

Formulae

- *P believes* ϕ : P believes that the statement ϕ is true.
- *P sees X* : some party has sent the message X to P and P is able to read X.
- *P has X* : P possesses the message X. P can send X to other parties or use it for further processing.
- *P says X* : P has sent the message X.
- *P CanProve* ϕ *to Q* : P can prove to Q that the statement ϕ is true.

- *P authorized payment-order(P, Q, Price, Date)* : *P* has authorization on making the payment amount *Price* to *Q* on the date of transaction *Date*.
- *P authorized debit(P, Q, Price, Date)* : *P* has authorization on requesting *Q* to deduct the amount *Price* from *P*'s account on the date of transaction *Date*.
- *P authorized credit(P, Q, Price, Date)* : *P* has authorization on requesting *Q* to transfer the amount *Price* to *P*'s account on the date of transaction *Date*.

Axioms

Inference Rules

MP: If ϕ and $\phi \rightarrow \psi$ then ψ

M: If ϕ is a theorem, then P *believes* ϕ is a theorem.
where theorem is a formula which can be derived from axioms alone.

Modalities: KD45-logic

K: P *believes* $\phi \wedge (\phi \rightarrow \psi) \rightarrow P$ *believes* ϕ

D: P *believes* $\phi \rightarrow \neg P$ *believes* $\neg\phi$

4: P *believes* $\phi \rightarrow P$ *believes* P *believes* ϕ

5: $\neg P$ *believes* $\phi \rightarrow P$ *believes* $\neg P$ *believes* $\neg\phi$

Possessions

H1: P *sees* $X \rightarrow P$ *has* X

H2: $(P$ *has* $X_1 \wedge ... \wedge P$ *has* $X_n) \rightarrow P$ *has* $(X_1, ..., X_n)$
where $(X_1, ..., X_n)$ stands for a list of message $X_1, X_2, ..., X_n$, respectively.

H3: P *has* $X \rightarrow P$ *has* $h(X)$

H4: $($ P *has* $(\{X\}_K, K) \wedge P$ *believes* $P \xleftrightarrow{K} Q$ $) \rightarrow P$ *has* X

H5: $(\ P\ has\ (\{X\}_{K_P}, K_P^{-1}) \wedge P\ believes \xrightarrow{K_P} P\ ) \rightarrow P\ has\ X$

H6: $(\ P\ has\ (\{X\}_{K_P^{-1}}, K_P) \wedge P\ believes \xrightarrow{K_P} P\ ) \rightarrow P\ has\ X$

Comprehensions

C1: $P\ sees\ X \rightarrow P\ believes\ P\ sees\ X$

C2: $P\ says\ X \rightarrow P\ believes\ P\ says\ X$

Seeing

SE1: $P\ sees\ (X_1, ..., X_n) \rightarrow (P\ sees\ X_1 \wedge P\ sees\ X_2 \wedge ... \wedge P\ sees\ X_n)$

Saying

SA1: $P\ says\ (X_1, ..., X_n) \rightarrow (P\ says\ X_1 \wedge P\ says\ X_2 \wedge ... \wedge P\ says\ X_n)$

SA2: $P\ says\ X \rightarrow P\ has\ X$

Provability

P1: $(P\ CanProve\ (\phi \rightarrow \psi)\ to\ V)$

$\rightarrow (P\ CanProve\ \phi\ to\ V \rightarrow P\ CanProve\ \psi\ to\ V)$

P2: $V\text{-}is\text{-}external\text{-}party \wedge P\ has\ X \wedge (V\ sees\ X \rightarrow V\ believes\ \phi)$

$\rightarrow P\ CanProve\ \phi\ to\ V$

The axiom **P2** is intended to deal with the provability of an external verifier. It is stated that if V is an external party, P has a message X, and if V receives X, V can infer that the statement ϕ is true, then P can prove to V that ϕ is true. Note that the external party is a party that is not relevant to the particular transaction.

P2': $V\text{-}is\text{-}internal\text{-}party \wedge P\ says\ X' \wedge V\ sees\ X\ \wedge$

$(\ V\ sees\ X \rightarrow (V\ believes\ \phi \wedge V\ has\ X')\ )$

$\rightarrow P\ CanProve\ \phi\ to\ V$

The axiom **P2'** is intended to deal with the provability of an internal verifier. It is stated that if V is an internal party, P sends a message X', V receives X, where X' and X may be different message, and if V receives X, V can infer that the statement ϕ is true and V also has X', then P can prove that ϕ is true to V.

Note that the internal verifier V is a party who is relevant to the transaction e.g. V can be the recipient of the message. Moreover, X may be different from X'. This is because V may be an indirect recipient of the message that P wants to prove. For example, P may send the message X' to another party Q. Then, Q signs X' with her signature to produce the message X. The message X is then sent to V. As V is an internal party, V may have initial information to derive the conclusion.

P3: $P\ \textit{has}\ \{X\}_{K_Q^{-1}} \wedge P\ \textit{has}\ (K_Q, X) \wedge P\ \textit{CanProve}\ (\xrightarrow{K_Q} Q)\ \textit{to}\ V$
$\rightarrow P\ \textit{CanProve}\ (Q\ \textit{says}\ X)\ \textit{to}\ V$

The axiom **P3** states that if P has a message X signed with the private key of Q, P has the message X and Q's public key K_Q, and P can prove to a verifier V that K_Q can be used to refer to Q, then P can prove to V that Q has sent the message X. Note that P possesses X in a way that P decrypts the signed message $\{X\}_{K_Q^{-1}}$ using K_Q to retrieve X. Moreover, P is able to prove that K_Q is referred to Q in a way that K_Q is identical to the public key of Q contained in the digital certificate of Q.

P4: $P\ \textit{CanProve}\ (Q\ \textit{says}\ h(X))\ \textit{to}\ V\ \wedge$
$P\ \textit{CanProve}\ (\ h(X)\textit{-is-fingerprint-of-}X\ )\ \textit{to}\ V$
$\rightarrow P\ \textit{CanProve}\ (Q\ \textit{says}\ X)\ \textit{to}\ V$

The above axiom states that if P can prove to V that Q has sent $h(X)$ and

P can prove to V that $h(X)$ is the fingerprint of X, then P can prove to V that Q has sent X.

P5: P *CanProve* (Q *says* $\{X\}_K$) *to* V $\wedge$
P *CanProve* (K'*-is-decrypting-key-for-*$\{X\}_K$) *to* V
$\rightarrow$ P *CanProve* (Q *says* X) *to* V

The above axiom states that if P can prove to V that Q has sent $\{X\}_K$ and P can prove to V that K' is the decrypting key for $\{X\}_K$, then P can prove to V that Q has sent X. In other words, if P knows the sender of $\{X\}_K$ and P can prove to V that the key K can be used to decrypt $\{X\}_K$, then P can prove to V that Q has sent X.

P6: P *CanProve* (Q *says* $(X_1, ..., X_n)$) *to* V
$\leftrightarrow$ [P *CanProve* (Q *says* X_1) *to* V $\wedge ... \wedge$ P *CanProve* (Q *says* X_n) *to* V]

KP's logic [KP02] was claimed to be able to analyze e-commerce protocols. However, only fixed-network payment protocols e.g. SET [Mas97] and iKP [BGH$^+$00] protocols, have been analyzed by it. In the next section, we will discuss about the possibility and efficiency of KP's logic on analyzing cryptographic protocols designed for wireless networks.

7.3 Accountability vs Protocols for Wireless Networks

Generally, in order to analyze accountability property of a protocol, we consider cryptographic operations applied to messages in the protocol. Digital signature can be used to specify the originator of the message since only its originator can sign the message using her own private key. Public-key encryption can be used to specify the intended recipient of the message because only the intended recipient can decrypt the encrypted message. However, we cannot specify the originator of a symmetric cryptographic message because the encrypting key is shared among engaging parties [Kai96]. For many years, analyzing accountability of e-commerce protocols therefore mainly focuses on asymmetric cryptographic messages.

However, some cryptographic protocols for wireless networks that deploy symmetric cryptography, such as Cimato's protocol [Cim02] and our proposed KSLv1 and KSLv2, offer the ability to identify the originator of a symmetric cryptographic message. This can be done by adding a secret known only to the originator of the message in the message.

Kungpisdan *et al.* argued in [KP02] that KP's logic can be used to deal with complex cryptographic messages. However, KP's logic is not general to analyze e-commerce protocols because it can be used to analyze only the accountability of public-key cryptographic protocols. In particular, there is no axiom reasoning about the accountability of a symmetric cryptographic message. The axiom **P3** can be used only to analyze a signed message. Therefore, we need to provide reasoning about the accountability of symmetric cryptographic messages.

7.4 The Proposed Accountability Logic

To overcome the limitation of KP's logic [KP02] regarding analyzing cryptographic protocols for wireless networks presented in the previous section, we present an extension of KP's logic (called KSL logic) which is able to analyze accountability property of cryptographic protocols that are composed of the combination of asymmetric and symmetric cryptographic messages. As the axiom **P3** for analyzing the accountability property of an asymmetric cryptographic message has been presented in section 7.2, in this section, we introduce axiom **P3'** which is able to deal with the accountability of a symmetric cryptographic message. The details of the axiom **P3'** are given as follows:

P3': If a party P has a message $\langle M_1, \langle M_2\rangle_{K_2}\rangle_{K_1}$ and a key K_1', P believes that K_1' is shared between a party Q and a party P', P can prove to a verifier V that the key K_1' can be used to decrypt $\langle M_1, \langle M_2\rangle_{K_2}\rangle_{K_1}$, and P can prove to V that $\langle M_2\rangle_{K_2}$ is shared between Q and a party R, then P can prove to V that Q has sent M_1 to P'.

$$\begin{array}{l}
P\ has\ (\langle M_1, \langle M_2\rangle_{K_2}\rangle_{K_1}, K_1^{'})\ \wedge \\
P\ believes\ (P' \overset{K_1^{'}}{\longleftrightarrow} Q)\ \wedge \\
P\ CanProve\ (\ K_1^{'}\text{-}is\text{-}decrypting\text{-}key\text{-}for\text{-}\langle M_1, \langle M_2\rangle_{K_2}\rangle_{K_1}\ )\ to\ V\ \wedge \\
P\ CanProve\ (Q \overset{\langle M_2\rangle_{K_2}}{\longleftrightarrow} R)\ to\ V \\
\rightarrow P\ CanProve\ (\ Q\ says\ (M_1, \langle M_2\rangle_{K_2}, ID_{P'})\ )\ to\ V
\end{array}$$

where M_1 and M_2 stands for the message contents. In other words, if P has a message $\langle M_1, \langle M_2\rangle_{K_2}\rangle_{K_1}$ and P is able to decrypt the message using the secret K_1' shared between Q and P', the verifier V is convinced that $\langle M_1, \langle M_2\rangle_{K_2}\rangle_{K_1}$ is originated by either P' or Q. If P can further prove that $\langle M_2\rangle_{K_2}$ is shared between Q and R, V then can infer that P' cannot generate $\langle M_1, \langle M_2\rangle_{K_2}\rangle_{K_1}$ since P' does not initially possess $\langle M_2\rangle_{K_2}$. Note that $\langle M_2\rangle_{K_2}$ is unreadable by P', but derivable by V. Thus, V can conclude that Q, who shares the secret K_1' with P', is the originator of the message $\langle M_1, \langle M_2\rangle_{K_2}\rangle_{K_1}$ and has sent $\langle M_1, \langle M_2\rangle_{K_2}\rangle_{K_1}$ to P' who is the intended recipient of the message. Note also that P may not be the same party as P' in that P may be able to retrieve the secret K_1' which is supposed to be shared only between P' and Q.

We can see that, in any symmetric-encrypted message, if we can prove that one of the two parties who hold the shared secret is the originator of the message, the other party would be the intended recipient of that message. Therefore, the axiom **P3'** not only can capture the accountability of a symmetric cryptographic message, but can also be used to identify the intended recipient of the message.

We further provide reasoning about the intended recipient of a message by considering a message which is encrypted with a public key that can be retrieved only by the party who holds the corresponding private key. We can see that every public-key encrypted message infers its intended recipient. This reasoning can be formalized by axiom **P5'** as follows:

P5': If a party P can prove that a party Q has sent an encrypted message $\{X\}_{K_R}$, a key K_R can be used to refer to a party R, and K_R^{-1} can be used to decrypt $\{X\}_{K_R}$ to a verifier V, then P can prove to V that Q has sent the message X to R.

$$\begin{aligned}
&P\ \mathit{CanProve}\ (Q\ \mathit{says}\ \{X\}_{K_R})\ \mathit{to}\ V \wedge \\
&P\ \mathit{CanProve}\ (\xrightarrow{K_R} R)\ \mathit{to}\ V \wedge \\
&P\ \mathit{CanProve}\ (K_R^{-1}\text{-}\mathit{is\text{-}decrypting\text{-}key\text{-}for}\text{-}\{X\}_{K_R})\ \mathit{to}\ V \\
&\rightarrow P\ \mathit{CanProve}\ (\ Q\ \mathit{says}\ (X, ID_R)\ )\ \mathit{to}\ V
\end{aligned}$$

The axiom **P5'** can be considered as a special case of the axiom **P5** of KP's logic [KP02]. It can be seen that we can identify the intended recipient of a public-key encrypted message from its encrypting public key.

7.5 Analysis of the Proposed SET Framework on Accountability

In this section, we perform the analysis of the SET system based on our framework previously introduced in chapter 4 on accountability property by using the proposed KSL logic. Section 7.5.1 introduces the goals of our analysis. Section 7.5.2 presents initial assumptions for analyzing the proposed system. Section 7.5.3 provides an overview of the analysis. In section 7.5.4, we analyze the proposed SET system on the accountability property.

7.5.1 Goals of the Analysis

To analyze our framework, we focus only on the interaction between the client and the issuer because other parts have been analyzed in [KP02]. The protocol message transmitted between the client and the issuer is shown as follows:

$$\mathbf{C{\rightarrow}I:} \quad \{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$$

The above message infers that the client has authorized the issuer to be the party whom the payment agent PA performs cryptographic operations at the issuer's server. In some cases, a dispute arises if the client claims that she has never requested the issuer to perform the transaction. To resolve such a dispute, the issuer should be able to prove to a verifier that the client has indeed requested it to perform the transaction. Considering the goal of this message, the client must be able to prove to the issuer that she has requested the issuer to do the task. Thus, the message sent from the client to the issuer could be seen as a valid proof.

To analyze the above message, we name the transaction between the client and the issuer as "*Transaction Request*". Based on [Her01] and [KP02], to resolve the dispute between the client and the issuer, we construct the goal of the proof as follows:

G1: I *believes* I *CanProve* (
$$C \textit{ authorized transaction-request}(C, I, Date)\ )\ \textit{to}\ V$$

where I stands for the issuer, C stands for the client, and V stands for an external verifier. The statement above states that the issuer believes that she can prove to the verifier that the client has requested to perform the transaction at the issuer's server.

In order to specify the goal of the transaction between the client and the issuer, the goal of the proof can be shown as follows:

G2: C *believes* C *CanProve* (
$$C \textit{ authorized transaction-request}(C, I, Date)\ )\ \textit{to}\ I$$

The goal states that the client believes that she can prove to the issuer that she has requested to perform the transaction at the issuer's machine.

7.5.2 Initial Assumptions

Based on the initial assumptions of SET protocol [Mas97] presented in [KP02], the initial assumptions which will be presented below are divided into general and protocol-specific assumptions. The general assumptions can be applied to any protocols whereas the protocol-specific ones are applicable only to our SET framework.

General Assumptions

A1: Every party believes that she has the certificates of client, merchant, and payment gateway.

$$P \textit{ believes } P \textit{ has } (Cert_C, Cert_M, Cert_{PG})$$

where P stands for the client, the merchant, or the payment gateway.

A2: Every party believes that if a verifier V has the certificate of a party Q, she will believe that a key K_Q is the public key of Q.

$$P \textit{ believes } (\ V \textit{ has } Cert_Q \rightarrow V \textit{ believes } (\xrightarrow{K_Q} Q)\ )$$

A3: Every party believes that she has only her own private key, and none of the parties believes that she has the private keys of other parties.

$$P \textit{ believes } P \textit{ has } K_P^{-1}, \qquad \neg P \textit{ believes } P \textit{ has } K_Q^{-1}$$

where Q stands for the party that is different from P.

A4: Every party believes that if a verifier V has a message X and a message Y, and X is $h(Y)$, V then believes that X is the fingerprint of Y.

$$P \textit{ believes } [\ (\ V \textit{ has } (X, Y) \wedge X = h(Y)\ ) \\ \rightarrow V \textit{ believes } X\textit{-is-fingerprint-of-}Y\ ]$$

A5: Every party believes that if a verifier V has $\{X\}_K$ and a key K', and the key K' can be used to decrypt the message $\{X\}_K$, V then believes that K' is the decrypting key for $\{X\}_K$.

$$P \textit{ believes } (\ V \textit{ has } (\{X\}_K, K') \wedge X = \{\{X\}_K\}_{K'} \\ \rightarrow V \textit{ believes } K'\textit{-is-decrypting-key-for-}\{X\}_K\ )$$

A6: Every party believes that if a verifier V has a signed message $\{X\}_{K_Q^{-1}}$, a key K_Q, and a message X, and V believes that the key K_Q is referred to the party Q, then V believes that Q has sent the message X.

$$P \textit{ believes } [\ (\ V \textit{ has } \{X\}_{K_Q^{-1}} \wedge V \textit{ has } (K_Q, X) \wedge V \textit{ believes } (\xrightarrow{K_Q} Q)\ ) \\ \rightarrow V \textit{ believes } (Q \textit{ says } X)\ ]$$

Protocol-specific Assumptions

No Disclosure of Private Information to Verifier

A7: Every party does not believe that a verifier V has payment information, the private keys of other parties, and order descriptions.

$\neg P$ *believes V has PI*,	where V is not the same party as PG.
$\neg P$ *believes V has* K_P^{-1},	where V is not the same party as P.
$\neg P$ *believes V has OD*	

Payment Information

A8: The client and the merchant believe that they have order information, order descriptions and price, and all parties believe that they are internal parties.

Q *believes* Q *has* $(OI, OD, Price)$,

R *believes* R*'-is-internal-party*

where Q stands for the client and the merchant, and R and R' stand for the client, the issuer, the merchant, and the payment gateway.

Sending and Receiving Messages

A9: Every party believes that $\{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$ has been sent by the client and received by the issuer.

$$\begin{array}{l} P \textit{ believes } C \textit{ says } \{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}, \\ P \textit{ believes } I \textit{ sees } \{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}} \end{array}$$

where P stands for the client, the issuer, the merchant, and the payment gateway.

Shared Secrets

A10: Both client and issuer believe that they possess the shared key K_{ex}, and both of them believe that K_{ex} is shared between them.

$$Q \textit{ believes } Q \textit{ has } K_{ex}, \qquad Q \textit{ believes } (C \overset{K_{ex}}{\longleftrightarrow} I)$$

where Q stands for the client or the issuer.

Payment Authorizations

A11: Every party believes that she can prove to a verifier V that if the client has sent the message containing the issuer's identity and the date of transaction, then the client has authorization on transaction request to the issuer.

$$\begin{array}{l} P \textit{ believes } P \textit{ CanProve } [\ C \textit{ says } (ID_I, Date) \\ \qquad\qquad\qquad C \textit{ authorized transaction-request}(C, I, Date)\ ] \textit{ to } V \end{array}$$

7.5.3 Outline of the Proof

In order to prove the goals of the analysis outlined in section 7.5.1, we transform all protocol messages into the form of our logic. For example, the following protocol step:

$$\mathbf{C}{\rightarrow}\mathbf{M}: \quad \{X\}_K \qquad \text{can be transformed into} \qquad M\ sees\ \{X\}_K$$

We then construct the receiver's belief, followed by the possession, of the received message by using the axioms **C1** and **H1**, respectively. Then we can derive the following statement:

$$M\ believes\ M\ has\ \{X\}_K$$

If K is the C's private key, we can apply the axiom **P3**. If K is the symmetric key shared between C and M, we apply the axiom **P3'** instead. For the latter case, if the condition in the axiom **P3'** that X cannot be generated by M holds, we can derive the following statement:

$$M\ believes\ M\ CanProve\ (C\ says\ (X, ID_M))\ to\ V$$

If X contains *Price* and *Date*, by applying the axiom **K** and the assumption **A12**, we can achieve the goal **G1**:

$$M\ believes\ M\ CanProve\ (\\ C\ authorized\ payment\text{-}order(C, M, Price, Date))\ to\ V$$

7.5.4 Analyzing the Proposed SET Framework on Accountability

Guideline of the Proof

We show how to read the proof statement in order to help the reader understand the proof. For example,

1, C1: *C believes C sees* $\{X\}_K$ *(2)*

This means that this is the statement *(2)* which is derived from applying the statement *(1)* with the axiom **C1**.

Proving the Goal G1

G1: *I believes I CanProve (*
C authorized transaction-request$(C, I, Date)$ *to V*

Consider **Step 2** of the proposed SET framework,

C→I: $\{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$

It can be transformed into:

I sees $\{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$

Details of the Proof

I sees $\{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$ *(1)*

1, C1: *I believes I sees* $\{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$ *(2)*

2, H1, M: *I believes I has* $\{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$ *(3)*

3, A10, H4, M:

I *believes* I *has* $(PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}})$ *(4)*

4, H2, M: I *believes* I *has* $\{ID_I, h(PReqGen)\}_{K_C^{-1}}$ *(5)*

A1, H2, M: I *believes* I *has* K_C *(6)*

A1, A2, K: I *believes* I *believes* $(\xrightarrow{K_C} C)$ *(7)*

5, 6, 7, H6, M: I *believes* I *has* $(ID_I, h(PReqGen))$ *(8)*

A1, A2, P2, M: I *believes* I *CanProve* $(\xrightarrow{K_C} C)$ *to* V *(9)*

5, 6, 8, 9, P3, M:

I *believes* I *CanProve* (C *says* $(ID_I, h(PReqGen))$) *to* V *(10)*

10, P6, M:

I *believes* I *CanProve* (C *says* $h(PReqGen)$) *to* V *(11)*

3, 8, H2, M: I *believes* I *has* $(PReqGen, h(PReqGen))$ *(12)*

12, A4, P2, M:

I *believes* I *CanProve* (

$h(PReqGen)$*-is-fingerprint-of-*$PReqGen$) *to* V *(13)*

11, 13, P4, M: I *believes* I *CanProve* (C *says PReqGen*) *to* V *(14)*

14, P6, M: I *believes* I *CanProve* (C *says Date*) *to* V *(15)*

10, 15, P6, M:

I *believes* I *CanProve* (C *says* $(ID_I, Date)$) *to* V *(16)*

16, A11, K:

I *believes* I *CanProve* (

C *authorized transaction-request*$(C, I, Date)$) *to* V *(17)*

The proof of the goal **G1** is successful.

Proving the Goal G2

G2: C *believes* C *CanProve* (

C *authorized transaction-request*$(C, I, Date)$) *to* I

Consider **Step 2** of the proposed SET framework,

C→I: $\{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$

It can be transformed into:

C *says* $\{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$

Details of the Proof

C *says* $\{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$ *(1)*

1, C2, M:

C *believes* C *says* $\{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$ *(2)*

2, SA2, M:

C *believes* C *has* $\{PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}}\}_{K_{ex}}$ *(3)*

3, A10, H4, M:

C *believes* C *has* $(PReqGen, \{ID_I, h(PReqGen)\}_{K_C^{-1}})$ *(4)*

4, H2, M: C *believes* C *has* $\{ID_I, h(PReqGen)\}_{K_C^{-1}}$ *(5)*

A1, H2, M: C *believes* C *has* K_C *(6)*

A1, A2, K: C *believes* C *believes* $(\xrightarrow{K_C} C)$ *(7)*

5, 6, 7, H6, M: C *believes* C *has* $(ID_I, h(PReqGen))$ *(8)*

A1, A2, P2, M: C *believes* C *CanProve* $(\xrightarrow{K_C} C)$ *to* I *(9)*

5, 6, 8, 9, P3, M:

C *believes* C *CanProve* $(C$ *says* $(ID_I, h(PReqGen)))$ *to* I *(10)*

10, P6, M: C *believes* C *CanProve* $(C$ *says* $h(PReqGen))$ *to* I *(11)*

A8, A9, A4, P2', M:

C *believes* C *CanProve* (

$h(PReqGen)$*-is-fingerprint-of-PReqGen*) *to* I *(12)*

11, 12, P4, M: C *believes* C *CanProve* $(C$ *says* $PReqGen)$ *to* I *(13)*

13, P6, M: C *believes* C *CanProve* $(C$ *says Date*$)$ *to* I *(14)*

10, 14, P6, M:

$$C\ believes\ C\ CanProve\ (C\ says\ (ID_I, Date))\ to\ I \qquad (15)$$

15, A11, K:

$$C\ believes\ C\ CanProve\ (\\ C\ authorized\ transaction\text{-}request(C, I, Date))\ to\ I \qquad (16)$$

The proof of the goal **G2** is successful. The results of the analyses show that the client can prove her request to the verifier and the issuer can infer the client's request with all necessary information. As a result, the SET system based on the proposed framework can provide the accountability of the extension of original SET protocol. More generally, the entire system satisfies the accountability property.

As mentioned in sections 3.1.7-3.1.9 that $Goals$, PR, and $TSec$ can be formalized into the proof statements of KP's logic [KP02], the successful proof on accountability of this system by using our KSL logic, a variant of KP's logic, infers that the proposed system satisfies such properties. Combining with the discussions previously presented in chapter 4 that the proposed SET system satisfies $Trust$ and ATP, the SET system based on the proposed framework therefore is considered as a practical and secure mobile payment system according to the Definition 3.15.

7.6 Analysis of KSLv1 and KSLv2 Protocols on Accountability

In this section, we analyze KSLv1 and KSLv2 on accountability property by using KSL logic. We show that KSL logic has practical usefulness; it is more general than existing logics [Kai96, KN98, KP02] in that it can be used to analyze the protocols based on both asymmetric and/or symmetric cryptographic messages efficiently. Section 7.6.1 introduces initial assumptions for the analysis of KSLv1 and KSLv2. Section 7.6.2 presents goals of the analysis. In sections 7.6.3 and 7.6.4, we analyze KSLv1 and KSLv2 on the accountability.

7.6.1 Initial Assumptions

The initial assumptions presented below can be used for both KSLv1 and KSLv2 protocols. They are divided into general and protocol-specific assumptions. The general assumptions can be applied to any protocols, whereas the protocol specific ones are applicable only to KSLv1 and KSLv2.

General Assumptions

A1: Every party believes that if a verifier V believes that a key K' is shared between a party Q and a party R, V has both the message $\langle X\rangle_K$ and K', and K' can be used to extract X from $\langle X\rangle_K$, then V believes that $\langle X\rangle_K$ is shared between Q and R.

$$\begin{aligned} &P \textit{ believes } [\ (V \textit{ believes } (Q \overset{K'}{\longleftrightarrow} R) \wedge\ V \textit{ has } (\langle X\rangle_K, K') \wedge\ X = \langle\langle X\rangle_K\rangle_{K'}) \\ &\qquad \to\ V \textit{ believes } (Q \overset{\langle X\rangle_K}{\longleftrightarrow} R)\ ] \end{aligned}$$

where Q and R stand for any two participants. The assumption **A1**, together with the axiom **P2**, can be used to analyze the provability of single-key cryptographic messages.

A2: Every party believes that if a verifier V believes that she does not generate the message $\langle X\rangle_K$ by herself, a key K' is shared between a party P' and a party Q, V has both $\langle X\rangle_K$ and K', and K' can be used to extract X from $\langle X\rangle_K$, then V believes that Q has sent X to P'.

$$\begin{aligned} &P \textit{ believes } (\ V \textit{ believes } P\text{'} \textit{ sees } \langle X\rangle_K \wedge \\ &\qquad V \textit{ believes } (P' \overset{K'}{\longleftrightarrow} Q) \wedge V \textit{ has } (\langle X\rangle_K, K') \wedge X = \langle\langle X\rangle_K\rangle_{K'} \\ &\qquad \rightarrow V \textit{ believes } Q \textit{ says } (X, ID_{P'})\ ) \end{aligned}$$

A3: Every party believes that if a verifier V has the public-key certificate of a party Q, she then believes that Q's public key K_Q can refer to Q.

$$P \textit{ believes } (\ V \textit{ has } Cert_Q \rightarrow V \textit{ believes } (\xrightarrow{K_Q} Q)\ )$$

A4: Every party believes that, if a verifier V has both X and Y, and X is $h(Y)$, V then believes that X is the fingerprint of Y.

$$\begin{aligned} &P \textit{ believes } [\ (V \textit{ has } (X, Y) \wedge X = h(Y)) \\ &\qquad \rightarrow V \textit{ believes } X\textit{-is-fingerprint-of-}Y\ ] \end{aligned}$$

A5: Every party believes that if a verifier V has the message $\{X\}_K$ and a key K', and the key K' can be used to decrypt $\{X\}_K$, V then believes that K' is the decrypting key for $\{X\}_K$.

$$\begin{aligned} &P \textit{ believes } (\ V \textit{ has } (\{X\}_K, K') \wedge X = \{\{X\}_K\}_{K'} \\ &\qquad \rightarrow V \textit{ believes } K\text{'}\textit{-is-decrypting-key-for-}\{X\}_K\ ) \end{aligned}$$

Protocol Specific Assumptions

The following assumptions are specific to KSLv1 and KSLv2.

Possessions

A6: Every party has its own identities.

$$P \textit{ believes } P \textit{ has } ID_P$$

Moreover, the client believes that she has the issuer's identity (ID_I). The merchant believes that she has the payment gateway's identity (ID_{PG}). The payment gateway believes that it has the merchant's identity (ID_M). The issuer believes that it has the client's identity (ID_C).

C *believes* C *has* ID_I, $\quad$ M *believes* M *has* ID_{PG},
PG *believes* PG *has* ID_M, $\quad$ I *believes* I *has* ID_C

Note that the client has ID_I because the issuer issues her a credit card. Therefore, the issuer also has ID_C. The merchant has ID_{PG} since she registers to the payment gateway to be a merchant in the payment system. Thus, the payment gateway also has ID_M.

A7: Only in KSLv1, the merchant and the payment gateway believe that they have their own certificates and each others. Also, they believe that they have only their own private keys.

Q *believes* Q *has* $(Cert_M, Cert_{PG})$,
Q *believes* Q *has* K_Q^{-1},
$\neg Q$ *believes* Q *has* K_R^{-1}

where Q stands for the merchant or the payment gateway. $Cert_M$ and $Cert_{PG}$ contain $\{K_M, ID_M\}$ and $\{K_{PG}, ID_{PG}\}$, respectively. K_R^{-1} stands for the private key of the party which is different from Q.

Shared Secrets

A8: The client and the merchant believe that X_i and $\langle N\rangle_{X_i}$ are shared between them, and they also have X_i. The client and the issuer believe that Y_i and $\langle N\rangle_{Y_i}$ are shared between them, and they also have Y_i.

P *believes* $(C \xleftrightarrow{X_i} M)$, $\quad P$ *believes* $(C \xleftrightarrow{\langle N\rangle_{X_i}} M)$,

Q *believes* $(C \xleftrightarrow{Y_i} I)$, $\quad Q$ *believes* $(C \xleftrightarrow{\langle N\rangle_{Y_i}} I)$,

P *believes* P *has* X_i , $\quad Q$ *believes* Q *has* Y_i

Only in KSLv2, the merchant and the payment gateway believe that Z_j and $\langle N\rangle_{Z_j}$ are shared between them, and they also have Z_j.

R *believes* $(M \xleftrightarrow{Z_j} PG)$, $\quad R$ *believes* $(M \xleftrightarrow{\langle N\rangle_{Z_j}} PG)$,

R *believes* R *has* Z_j

where P stands for the client and the merchant, Q stands for the client and the issuer, and R stands for the merchant and the payment gateway.

A9: Every party believes that the merchant does not receive Y_i and the issuer does not receive X_i.

P *believes* $\neg M$ *sees* Y_i, $\quad P$ *believes* $\neg I$ *sees* X_i

Only in KSLv2, the client does not receive Z_j.

P *believes* $\neg C$ *sees* Z_j

Verifier's Beliefs

A10: Every party believes that V is an external verifier to which she is able to reveal her own secret information.

P *believes V-is-external-party*

Every party believes that V believes that only X_i and $\langle N\rangle_{X_i}$ are shared between the client and the merchant, and only Y_i and $\langle N\rangle_{Y_i}$ are shared between the client and her issuer.

$$\begin{array}{ll} P \textit{ believes } V \textit{ believes } (C \xleftrightarrow{X_i} M), & P \textit{ believes } V \textit{ believes } (C \xleftrightarrow{\langle N\rangle_{X_i}} M), \\ P \textit{ believes } V \textit{ believes } (C \xleftrightarrow{Y_i} I), & P \textit{ believes } V \textit{ believes } (C \xleftrightarrow{\langle N\rangle_{Y_i}} I), \\ P \textit{ believes } \neg V \textit{ believes } (C \xleftrightarrow{T_1} M), & P \textit{ believes } \neg V \textit{ believes } (C \xleftrightarrow{T_2} I) \end{array}$$

Only in KSLv2, Z_j and $\langle N\rangle_{Z_j}$ are shared between the merchant and the payment gateway.

$$\begin{array}{l} Q \textit{ believes } V \textit{ believes } (M \xleftrightarrow{Z_j} PG), \\ Q \textit{ believes } V \textit{ believes } (M \xleftrightarrow{\langle N\rangle_{Z_j}} PG), \\ Q \textit{ believes } \neg V \textit{ believes } (M \xleftrightarrow{T_3} PG) \end{array}$$

where,

- P stands for every party,
- N stands for a message,
- T_1 stands for the message which is different from X_i and $\langle N\rangle_{X_i}$.,
- T_2 stands for the message which is different from Y_i and $\langle N\rangle_{Y_i}$, and
- T_3 stands for the message which is different from Z_j and $\langle N\rangle_{Z_j}$.

Moreover, in both KSLv1 and KSLv2, every party believes that the verifier believes that the issuer does not generate any messages by itself. In other words, the issuer only receives messages from other parties.

P *believes* V *believes* I *sees* X, where X stands for any message.

Payment Information

A11: The client and the merchant believe that they possess order information, price, and goods descriptions.

Q *believes* Q *has* $(OI, Price, OD)$

where Q stands for the client or the merchant. Note that the merchant has OI because she generates TID by herself, and she has the information about goods, such as the descriptions and price.

Payment Authorizations

A12: Every party believes that she can prove to a verifier that, if the client has sent a message containing the merchant's identity, price, and the date of transaction, then the client has authorization on making a payment to the merchant.

P *believes* P *CanProve* (C *says* $(ID_M, Price, Date)$
$\rightarrow$ C *authorized payment-order*$(C, M, Price, Date)$) *to* V

Every party believes that she can prove to a verifier that, if the merchant has sent a message containing the client's identity, price, and the date of transaction, then the merchant has authorization on issuing a payment receipt to the client.

P *believes* P *CanProve* (M *says* $(ID_C, Price, Date)$
$\rightarrow$ M *authorized payment-order*$(M, C, Price, Date)$) *to* V

Every party believes that she can prove to a verifier that, if the issuer has sent a message containing the client's identity, price, and the date of transaction, then the issuer has authorization on *Debit* from the client's account.

$$\begin{aligned} &P \textit{ believes } P \textit{ CanProve } (\ I \textit{ says } (ID_C, Price, Date) \\ &\qquad\qquad \rightarrow I \textit{ authorized debit}(I, C, Price, Date)\ ) \textit{ to } V \end{aligned}$$

Every party believes that she can prove to a verifier that, if the client has sent a message containing the issuer's identity, price, and the date of transaction, then the client has authorization on requesting the issuer to deduct the requested amount of money from her account.

$$\begin{aligned} &P \textit{ believes } P \textit{ CanProve } (\ C \textit{ says } (ID_I, Price, Date) \\ &\qquad\qquad \rightarrow C \textit{ authorized debit}(C, I, Price, Date)\ ) \textit{ to } V \end{aligned}$$

Every party believes that she can prove to a verifier that, if the payment gateway has sent a message containing the merchant's identity, price, and the date of transaction, then the payment gateway has authorization on transferring the requested amount of money to the merchant's account.

$$\begin{aligned} &P \textit{ believes } P \textit{ CanProve } (\ PG \textit{ says } (ID_M, Price, Date) \\ &\qquad\qquad \rightarrow PG \textit{ authorized credit}(PG, M, Price, Date)\ ) \textit{ to } V \end{aligned}$$

Every party believes that she can prove to a verifier that, if the merchant has sent a message containing the payment gateway's identity, price, and the date of transaction, then the merchant has authorization on requesting the payment gateway to transfer the requested amount of money to the merchant's account.

$$P \textit{ believes } P \textit{ CanProve } (\ M \textit{ says } (ID_{PG}, Price, Date) \\ \rightarrow M \textit{ authorized credit}(M, PG, Price, Date)\ ) \textit{ to } V$$

7.6.2 Goals of the Analysis

In order to analyze accountability property of KSLv1 and KSLv2 protocols, the following statements need to be proved [KP02].

G1: $M \textit{ believes } M \textit{ CanProve } ($
$$C \textit{ authorized payment-order}(C, M, Price, Date)) \textit{ to } V$$

The merchant M believes that she can prove to a verifier V that the client C has sent an authorized message on making the payment amount $Price$ to the merchant.

G2: $C \textit{ believes } C \textit{ CanProve } ($
$$M \textit{ authorized payment-order}(M, C, Price, Date)) \textit{ to } V$$

The client C believes that she can prove to a verifier V that the merchant M has sent an authorized message on acknowledgement of the payment amount $Price$ to the client.

G3: $I \textit{ believes } I \textit{ CanProve } ($
$$C \textit{ authorized debit}(C, I, Price, Date)) \textit{ to } V$$

The issuer I believes that it can prove to a verifier V that the client C has sent an authorized message on requesting it to deduct the amount $Price$ from the client's account.

G4: C *believes* C *CanProve* (

$$I \textit{ authorized } \textit{debit}(I, C, Price, Date)) \textit{ to } V$$

The client C believes that she can prove to a verifier V that the issuer I has sent an authorized message on acknowledgement of requesting to deduct the amount $Price$ from the client's account.

G5: PG *believes* PG *CanProve* (

$$M \textit{ authorized } \textit{credit}(M, PG, Price, Date)) \textit{ to } V$$

The payment gateway PG believes that it can prove to a verifier V that the merchant M has sent an authorized message on requesting to transfer the amount $Price$ to the merchant's account.

G6: M *believes* M *CanProve* (

$$PG \textit{ authorized } \textit{credit}(PG, M, Price, Date)) \textit{ to } V$$

The merchant M believes that she can prove to a verifier V that the payment gateway PG has sent an authorized message on acknowledgement of transferring the amount $Price$ to the merchant's account.

7.6.3 Analyzing KSLv1 on Accountability

In this section, we show that the proposed KSL logic is able to reason about both symmetric and asymmetric cryptography by analyzing KSLv1 protocol which exploits both kinds of cryptographic operations.

Proving the Goal G1 of KSLv1

G1: *M believes M CanProve (*
C authorized payment-order$(C, M, Price, Date)$*) to V*

Consider **Step 2** of KSLv1,

C→M: $\{OI, Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i}$,
$MAC[(OI, Price, ID_C, ID_I), X_{i+1}]$

It can be transformed into:

M sees $\{OI, Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i}$,
$MAC[(OI, Price, ID_C, ID_I), X_{i+1}]$

Details of the Proof

M sees $\{OI, Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i}$,
$MAC[(OI, Price, ID_C, ID_I), X_{i+1}]$ *(1)*

1, C1, H1, H2, M:
M believes M has
$\{OI, Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i}$ *(2)*

2, A8, H2, M: *M believes M has (*
$\{OI, Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i}, X_i$ *)* *(3)*

A10, 3, A5, P2, M:
M believes M CanProve (X_i*-is-decrypting-key-for-*
$\{OI, Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i}$*) to V* *(4)*

3, A8, H4, M:
M believes M has (
$OI, Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]$ *)* *(5)*

5, H2, M: *M believes M has* $MAC[(Price, h(OI), ID_M), Y_i]$ *(6)*

4, H3, M: *M believes M has* $(Price, h(OI))$ *(7)*

7, A6, H2, M: *M believes M has* $(Price, h(OI), ID_M)$ *(8)*

A10, 6, 8, H2, A1, P2, M:

M believes M CanProve $(C \overset{MAC[(Price,h(OI),ID_M),Y_i]}{\longleftrightarrow} I)$ *to V* *(9)*

3, A8, 4, 9, P3', M:

M believes M CanProve (

C says $(OI, Price, ID_C, ID_I,$

$MAC[(Price, h(OI), ID_M), Y_i], ID_M)$) to V *(10)*

10, P6, M:

M believes M CanProve (*C says* $(Price, ID_M, Date)$) *to V* *(11)*

11, A12, M:

M believes M CanProve (

C authorized payment-order$(C, M, Price, Date)$) *to V* *(12)*

The goal **G1** is successfully proved. However, KSLv1 fails to prove the goal **G2** because the client is not able to identify the originator of the message sent to her. Consider **Step 6** of KSLv1,

$$\mathbf{M}\rightarrow\mathbf{C}: \quad \{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$$

To achieve the goal **G2**, the client needs to prove '*C believes C CanProve* (*M says* $(h(OI), Yes/No, ID_C)$) *to V*' mainly by using the axiom **P3'**. To prove by using the axiom **P3'**, the statement '*C believes C CanProve* $(M \overset{\{(h(OI),Yes/No)\}_{Y_i}}{\longleftrightarrow} I)$ *to V*' must hold. However, this proof fails since it contradicts the assumption **A10** in that the verifier does not believe that the merchant has Y_i. The details of the proof are shown as follows:

Proving the Goal G2 of KSLv1

G2: *C believes C CanProve (*

M authorized payment-order$(M, C, Price, Date)$ *) to V*

Consider **Step 6** of KSLv1,

M→C: $\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$

It can be transformed into:

C sees $\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$

Details of the Proof

C sees $\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$ *(1)*

1, C1, M: *C believes C sees* $\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$ *(2)*

2, H1, M: *C believes C has* $\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$ *(3)*

3, A8, H2, M: *C believes C has* $(\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}, X_{i+1})$ *(4)*

A10, 4, A5, P2, M:

C believes C CanProve (

X_{i+1}*-is-decrypting-key-for-*$\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$ *) to V* *(5)*

3, A8, H2, H4, M: *C believes C has* $\{h(OI), Yes/No\}_{Y_i}$ *(6)*

6, A8, H2, M: *C believes C has (* $\{h(OI), Yes/No\}_{Y_i}, Y_i$ *)* *(7)*

A10, 7, A1, P2, M:

C believes V-is-external-party $\wedge$

C believes C has $(h(OI), Yes/No)$ $\wedge$

C believes (

V believes $(\boldsymbol{M} \overset{\{(h(OI), Yes/No)\}_{Y_i}}{\longleftrightarrow} \boldsymbol{I}) \wedge$ −−−−−−−−−−−−−−−−−***(a)***

V has $(h(OI), Yes/No)$

$\rightarrow$ *V believes* $(M \overset{\{(h(OI), Yes/No)\}_{Y_i}}{\longleftrightarrow} I)$

$\rightarrow$ *C believes C CanProve* $(M \overset{\{(h(OI), Yes/No)\}_{Y_i}}{\longleftrightarrow} I)$ *to V* *(8)*

4, A8, 5, 8, P3', M:

C believes C has $(\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}, X_{i+1}) \wedge$

C believes C believes $(C \overset{X_{i+1}}{\longleftrightarrow} M) \wedge$

C believes C CanProve (

X_{i+1}*-is-decrypting-key-for-*$\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$) *to V* $\wedge$

C believes C CanProve ($\boldsymbol{M \overset{\{\{h(OI), Yes/No\}_{Y_i}}{\longleftrightarrow} I}$) ***to V −− from (8)***

$\rightarrow$ *C believes C CanProve* (

M says $(\{h(OI), Yes/No\}_{Y_i}, ID_C)$) *to V* *(9)*

We can see that, to achieve the statement *(9)*, the statement *(8)*, which is one of its conditions, must hold. However, the statement *(a)* contradicts the assumption **A10** in that the verifier does not believe that $\{h(OI), Yes/No\}_{Y_i}$ is shared between the merchant and the issuer. This is because the merchant does not have Y_i. Thus, the proof fails. The kind of contradiction also leads to the failure of proving the goal **G4**.

From **Step 6** of KSLv1, the client is able to generate the payment receipt by herself and make a claim to the merchant. However, KSLv1 can still provide the dispute resolution property. We can see that, although the client can generate the protocol **Step 6** by herself, she cannot claim this fake payment receipt to other parties since the keys X_i and Y_i used for construct the fake **Step 6** have not yet been updated in the revocation lists stored at the merchant and the issuer. In addition, even though the client reuses the keys which have already updated at the merchant and the issuer and generates the fake **Step 6**, for example, by changing the amount or goods descriptions, her attempt will not be successful since those keys have already been bound with $h(OI)$ of the previous transaction recorded in the revocation list stored at the merchant and the issuer. As a result, KSLv1 has indirect proofs of accountability regarding to the goals **G2** and **G4**. The details of proving the goal G4 will be presented in appendix A.1.1.

Proving the Goal G3 of KSLv1

G3: *I believes I CanProve (*

C authorized debit$(C, I, Price, Date)$ *) to V*

Consider **Step 4.1** of KSLv1,

PG→I: $MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price,$
ID_C, ID_M

It can be transformed into:

I sees $(MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price, ID_C, ID_M)$

Details of the Proof

I sees $(MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price, ID_C, ID_M)$ *(1)*

1, C1, M: *I believes I sees* $(MAC[(Price, h(OI), ID_M), Y_i], h(OI), i,$
$TID, Price, ID_C, ID_M)$ *(2)*

2, H1, H2, M: *I believes I has* $(MAC[(Price, h(OI), ID_M), Y_i],$
$h(OI), i, TID, Price, ID_C, ID_M)$ *(3)*

3, A8, H2, M: *I believes I has* $(MAC[(Price, h(OI), ID_M), Y_i],$
$h(OI), i, TID, Price, ID_C, ID_M, Y_i)$ *(4)*

A10, 4, A2, P2, M:

I believes V-is-external-party $\wedge$

I believes I has (

$MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price, ID_C, ID_M, Y_i)$ $\wedge$

I believes (

V believes $(C \xleftrightarrow{Y_i} I)$ $\wedge$

V *believes* I *sees* $MAC[(Price, h(OI), ID_M), Y_i] \wedge$

V *has* $(MAC[(Price, h(OI), ID_M), Y_i], Price, h(OI), ID_M, Y_i) \wedge$

$MAC[(Price, h(OI), ID_M), Y_i] = MAC[(Price, h(OI), ID_M), Y_i]$

$\rightarrow$ V *believes* C *says* $(Price, h(OI), ID_M, ID_I)$

$\rightarrow$ I *believes* I *CanProve* (C *says* $(Price, h(OI), ID_M, ID_I)$) *to* V *(5)*

5, P6, M: I *believes* I *CanProve* (C *says* $(Price, ID_I)$) *to* V *(6)*

The issuer cannot prove that the client has sent the date of transaction *Date* to her. However, *Date* has already been contained in the message sent to the issuer under the banking private network. Thus, it does not compromise the security of the system.

The details of the analysis of the goals **G4-G6** will be shown in appendix A.1. The successful proofs of accountability of KSLv1 protocol infer that a payment system that deploys KSLv1 protocol will satisfy not only goals (*Goals*) and requirements for payment transactions (PR), but also transaction security properties ($TSec$) stated in the formal model presented in chapter 3.

7.6.4 Analyzing KSLv2 on Accountability

In this section, we show how to analyze KSLv2 protocol on accountability property. The goals of KSLv2 protocol can be analyzed as follows.

Proving the Goal G1 of KSLv2

G1: M *believes* M *CanProve* (

C *authorized payment-order*$(C, M, Price, Date)$) *to* V

Consider **Step 2** of KSLv2,

C→M: $\{OI, h(Y_i), Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i}$,

$MAC[(OI, Price, ID_C, ID_I), X_{i+1}]$

It can be transformed into:

M sees $\{OI, h(Y_i), Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i},$
$MAC[(OI, Price, ID_C, ID_I), X_{i+1}]$

Details of the Proof

M sees $\{OI, h(Y_i), Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i},$
$MAC[(OI, Price, ID_C, ID_I), X_{i+1}]$ *(1)*

1, C1, H1, H2, M:

M believes M has
$\{OI, h(Y_i), Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i}$ *(2)*

2, A8, H2, M: *M believes M has (*
$\{OI, h(Y_i), Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i}, X_i$ *)* *(3)*

A10, 3, A5, P2, M:

M believes M CanProve (X_i*-is-decrypting-key-for-*
$\{OI, h(Y_i), Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i}$*) to V* *(4)*

3, A8, H4, M:

M believes M has (
$OI, h(Y_i), Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]$ *)* *(5)*

5, H2, M: *M believes M has* $MAC[(Price, h(OI), ID_M), Y_i]$ *(6)*

4, H3, M: *M believes M has* $(Price, h(OI))$ *(7)*

7, A6, H2, M: *M believes M has* $(Price, h(OI), ID_M)$ *(8)*

A10, 6, 8, H2, A1, P2, M:

M believes M CanProve ($C \overset{MAC[(Price, h(OI), ID_M), Y_i]}{\longleftrightarrow} I$*) to V* *(9)*

3, A8, 4, 9, P3', M:

M believes M CanProve (

C says ($OI, Price, ID_C, ID_I,$

$$MAC[(Price, h(OI), ID_M), Y_i], ID_M)\ )\ \text{to V} \qquad \textit{(10)}$$

10, P6, M:

M believes M CanProve (*C says* $(Price, ID_M, Date)$) *to V* *(11)*

11, A12, M:

M believes M CanProve (

C authorized payment-order$(C, M, Price, Date)$) *to V* *(12)*

The goal **G1** is successfully proved. Note that the details of the analysis of the goals **G2-G6** will be presented in appendix A.1. The results from the analysis show that KSLv2 protocol satisfies accountability property. Therefore, this infers that a payment system that deploys KSLv2 protocol will satisfy $Goals$, PR, and $TSec$ stated in the proposed mobile payment model.

As a result, combining the from the discussions and analyses previously presented in chapter 5, it can be concluded that the payment system based on KSLv1 or KSLv2 is considered as a practical and secure mobile payment system stated in the Definition 3.15 of the proposed model.

7.7 Party's Requirements for Payment Transactions as Accountability

In this section, we show that party's requirements for payment transactions of iKP protocol [BGH$^+$00] can be reasoned by accountability and can be formalized into the party's requirements for payment transactions (PR) of the proposed mobile payment model. In section 7.7.1, the party's requirements for payment transactions of iKP protocol are outlined. Section 7.7.2 presents how they can be reasoned and formalized by the proposed model.

7.7.1 Party's Requirements for Payment Transactions of iKP Protocol

Bellare *et al.* [BGH$^+$00] argued that iKP protocol satisfies the following party's requirements for payment transactions:

Issuer and Acquirer's Requirements

IA1: *Proof of Transaction Authorization by Client*: it is the unforgeable proof that the client has authorized the issuer and the acquirer to deduct the money from her account.

IA2: *Proof of Transaction Authorization by Merchant*: the payment gateway must be able to prove that the merchant has requested it to transfer the money to her account.

Merchant's Requirements

S1: *Proof of Transaction Authorization by Payment Gateway*: the merchant must be able to prove that the payment gateway has authorized the payment to her.

S2: *Proof of Transaction Authorization by Client*: the merchant must be able to prove that the client has requested to make the payment to her.

Client's Requirements

B1: *Impossibility of Unauthorized Payment*: the client must not be charged on the payment she has never made. This requirement is composed of *Impossibility* and *Disputability*. The impossibility property states that, if the payment gateway is honest, it is impossible for any unauthorized payment to be made. The disputability property states that, even though the payment gateway's secret is available to adversary, the client can still prove that she has not authorized the payment.

B2: *Proof of Transaction Authorization by Payment Gateway*: the client must be able to prove that the payment gateway has authorized the transaction relevant to her.
B3: *Certification and Authorization of Merchant*: the client can guarantee that she makes the payment to the valid merchant.
B4: *Receipt from Merchant*: the client must be able to prove that the merchant, whom she has requested to make the payment to, has received the payment and committed to deliver the goods to her.

7.7.2 Formalizing Party's Requirements Using the Proposed Logic

In chapter 3, we have shown that the party's requirements for payment transactions presented in the previous section are relevant to the ability to prove the authorization of transactions by particular parties in each party's point of view. In this section, we show that the party's requirements can be formalized by the proposed logic and modelled into the proposed mobile payment model.

The issuer and acquirer's requirement **IA1** and the client's requirement **B1** can be formalized by the following statement:

IA1, B1: $I \textit{ believes } I \textit{ CanProve } ($
$$C \textit{ authorized } \textit{debit}(C, I, Price, Date)) \textit{ to } V$$

According to the requirement **IA1**, the issuer is able to prove to the verifier that the client has authorized it to deduct the money from the client's account. According to the requirement **B1**, the issuer can use this proof to show that the client has really requested it to charge the money from the client's account, and the issuer will deduct the money if it receives the request from only the client.

Referred to the party's requirements of the proposed mobile payment model in section 3.1.8, the above statement is considered as one the the requirements to complete a transaction in the payment system provider (PSP)'s point of view in that, before charging the money from the client's account, PSP must ensure that the client has really requested it to perform such a transaction by considering the message originated by the client sent to it as a *Debit* request.

The issuer and acquirer's requirement **IA2** can be formalized by the following statement:

IA2: $\textit{PG believes PG CanProve}\ ($
$$\textit{M authorized credit}(M, PG, Price, Date))\ \textit{to}\ V$$

With this proof, the payment gateway can ensure that the merchant has requested it to transfer the requested money to the merchant's account. Referred to the proposed mobile payment model, the above statement is considered as one of the requirements to complete a transaction in the payment system provider (PSP)'s point of view in that PSP must ensure that it receives an authorized message as a request from the merchant to transfer money to the merchant's account.

It is not hard to see that both the requirements **IA1** and **IA2** of iKP protocol can be formalized as the conditions to meet PSP's requirements of the proposed mobile payment model presented in section 3.1.8. However, the requirement **B1** of iKP protocol is also considered as one of the PSP's requirements of the proposed model. This is because charging the amount from the client's account is performed by the PSP (in the client's point of view). Therefore, the reasoning about this regard should be considered as one of the PSP's requirements.

The merchant's requirements **S1** and **S2** can be formalized into the following statements:

S1: $M \textit{ believes } M \textit{ CanProve } ($
$$PG \textit{ authorized credit}(PG, M, Price, Date)) \textit{ to } V$$
S2: $M \textit{ believes } M \textit{ CanProve } ($
$$C \textit{ authorized payment-order}(C, M, Price, Date)) \textit{ to } V$$

According to the requirement **S1**, the merchant can ensure that the payment gateway has authorized the transaction relevant to her. In other words, the payment gateway has accepted the merchant's *Credit* request by sending a *Credit* response to the merchant. In the requirement **S2**, the merchant can also ensure that the client has really requested to make the payment to her.

Referred to the mobile payment model presented in chapter 3, the requirements **S1** and **S2** can be modelled as the requirements to complete a transaction in the merchant's point of view in that, before delivering goods or services to the client, the merchant needs a guarantee that she will receive money from the client.

The client's requirements **B2**, **B3**, and **B4** can be formalized by the following statements:

B2, B3: $C \textit{ believes } C \textit{ CanProve } ($
$$I \textit{ authorized debit}(I, C, Price, Date)) \textit{ to } V$$
B4: $C \textit{ believes } C \textit{ CanProve } ($
$$M \textit{ authorized payment-order}(M, C, Price, Date)) \textit{ to } V$$

If the proof of the requirement **B2** is successful, the client can ensure that the issuer has authorized her *Debit* request. In other words, the issuer has deducted the requested amount from the client's account and sent an acknowledgement of the money deduction to the client.

According to the requirement **B3**, in order to guarantee that the client have made the payment to the accredited merchant, the client's *Debit* request must be authorized by the issuer. This means that the issuer has consulted with the acquirer, which is responsible for the merchant's *Credit* request, and agreed to transfer the money from the client's account to the merchant's account. To perform the money transfer, the merchant must be recognized and accredited by the acquirer.

The successful proof of the requirement **B4** ensures that the client has received the message containing a *Payment Ordering* response which can be considered as a receipt of payment from the merchant. It guarantees that the merchant will deliver the requested goods or services to the client after the completion of the transaction.

The requirements **B2**, **B3**, and **B4** can be modelled as the requirements to complete a transaction in the client's point of view stated in section 3.1.8 in that, to complete the transaction, the client needs guarantees that she will receive the goods or services requested after paying for them.

It can be seen that all of the party's requirements for payment transactions of iKP protocol can be formalized into the party's requirements of the proposed mobile payment model presented in section 3.1.8. Recall that Bellare *et al.* [BGH$^+$00] argued that 3KP protocol, the most secure version of iKP protocol, satisfies all the party's requirements for payment transactions. In this thesis, the party's requirements for payment transactions of KSLv1 and KSLv2 have been successfully analyzed in sections 7.6.3 and 7.6.4. The results show that our proposed protocols also satisfy all of the party's requirements stated in section 3.1.8. As mentioned above, the party's requirements of iKP protocol can be formalized into the proposed mobile payment model. Therefore, the proposed protocols satisfy the requirements as that of public-key based iKP protocol [BGH$^+$00] without any public-key operations required at the client side. As stated in [KP02], the achievement of proving the party's requirements infers that our protocols have the ability to resolve disputes among parties. In particular, KSLv1 and KSLv2 can provide undeniable proofs of all transactions relevant to involved parties.

7.8 Summary

In this chapter, we aimed to analyze the proposed framework and protocols on essential properties stated in the formal mobile payment model previously introduced in chapter 3 namely, goals of engaging parties ($Goals$), party's requirements for payment transactions (PR), and transaction security properties ($TSec$), to illustrate that the payment system based on the proposed framework or protocols satisfies the requirements of the practical and secure mobile payment system. As we mentioned in section 3.3 that such properties can be analyzed by using accountability logic. Then, we have investigated several existing accountability logics [Kai96, KN98, KP02] and found that they are inadequate to analyze e-commerce protocols based on symmetric cryptography which plays an important role in many mobile payment protocols including the proposed protocols: KSLv1 and KSLv2. To overcome this constraint, we proposed an extension of KP's logic [KP02] by presenting additional axioms and general assumptions which capture comprehensible reasoning about accountability of symmetric cryptographic messages. Moreover, the proposed logic provides clear explanation if a protocol fails to prove the accountability.

We then demonstrated the practical usefulness of the proposed logic by analyzing the proposed framework and protocols: our SET framework, KSLv1, and KSLv2. The results have shown that the accountability of the proposed SET framework and KSLv2 has been successfully proved. On the analysis of KSLv1, although some goals cannot be proved, the protocol has mechanisms which are capable of providing the accountability property. The successful proofs infer that the proposed framework and protocols achieve the following:

- They satisfy $Goals$, PR, and $TSec$, stated in the proposed mobile payment model.

- From the major use of the accountability property mentioned in KP's logic [KP02], the proposed framework and protocols offer the ability to resolve disputes among parties.

Combining the above proof results with the successful analysis on trust relationships among parties ($Trust$) and transaction performance (TP) of the payment systems based on the proposed framework or protocols previously presented in chapters 4 and 5, it can be seen that such systems satisfy the requirements of practical and secure mobile payment system stated in the Definition 3.15 of the proposed mobile payment model.

In addition, we have shown that the proposed logic is able to capture the party's requirements for payment transactions of iKP protocol [BGH$^+$00] by formalizing the requirements into the proof statements of the proposed logic naturally. We found that the party's requirements of iKP protocol can be reasoned as parts of the party's requirements (PR) of the proposed model. This emphasizes the generality of the proposed model.

Chapter 8

Limited-Use Key Generation for Internet Transactions

8.1 Background

Nowadays, several Internet services, such as file transfers, web services, or viewing electronic articles, deploy shared secrets. Referred to section 2.6, generally, the shared secret is used for two main purposes: as an authentication token, e.g. username and password, and as a key for cryptographic operations, e.g. an encrypting key or a key for keyed-hash operation. For simplicity, later on, we call the information shared between parties as a "*shared key*".

Among the applications of shared secrets stated above, on one hand, the access password needs to be updated periodically, and more frequently for encrypting keys to enhance security of the system. However, the higher frequency the secrets need to be updated, the lower performance the system will be. On the other hand, the credit-card number which is static, reusable information transferred in every transaction is possible to be eavesdropped by an attacker.

In this thesis, we focus our consideration of the security of shared secrets, either as authentication tokens or as cryptographic keys, in payment scenario. As previously discussed in section 2.6, several techniques have been proposed to secure the transfer of shared secrets during transactions [RW02, LZ04]. In chapter 5, we have presented session key generation techniques which have been applied to KSLv1 and KSLv2 protocols. These techniques generate a set of session shared keys based on hashing of bit-shifting of a long-term shared key. The session shared keys generated from this technique can be used either as authentication tokens or as the keys for encryptions and MAC (Message Authentication Code) operations.

However, as well as the security concern in [RW02] and [LZ04] discussed in sections 2.6.1 and 2.6.2, respectively, the key generation techniques presented in chapter 5 still require long-term shared keys between parties. Although such keys are not transferred during transactions, they are possible to be compromised by a party who intercepts session keys and successfully deciphers them.

In this chapter, we present a new session key generation technique which is able to solve the problems stated above. We select payment system as a reference scenario to illustrate our idea, yet our technique can be applied to other kinds of Internet applications.

This chapter is organized as follows. Section 8.2 presents our proposed session key generation technique in details. In section 8.3, the proposed session key generation technique is applied to KSLv1 and KSLv2 protocols to enhance their security. Section 8.4 shows that the session keys generated by the proposed technique are secure against attacks. Then, in section 8.5, we demonstrate the generality and practicability of the proposed technique by applying the technique to enhance the security of other kinds of payment schemes. Section 8.6 summarizes the chapter.

8.2 The Proposed Key Generation Technique

In this section, we propose a technique to generate session keys that solves the problems and limitations of existing techniques [RW02, LZ04] previously discussed in section 2.6. Our technique does not rely on any long-term shared keys. Therefore, the compromise of the long-term keys will not compromise the security of the system. The higher number the session keys have been used, the less chance the attacker can compromise the system. Moreover, our limited-use shared keys can be used as single-use authentication tokens such as disposable credit-card numbers (DCNs in short) or as the keys for cryptographic operations, such as encryptions or keyed-hash functions.

Section 8.2.1 presents initial settings of the proposed technique. In sections 8.2.2 and 8.2.3, the proposed key generation and key update techniques are given in details.

8.2.1 Initial Settings

Traditionally, the security of shared-key based systems relies heavily on the privacy of long-term shared keys. If the keys are revealed to an attacker, the security of the entire system will be compromised. In this section, we present a limited-use key generation technique that the key used in each session does not rely on any long-term shared keys so that the compromise of the long-term keys does not affect the security of the system. In the rest of the chapter, we call the long-term key shared between two parties as "*master key*" and the shared key used in each session as "*session key*".

Consider a situation where two parties, Alice and Bob, communicate to each other using a shared key K_{AB}. The key K_{AB} can be either used as an encrypting key or a key for keyed-hash function, e.g. $\{X\}_{K_{AB}}$ or $h(X, K_{AB})$, where X is a message and K_{AB} is the key shared between Alice and Bob, or as an authentication token. As the authentication token, K_{AB} needs to be protected by applying cryptographic operations to it, e.g. $\{K_{AB}\}_{K_{ex}}$, where K_{ex} is a session shared key generated from running a key exchange protocol between Alice and Bob. Practically, K_{AB} should be used for only short periods of time before being updated. Assume that the updated key is securely distributed between parties, what we concern is that it would increase performance of the system if the frequency of the key update process can be minimized.

As previously discussed in section 2.6, generating the set of session keys from a master key shared between parties is a possible solution. However, the key generation technique must be secure in that it is difficult for an attacker to compute the master key from the captured session keys.

To illustrate our technique, we assume that Alice and Bob want to communicate securely. The following initial settings have been made:

1. Alice and Bob share the master key K_{AB}. K_{AB} can be assumed to be never expired.

2. The "*distributed key*" DK is another shared key between Alice and Bob. However, DK needs to be updated periodically or upon their request. It is distributed by performing an authenticated key-exchange protocol between Alice and Bob.

3. Alice and Bob's computing devices are not necessarily secure against attacks. If required, they can be implemented using tamper-proof resistant devices such as smartcard. Moreover, they are capable of performing at least hash operations. Referred to the analysis by Krawczyk in [Kra99], to guarantee sufficient security, HMAC-supported devices are preferred.

4. $h(M, K)$ stands for the keyed-hash value of the message M and the key K. However, to guarantee higher security, HMAC is preferred.

8.2.2 Session Key Generation

The following steps show the details of the proposed technique:

1. Alice generates the key DK and distributes to Bob through a secure channel.

2. Alice and Bob generate a set of "*preference keys*" K_i, where $i = 1, ..., m$, based on K_{AB} and DK, and store it on their devices. The generation of K_i can be performed as follows:

   $$K_1 = h(DK, K_{AB}), K_2 = h(DK, K_1), ..., K_m = h(DK, K_{m-1})$$

   Later on, we name the kind of process as $K_A * K_B$, which denotes the generation of a set of keys from the output of $h(K_A, K_B)$ as described above. After generating K_i, the master key K_{AB} can be removed from the system. The set of K_i is not used in any transaction, but it is later used as a seed for generating session keys.

3. Alice generates a random number r, and sends it to Bob.

   Alice→Bob: r

   Alice selects two preference keys; one is K_{Mid_1}, where K_{Mid_1} is the middle key among $\{K_1, ..., K_w\}$, $w = r$ *mod* m. The other is K_{Mid_2}, where K_{Mid_2} is the middle key among $\{K_1, ..., K_{Mid_1}\}$. Then, Alice calculates "*session initialization*" (SI) key SIK, where

   $$SIK = h(K_{Mid_1}, K_{Mid_2})$$

   Note that, in general, K_{Mid_1} and K_{Mid_2} do not have to be the middle keys among the set of $\{K_1, ..., K_m\}$. It can be selected by using some other method which depends on the agreement between Alice and Bob. Moreover, after generating SIK, K_{Mid_1} and K_{Mid_2} are then removed from the system. On receiving r, Bob also can generate SIK.

4. Alice and Bob generate a set of session keys SK_j, where $j = 1, ..., n$, as shown below:

$$SK_1 = h(SIK, DK), SK_2 = h(SIK, SK_1), ..., SK_n = h(SIK, SK_{n-1})$$

Figure 8.1 depicts the proposed session key generation technique. After the set of SK_j has been generated, Alice and Bob can make use of it in several ways. Firstly, Alice can send SK_j as an authentication token to authenticate herself or authorize some actions to Bob over a secure channel, e.g. SSL channel or the encryption with Bob's public key. Secondly, SK_j can be used as an encrypting key in that Alice encrypts a message with SK_j and sends it to Bob. Alternatively, SK_j can be used as a key for keyed-hash function. On receiving the message, Bob can verify the message by considering whether or not the received SK_j matches the one he has. If they are matched, it infers that Alice has sent the valid SK_j to Bob. The applications of the proposed technique to KSLv1 and KSLv2 will be presented in section 8.3.

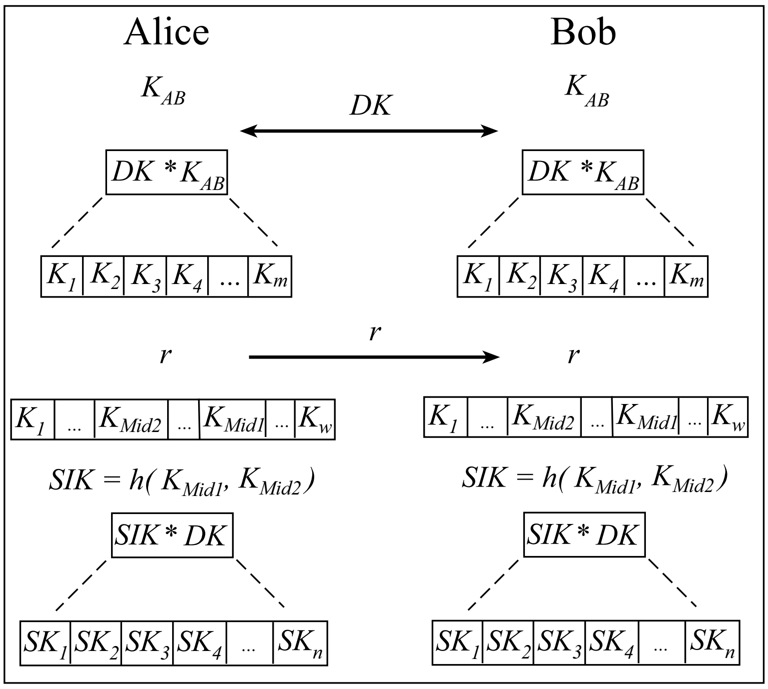

Alice
Bob
K_{AB}
DK
K_{AB}
$DK * K_{AB}$
$DK * K_{AB}$
K_1 K_2 K_3 K_4 ... K_m
K_1 K_2 K_3 K_4 ... K_m
r
r
r
K_1 ... K_{Mid2} ... K_{Mid1} ... K_w
K_1 ... K_{Mid2} ... K_{Mid1} ... K_w
$SIK = h(K_{Mid1}, K_{Mid2})$
$SIK = h(K_{Mid1}, K_{Mid2})$
$SIK * DK$
$SIK * DK$
SK_1 SK_2 SK_3 SK_4 ... SK_n
SK_1 SK_2 SK_3 SK_4 ... SK_n

Figure 8.1: Session key generation

8.2.3 Session Key Update

After being used for transactions for a specified period, the set of SK_j needs to be updated. The session key update can be performed as follows:

1. Suppose that, both Alice and Bob have used SK_j up to SK_p, where $1 \leq p \leq n$. Either Alice or Bob selects two preference keys; one is K'_{Mid_1}, where K'_{Mid_1} is the middle key among $\{K_1, ..., K_q\}$, $q = p\ mod\ rm$, rm is the number of remaining members in the set of K_i. The other is K'_{Mid_2}, where K'_{Mid_2} is the middle key among $\{K_1, ..., K'_{Mid_1}\}$. Then, they generate a new SI key SIK' as follows:

   $$SIK' = h(K'_{Mid_1}, K'_{Mid_2})$$

   After generating SIK', K'_{Mid_1} and K'_{Mid_2} are removed from the set of $\{K_1, ..., K_q\}$.

2. Alice and Bob then generate a new set of session keys SK'_j, where $j = 1, ..., n$, as follows:

   $$SK'_1 = h(SIK', DK), SK'_2 = h(SIK', SK'_1), ..., SK'_n = h(SIK', SK'_{n-1})$$

   Note that the distributed key DK does not need to be updated every time a new set of session keys SK_j is generated. Updating DK also results in updating the set of K_i. Alice and Bob can repeatedly use DK until the set of K_i is depleted. Figure 8.2 depicts the proposed session key update technique.

3. To update a new distributed key, after the new key distributed key DK' has been distributed, the following process can be performed. Assume that the currently-used session key is SK_u, Alice and Bob select K_v among the set of $\{K_1, ..., K_{rm}\}$, where $v = u\ mod\ rm$. Then, they

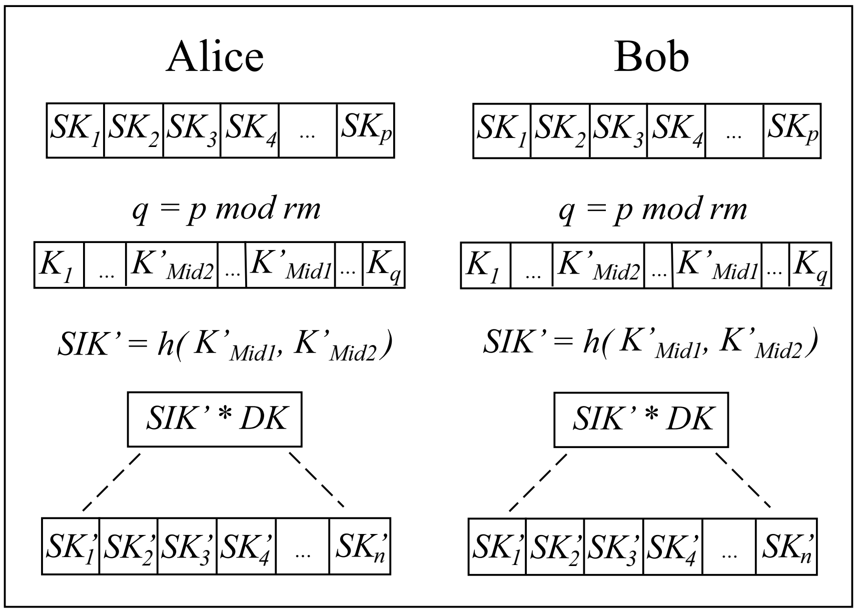


Figure 8.2: Session key update

compute $K_v * DK'$ to generate a new set of preference keys K'_i, where $i = 1, ..., n$. Alice and Bob can follow the key updating process to generate a new SI key and a new set of session keys.

It can be seen that generating each set of session keys does not rely on any long-term keys, but on randomly chosen preference keys (as a result of selecting the p-th session as the last session before SK_j is being updated). As a result, the proposed technique not only enhances the security of the system against key compromise, but also reduces the frequency of key update process which results in an increase of transaction performance.

8.3 Applying the Proposed Key Generation Technique to Secure KSLv1 and KSLv2 Protocols

In this section, we demonstrate that the session key generation technique presented in section 8.2 can enhance the security of KSLv1 and KSLv2 protocols. We found that the proposed technique can be applied directly to KSLv1 and KSLv2 due to the similar environment settings as illustrated below:

1. CCI is considered as a master key shared between the client and the issuer.
2. X, Y, and Z are considered as the distributed keys shared between the client and the merchant, the client and the issuer, and the merchant and the payment gateway, respectively.
3. The sets of Y_i, X_i, and Z_j, where $i, j = 1, ..., n$, are considered as the sets of session keys shared among engaging parties.

Sections 8.3.1-8.3.3 illustrate how to generate the sets of Y_i, X_i, and Z_j, respectively.

8.3.1 Generating Y_i

In KSLv1 and KSLv2, the client needs to register herself to the issuer in order to share the distributed key Y. Applied the proposed technique, the master key CCI is also distributed during the registration process. The following steps can be done to generate the set of session keys Y_i.

1. Both client and issuer generate a set of preference keys K_i^Y, where $i = 1, ..., m$, by computing $Y * CCI$, and store it on their devices. At this stage, CCI is removed from the system.

2. The client generates a random number r and sends it to the merchant in **Step 1** (referred to the protocol descriptions of KSLv1 and KSLv2 presented in sections 5.4.3 and 5.5.3). r will then be forwarded to the issuer in **Step 4.1**. **Step 1** and **Step 4.1** of KSLv1 are now modified as follows:

   Step 1 C→M: $ID_C, r, s, TIDReq, MIDReq$

   Step 4.1 PG→I: $MAC[(Price, h(OI), ID_M), Y_i], h(OI), r, TID,$
   $Price, ID_C, ID_M$

   Note that, The above modification applies to both KSLv1 and KSLv2 protocols. In **Step 1**, s stands for a random number generated by the client for generating X_i (its details will be presented in the next section). Alternatively, r can be generated by either the client or the issuer during the registration.

3. The client and the issuer select the keys $K^Y_{Mid_1}$ and $K^Y_{Mid_2}$ and compute SIK, where $SIK = h(K^Y_{Mid_1}, K^Y_{Mid_2})$.

4. The client and issuer generate the set of session keys Y_i, where $i = 1, ..., n$, by following $SIK * Y$ and use it in the transactions.

$$Y_1 = h(SIK, Y), Y_2 = h(SIK, Y_1), ..., Y_n = h(SIK, Y_{n-1})$$

In KSLv1 and KSLv2, Y_i is used as a MAC key to authenticate the client to the issuer. On receiving the message in **Step 4.1**, the issuer can verify the message with the session key Y_i that it has. If the verification is successful, the issuer can infer the client with valid credit-card information. In **Step 6**, the client can verify that the message has been sent from the issuer by decrypting the message with the Y_i that she has.

8.3.2 Generating X_i

Based on the proposed technique, the set of X_i can be generated as follows.

1. After the client and the merchant share the distributed key X, they generate a set of preference keys $\{K_1^X, ..., K_m^X\}$ as follows.

   $$K_1^X = h(X, X_{m-bit-shift}), K_2^X = h(X, K_1^X), ..., K_m^X = h(X, K_{m-1}^X)$$

   where $X_{m-bit-shift}$ stands for m-bit cyclic shifting of X. The set of K_i^X is stored at both client and merchant's devices.

2. The client generates a random number s and sends it to the merchant in **Step 1**. Both client and merchant then select two preference keys: $K_{Mid_1}^X$ and $K_{Mid_2}^X$. Then, they calculate SIK', where $SIK' = h(K_{Mid_1}^X, K_{Mid_2}^X)$.

3. The client and the merchant generate the set of X_i, where $i = 1, ..., n$, by following $SIK' * X$ and use it in the transactions.

   $$X_1 = h(SIK', X), X_2 = h(SIK', X_1), ..., X_n = h(SIK', X_{n-1})$$

The update of the sets of Y_i and X_i follows the process presented in section 8.2.3.

8.3.3 Generating Z_j

Generating the set of Z_j, where $j = 1, ..., n$, follows the key generation technique for X_i by replacing X with the distributed key Z shared between the merchant and the payment gateway. Either the merchant or the payment gateway also generates a random number t which is used in preference key selection process. As Z_j is used only in KSLv2 protocol, **Step 3** of KSLv2 is modified as follows:

$$\textbf{M}\rightarrow\textbf{PG}\text{: } \{MAC[(Price, h(OI), ID_M), Y_i], h(OI), r, t, TID, Price, ID_C, ID_I\}_{Z_j}, ID_M, MAC[(h(OI), TID, ID_C, ID_I), Z_{j+1}]$$

At the end of the process, the set of preference keys $\{K_1^Z, ..., K_m^Z\}$ and the set of session keys $\{Z_1, ..., Z_n\}$ are generated and stored on both merchant's and payment gateway's terminals. The merchant and the payment gateway can use the set of session keys Z_j, where $j = 1, ..., n$, in transactions.

8.3.4 Implementation Issues

In this section, we discuss major issues related to the implementation of the proposed technique in KSLv1 and KSLv2 protocols. The results obtained from the discussion can also be applied to other kinds of Internet applications.

Key Distribution

The proposed session key generation technique rather focuses on the deployment of the master key than the semi-secret credit-card number. Therefore, the set $\{CCI, Y, r\}$ requires to be distributed between the client and the issuer.

Before making the first payment, the client needs to register herself to the issuer in order to share $\{CCI, Y, r\}$. In a smartcard-based environment, if the client is provided a smartcard reader, the issuer can generate the set $\{CCI, Y, r\}$ and store it on the smartcard before issuing the card to the client. Therefore, r and Y do not need to be transferred in any transactions resulting in fully off-line key generation.

Storing and Managing Keys

The proposed technique requires two sets of keys: preference keys and session keys, to be generated at each party's device. However, only the entire set of preference keys is stored on the devices, whereas each member in the set of session keys is generated at the beginning of each transaction in order to reduce storage requirement. As the preference keys are not transmitted in any transactions, their security is guaranteed as long as the device is not compromised.

After generating the set of preference keys, the master keys are no longer used in the system. As well as the session keys X_i, Y_i, and Z_j, after a new session key has been generated, the previously used key is removed from the system.

8.4 Security of Session Keys Against Attacks

Obviously, one of the most concerns in designing KSLv1 and KSLv2 protocols is security of the session keys against attacks. In this section, we discuss how the session keys generated from the proposed key generation technique are secure against various kinds of attacks.

Referred to the technique presented in section 8.2.2, we consider the situation where a session key SK_j is transmitted in cleartext over an insecure channel. First of all, due to the one-way property of hash function, reverse operation of SK_j to retrieve SIK is computationally infeasible. Referred to [Kra99], HMAC algorithm is suggested due to its proven security. One possibility is to collect a number of session keys and try to guess the next value of SK_j. However, the fraud will be detected by allowing limited number of attempts on user authentication process. For example, in payment system scenario, an attacker who impersonates as a client tries to authenticate herself to the issuer by sending a number of session keys to it. If the attempts exceed the specified limit, the client's account is suspended. The system then notifies the client that there were unauthorized attempts on her account and requests the client to update the SI key. After the client updates the SI key, the attacker has to repeat all the attacking processes from the beginning. Moreover, as stated that SK_j can be used as a key for encryption and keyed-hash function, its key length should be at least 128 bits that is compatible with AES and HMAC-MD5 algorithms. Therefore, the search space to retrieve the correct SK_j is at least 2^{128}.

Considering the case that the above attempts have not been detected by the system, the set of session keys SK_j is valid for the short periods of time. After the session keys are updated, the keys in the attacker's hands are no longer valid.

As the initial input for the set of SK_j are SIK and DK. The attacker must be able to record all session keys from SK_1 and try to compute SIK and DK. In the case that the attempt is successful, the attacker can generate the next session key. However, the fraud will be eventually detected because the session key generated by the client is not valid in the system in that it will be notified as *used* by the issuer. At this stage, the client does not need to update the master key. What she needs to do is requesting the issuer to update the set of session keys. After the new set of session keys SK'_j becomes valid, the current SI key SIK is no longer valid because SIK' is based on the preference keys that are not used in any transactions, not SIK. To retrieve SIK', the attacker needs to capture the message that contains SK'_1 and attempts to compute SIK'.

The only possible successful attack to the proposed technique is that the attacker must be able to perform the following:

1. Capture DK distributed over the network,
2. Access to each party's device to retrieve the entire set of preference keys K_i,
3. Record all SK_j transmitted in all transactions, and
4. Detect the request to update the set of session keys so that she can know

p in order to select K_{Mid_1} and K_{Mid_2} from the set of K_i.

In the worst case, if the attacker succeeds in the above process, she can generate and use the valid SK_j until being detected by the system.

We can see that the proposed key generation technique is not based on any long-term shared keys. Generating each set of session keys is based on dynamic parameters randomly chosen from the set of preference keys K_i. As a result, the higher number the transactions have been performed, the less chance the system is compromised.

8.5 Applications of the Proposed Technique to Other Kinds of Internet Transactions

In this section, we demonstrate the usability of the proposed session key generation technique to various kinds of Internet payment systems, namely credit-card payment scheme over SSL protocol [FKK96], SET protocol [Mas97], and conventional credit-card payment scheme. Sections 8.5.1 to 8.5.3 present their details.

8.5.1 Credit-card Payment Over SSL

As discussed in section 2.6, several techniques [RW02, LZ04] have been proposed to solve the problems of credit-card payment over SSL that mainly come from the security of the SSL protocol itself and the exposure of the client's credit-card number to the merchant as it is transferred in cleartext over SSL channel. However, they are still based on long-term shared keys which are vulnerable to attacks.

Applying our technique to the credit-card payment over SSL protocol is more straightforward compared to KSLv1 and KSLv2 protocols. That is, after a SSL connection between a client and a merchant is established, payment-related information is then transferred through it. What the client needs to do is transmitting a session key representing the client's credit-card information through the SSL channel.

To apply the proposed technique to this system, three secrets are shared between the client and the issuer: the master key K_{CI}, the distributed key DK, and the semi-secret, 16-digit credit-card number CCN.

After the client selects products or services from the merchant's site, the payment screen is displayed. The client is then required to fill in her credit-card number and password to authenticate herself to her device. After the authentication is successful, the session key SK_j is generated and transmitted to the merchant together with payment information over the SSL channel as follows:

$$\mathbf{C{\rightarrow}M:} \quad \{OD, Price, ID_I, r, h(Price, SK_j, CCN, r)\}_K$$

where K stands for the key shared between the client and the merchant generated from the SSL handshake session. $Price$ and OD stand for the client's requested price and goods descriptions (including transaction ID and timestamp), respectively. ID_I stands for the issuer's identity which is normally the first four digits of CCN. Note that the random number r needs to be transmitted only in the first transaction in the system.

On receiving the message, the merchant can retrieve only OD, ID_I and $Price$. She recognizes the issuer from ID_I. The merchant then forwards $\{Price, r, h(Price, SK_j, CCN, r)\}$ to the issuer. On receiving the message, the issuer verifies $h(Price, SK_j, CCN, r)$ and then sends an approval result to the merchant and the client.

It can be seen that other parties including the merchant cannot retrieve the master secret K_{CI} or even CCN. Applying the hash function to SK_j together with $Price$ and CCN, not only the merchant cannot retrieve the session key

SK_j, but the issuer is also guaranteed that the merchant cannot modify the requested price $Price$.

8.5.2 SET Protocol

The technique presented in section 8.5.1 can be exploited in SET protocol [Mas97]. In SET, the encrypted client's *Payment Information* (PI) containing credit-card information is decrypted by the payment gateway and later forwarded to the issuer under a banking private network. In fact, the credit-card information should be known only to the issuer because, in practical, the payment gateway and the issuer may be different parties.

Based on the fact that the purpose of transferring credit-card information is to authenticate the client to the issuer. It is therefore not necessary to be revealed to any other parties including the payment gateway (if the payment gateway and the issuer are different parties). Applying the proposed technique can conceal the credit-card information from the payment gateway. In SET protocol, the original client's PI is shown as follows:

$$\{TID, h(OD, Price), Price, ID_M, CCI\}_{K_{PG}}$$

where K_{PG} stands for the payment gateway's public key, TID stands for transaction ID containing time and date of the transaction, OD stands for order descriptions, ID_M stands for the merchant's identity, and CCI stands for credit-card information. Applied the proposed technique, PI can be modified as follows:

$$\{TID, h(OD, Price), Price, ID_M, SK_j, r\}_{K_{PG}}$$

Note that r has to be transmitted only in the first transaction in the system. With this method, the payment gateway only recognizes that the client has requested it to authorize the issuer for this payment, but SK_j is transparent to it.

8.5.3 Conventional Credit-card Payment

One of the concerns about the deployment of disposable credit-card numbers (DCNs) that has been pointed out in [RW02] is the compatibility with conventional electronic commerce sites which accept 16-digit credit-card numbers. Thus, the length of each DCN must be 128 bits and readable by the client. Alphanumeric numbers should be the proper format. In the proposed key generation technique presented in section 8.2, the length of each session key is equivalent to the length of the key for AES encryption algorithm or the key for HMAC-MD5 algorithm so that the session key can be applied to cryptographic operations as stated in section 8.2.2.

However, as stated in [RW02], if the keys need to be alphanumeric, the search space to generate the right key is 36^{11} (the first 4-digit number is the issuer's identity and the last digit is checksum). With the security of our technique discussed in section 8.4, the proposed key generation technique is still secure with this key length.

8.6 Summary

In this chapter, we have pointed out that the session key generation technique applied to KSLv1 and KSLv2 protocols cannot provide sufficient security against long-term key compromise. In other words, the compromise of the long-term shared key will compromise the security of the entire system. We then proposed an efficient technique to generate and update limited-use shared keys. We have demonstrated that the proposed technique is secure against key compromise attacks. Also, the generation of each set of session keys is based on dynamically chosen preference keys so that the system is still secure although some keys are compromised.

We have demonstrated the usability of the proposed technique by applying it to enhance the security of KSLv1 and KSLv2 protocols. Moreover, we have shown the generality and practical usefulness of the proposed technique by demonstrating the applications of the proposed technique to various kinds of credit-card payment systems including SSL-based payment scheme [FKK96] and SET protocol [Mas97]. The results have shown that the proposed technique is advantageous not only to credit-card payment systems, but also to other kinds of Internet applications that deploy shared secrets.

Chapter 9

Securing Token-Based Mobile Payment

In chapter 5, we have presented two non-proxy, account-based mobile payment protocols, KSLv1 and KSLv2, that are compatible with the proposed formal mobile payment model presented in chapter 3. KSLv1 and KSLv2 deploy the symmetric-key based cryptographic technique which not only reduces party's computation, but also satisfies transaction security properties stated in the proposed formal model.

In this chapter, we demonstrate an extensive application of the proposed cryptographic technique by applying it to enable secure token-based mobile payment. In particular, we present a prepaid non-proxy, micropayment protocol which is suitable for wireless environments. The proposed protocol offers higher security and transaction performance compared to existing micropayment protocols while applying it to a wireless environment. We then propose an extension of the proposed protocol to perform postpaid micropayment transactions.

This chapter is organized is follows. Section 9.1 outlines necessary background. In section 9.2, the proposed micropayment protocol is introduced. Our protocol is based on prepaid micropayment, yet can be extended to perform postpaid micropayment as shown in section 9.3. Section 9.4 discusses its transaction security and performance issues. Section 9.5 summarizes the chapter.

9.1 Background

Micropayment seems to be more widely accepted than other kinds of payment systems for wireless environments because of its lightweight, lower setup cost, and lower transaction cost. Moreover, most payment-related applications for wireless networks are conducted with small-valued goods or services e.g. downloading ring tones, operator logos, or electronic document.

Traditionally, micropayment protocols such as PayWord [RS96] or NetCard [AMS97] deploy public-key operations and chains of hash values. Although these protocols apply well for fixed networks, they are not suitable for applying to wireless networks due to a number of limitations of wireless environments [KSL03a, RdS98] previously discussed in section 1.3.

Recently, a prepaid micropayment protocol called PayFair [Yen01] offers the ability to perform payment transactions on limited computational capability devices. It deploys symmetric-key operations and keyed-hash functions to reduce the computation at engaging parties. However, PayFair lacks of transaction privacy since payment-related information of engaging parties is transferred in cleartext during transactions. In PayFair, the bank is assumed to be fully trusted by its clients. This is because a message sent from a client to a merchant in PayFair lacks of non-repudiation property. This offers an opportunity for the bank to impersonate as its client to perform transactions. Note that we have discussed the issue related to the trust relationships among engaging parties in chapter 3, and a concrete example has been provided in the discussion of SET protocol [Mas97], whereby the client has to trust the payment gateway that it will not reveal her credit-card information to other parties, in section 4.4.2. In addition, in PayFair, a payment token authorized by the bank is merchant-specific in that it can be used to generate the coins to spend with only one specified merchant. Thus, the client has to request the bank to issue a new payment token every time she wants to make a payment to a new merchant.

In this chapter, we propose a prepaid micropayment protocol which deploys the secure symmetric cryptographic technique presented in section 5.2. With this technique, not only the computation at all engaging parties, especially at the client, is reduced, but the protocol also satisfies transaction security properties, including non-repudiation, stated in our mobile payment model (see section 3.1.9 for details). Moreover, it offers the ability to resolve disputes among parties. In our micropayment protocol, the private information, such as payment information and secret keys, of all engaging parties is well-protected. Furthermore, we apply the key generation technique presented in chapter 8 to enhance the security of the system.

In any prepaid payment system, a client has to purchase an electronic coupon which contains spending credits. The amount paid by the client is transferred to a specified merchant before the first transaction takes place. In our proposed protocol, we present an efficient method to refund either unspending credits or coupons. This offers the practicability to the system. Moreover, the coupon in our protocol is general-purposed in that it can be split into smaller valued, merchant-specific coupons to spend with many merchants.

Besides, we show that our micropayment protocol is general enough to be used for both prepaid and postpaid environments. That is, in addition to prepaid micropayment, it can perform postpaid micropayment transactions. Therefore, this results in a general micropayment framework.

We analyze performance of the proposed protocol and compare it with the performance of other micropayment protocols viz., PayWord [RS96], PayFair [Yen01], and Park *et al.*'s protocol [PBD01] to show that our protocol has better performance than the others in terms of the party's computation and the number of message passes. Therefore, the proposed protocol can be implemented on limited capability wireless devices with higher transaction performance than the existing micropayment protocols.

9.2 The Proposed Micropayment Protocol

9.2.1 Overview of the Proposed Protocol

The proposed protocol is composed of three engaging parties: client, merchant, and bank. For simplicity, we assume that the bank performs the tasks of issuer, acquirer, and payment system provider (referred to our mobile payment model presented in chapter 3). At the beginning of the protocol, the client requests the bank for the authorization to perform a micropayment transaction. The bank checks the validity of the client's account and issues a *bank coupon* containing the amount requested by the client. Meanwhile, the bank deducts the amount that is equivalent to the value of the bank coupon from the client's account.

To make a payment to the merchant, the client generates a *merchant coupon* containing the value specified to the merchant. The value of the merchant coupon must not exceed the value of the bank coupon. This coupon has to be validated by the bank before being used. To validate the merchant coupon, the client generates a set of coins, attaches it into the merchant coupon, and sends the merchant coupon to the bank. After the validation, the bank transfers the amount equivalent to the value of the coins to the merchant's account. Then, the client can make payment to the merchant up to the amount specified in the merchant coupon.

In the proposed protocol, the bank is trusted by the client to generate the correct number and value of the coins for coin validation purpose, but it is not trusted to create payment initialization requests to merchants by itself. This is because the bank itself can generate the sets of coins. It is possible to generate fake requests on behalf of its clients according to the trust relationships among engaging parties stated in the Definition 3.11 of the proposed formal model.

9.2.2 Initial Assumptions

The initial assumptions of the proposed protocol are given as follows:

1. The client shares a long-term secret CCI with the bank. CCI represents the client's account information which is known only between the client and the bank.

2. Three secrets are shared among engaging parties as follows:

   - X is shared between the client and the merchant.
   - Y is shared between the client and the bank.
   - Z is shared between the merchant and the bank.

   The above secrets are distributed and updated periodically or upon the requests by the engaging parties. After the secrets X, Y, and Z have been distributed, each party generates the sets of secrets X_i, Y_i, and Z_j, where $i, j = 1, ..., n$, by using the session key generation techniques presented in sections 8.3.2, 8.3.1, and 8.3.3, respectively, and stores them on each party's device. These sets of secrets will be used as encrypting keys in the protocol. The purpose of generating these sets of keys is to reduce the frequency of key update process and to enhance security of the system. Note that such secrets can be distributed by using an authenticated key-exchange (AKE) protocol for wireless networks. The details of existing AKE protocols for wireless networks have been presented in section 4.4.9.

3. The bank is trusted by the client in that it will not reveal the client's CCI to other parties.

4. $h(X, K)$ represents the keyed-hash value of a message X with the key K. It can be presented by Message Authentication Code (MAC) to achieve higher security.

Our proposed protocol is composed of 6 sub-protocols: *Setup, Payment Initialization, Payment, Extra Credit Request, Coupon Cancellation,* and *Coin Return* protocols. Sections 9.2.3 to 9.2.8 demonstrate the details of each sub-protocol. The proposed protocol is depicted in Figure 9.1.

9.2.3 Setup Protocol

In *Setup Protocol*, the client requests the bank for the authorization on making a micropayment transaction with the amount CL_T as follows:

$$\textbf{C}{\rightarrow}\textbf{B}: \quad ID_C, CL_T, T_{CP}, h(CL_T, T_{CP}, Y_i), r \tag{9.1}$$

Note that CL_T stands for total credits that the client is allowed to spend in the system. T_{CP} stands for the timestamp while generating the request. r stands for a random number used to generate the set of Y_i (referred to the technique for generating the set of Y_i presented in section 8.3.1). $h(CL_T, T_{CP}, Y_i)$ is used to protect integrity of the message. The bank checks the validity of the client's account and then deducts the amount CL_T from the client. The bank sends the client a *bank coupon* that can be used to perform transactions.

$$\textbf{B}{\rightarrow}\textbf{C}: \quad \{CL_T, T_T, T_{CP}, SN, c\}_{Y_i} \tag{9.2}$$

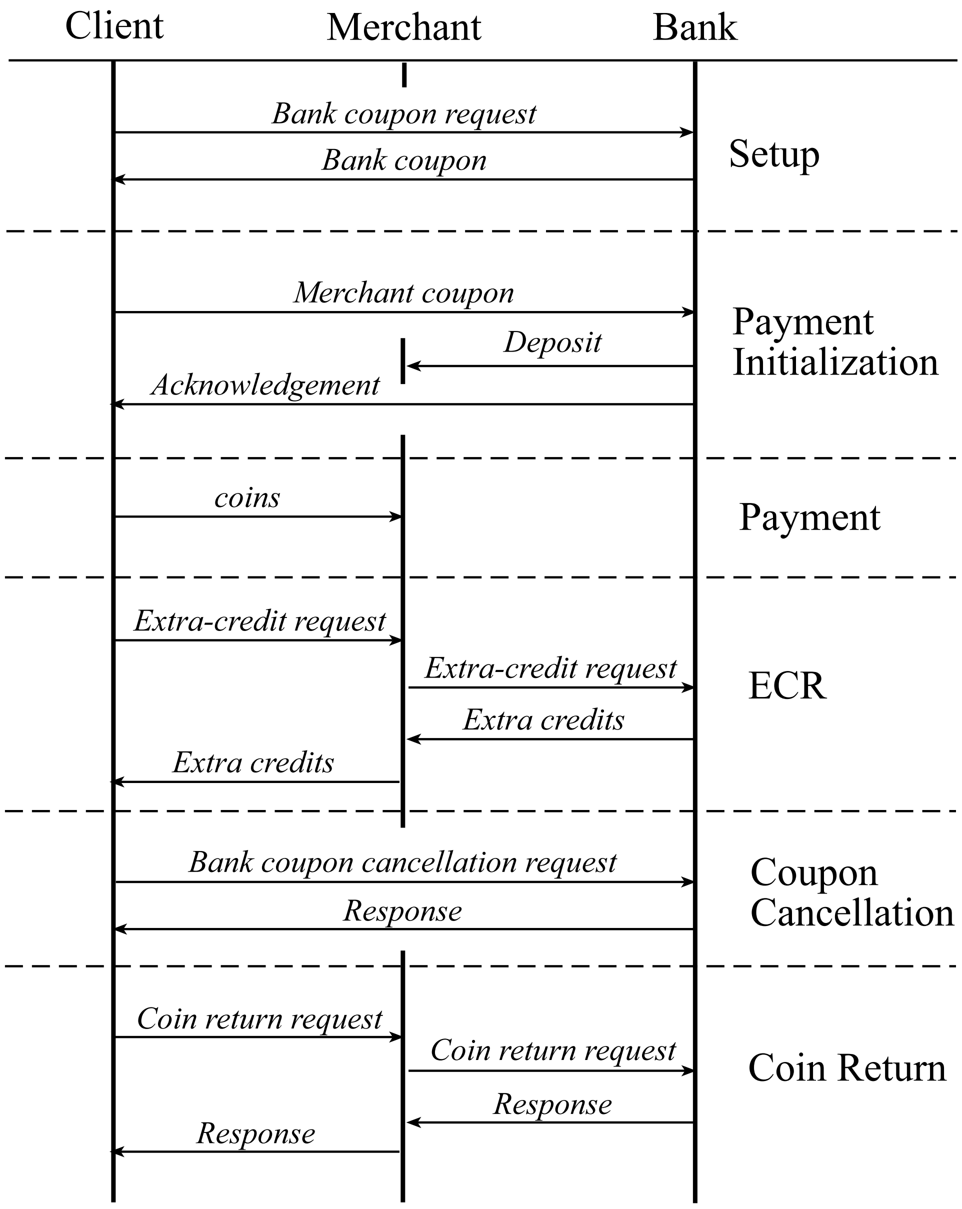


Figure 9.1: The proposed micropayment protocol

The bank coupon contains a unique serial number SN assigned by the bank and authorized credits CL_T. T_T stands for timestamp while issuing CL_T, and c is a random number generated by the bank used for generating coins. With this bank coupon, the client can make payment to many merchants repeatedly up to CL_T. After running out of the credits, the client needs to run this protocol to request the bank for a new CL_T again.

9.2.4 Payment Initialization Protocol

To make a payment to the merchant, the client generates a set of coins c_j, where $j = 0, ..., m,\ m = CL_T$, as follows:

$$\begin{aligned} &c_m = \{c, T_G\} \\ &c_j = h(c_{j+1}) \qquad \text{where} \quad j = 0, ..., m-1 \end{aligned}$$

The client specifies the amount CL_M to spend with the merchant. The client attaches the coins and CL_M into a *merchant coupon*, and sends the coupon to the bank:

$$\begin{aligned} \mathbf{C}{\rightarrow}\mathbf{B}: \quad & h(ID_M, c_0, T_G, CL_M, CL_T, T_T, SN, Y_i), \\ & h(c_0, T_G, CL_M, s, X_i), T_G, s \end{aligned} \tag{9.3}$$

where T_G stands for the timestamp while generating the set of coins. s stands for a random number used for generating the set of X_i (referred to section 8.3.2). Note that the client can either spend the entire credits to only one merchant or spend some credits to this merchant and spend the rest to other merchants. We can see that $h(c_0, T_G, CL_M, s, X_i)$ represents a part of *Payment Ordering* request (referred to the Definition 3.6 of the proposed formal model) from the client to the merchant which is unreadable by the bank.

The bank retrieves CL_T and CL_M from $h(ID_M, c_0, T_G, CL_M, CL_T, T_T, SN, Y_i)$, which is considered as a *Debit* request, and checks whether $CL_T < CL_M$. If so, it rejects the request. If $CL_T > CL_M$, the bank calculates the client's remaining credits CL_{TR}, where $CL_{TR} = CL_T - CL_M$. It then maintains the list of CL_{TR} to prevent over-spending problem. At this stage, the bank transfers CL_M to the merchant's account. Then, the bank sends the following messages to the merchant and the client, respectively:

$$\begin{aligned}\mathbf{B{\rightarrow}M}: \quad & \{c_0, T_G, SN, CL_M, h(ID_M, SN, CL_{TR}, T_{TR}, Y_i)\}_{Z_j}, \\ & h(c_0, T_G, CL_M, s, X_i), s, t \end{aligned} \tag{9.4}$$

$$\mathbf{B{\rightarrow}C}: \quad h(ID_M, SN, CL_{TR}, T_{TR}, Y_i), T_{TR} \tag{9.5}$$

where T_{TR} stands for the timestamp while the bank updates CL_{TR}. Note that T_T is updated to T_{TR} after calculating CL_{TR}. t stands for a random number used to generate the set of Z_j (referred to section 8.3.3). The message 9.4 is considered as a *Credit* response sent from the bank to the merchant. The merchant retrieves c_0 and CL_M from the encrypted message. The merchant also knows that the client has requested to make the payment to her from $h(c_0, T_G, CL_M, s, X_i)$, and the client's request has been authorized by the bank from the message encrypted with Z_j shared between the bank and herself. After receiving the message 9.5, later on, the client can use $\{CL_{TR}, T_{TR}\}$ to make a payment to another merchant. Note that the message 9.5 is considered as a *Debit* response sent from the bank to the client.

In this protocol, the merchant or the bank cannot impersonate as the client because they communicate to one another with different sets of secret keys and those keys are not revealed during the transaction.

9.2.5 Payment Protocol

After completing the *Payment Initialization Protocol*, the client can make a payment to the merchant by sending the coins as follows.

$$\mathbf{C}\!\rightarrow\!\mathbf{M}: \quad c_j \qquad where \quad j = 1, ..., m \tag{9.6}$$

Note that the combination of c_j and $h(ID_M, SN, CL_{TR}, T_{TR}, Y_i)$ from the messages 9.4 or 9.5 represents the *Payment Ordering* request from the client. The merchant verifies the requested amount by comparing c_j with c_0. After the verification, she delivers goods or services to the client. At the completion of each payment, the amount in the merchant coupon CL_M is deducted. The client is allowed to make the payment up to CL_M without any payment authorization required by the bank. If the remaining credits are insufficient to make another payment, the client can request the bank for extra credits by running *Extra Credit Request Protocol* which will be presented in the next section.

9.2.6 Extra Credit Request Protocol

Normally, when the client spends the credits up to CL_M, she needs to run the *Setup Protocol* to issue a new bank coupon. In our protocol, we reduce the frequency of performing this process by running *Extra Credit Request (ECR) Protocol* instead. With the *ECR Protocol*, the number of message passes in the entire transaction is reduced. Before the next transaction, the client checks whether $j > CL_M$. If so, the client still can purchase the goods, but she needs to request for extra credits from the bank. The client realizes that, if her request for the extra credits has been approved, her total credits CL_{TR} will be deducted by CL_M. To request for the extra credits, the client sends the following message to the merchant:

$$\mathbf{C}\!\rightarrow\!\mathbf{M}: \quad c_j, CL_M, h(ID_M, CL_M, T_G, SN, CL_{TR}, T_{TR}, Y_i) \tag{9.7}$$

At this stage, CL_M stands for the new credits to spend with the merchant whose identity is ID_M. The merchant retrieves CL_M and forwards the following message to the bank:

$$\mathbf{M}\!\rightarrow\!\mathbf{B}: \quad ID_M, h(ID_M, CL_M, T_G, SN, CL_{TR}, T_{TR}, Y_i) \tag{9.8}$$

The bank retrieves CL_{TR} and CL_M, and then calculates a new CL_{TR}, where $newCL_{TR} = currentCL_{TR} - CL_M$. The bank transfers CL_M to the merchant's account, and sends the response to the merchant as follows:

$$\begin{aligned} \mathbf{B}\!\rightarrow\!\mathbf{M}: \quad & h(ID_M, SN, CL_{TR}, T_{TR}, Y_i), T_{TR}, Yes, \\ & h(Yes, CL_M, T_{TR}, Z_j) \qquad \textit{if approved} \\ (\textit{or Rejected} \quad & \textit{if client has insufficient credits}) \end{aligned} \tag{9.9}$$

The merchant then checks whether or not the authorized CL_M in $h(YES, CL_M, T_{TR}, Z_j)$ is equal to CL_M received from the client in the message 9.7. If so, the merchant sends the client the following message:

$$\mathbf{M}\!\rightarrow\!\mathbf{C}: \quad h(ID_M, SN, CL_{TR}, T_{TR}, Y_i), T_{TR} \tag{9.10}$$

The client expects to receive the updated CL_{TR}, where $updatedCL_{TR} = currentCL_{TR} - CL_M$. She calculates CL_{TR} and compares with the received CL_{TR}. If they are matched, the client can infer the remaining credits in the updated bank coupon from CL_{TR}. The above message is considered as a notification of the client's remaining total credits. Note that, to make the payment to a new merchant, the client repeats the *Payment Initialization Protocol* with the updated bank coupon with the current CL_{TR} without running the *Setup Protocol* as that in existing protocols [RS96, Yen01].

9.2.7 Coupon Cancellation Protocol

In our protocol, the client is able to refund any un-used bank coupon previously purchased from the bank by sending the following message to the bank:

$$\mathbf{C}\rightarrow\mathbf{B}: \quad SN, T_{CR}, h(SN, CL_T, T_T, T_{CR}, Y_i) \tag{9.11}$$

where T_{CR} is timestamp while requesting for the coupon cancellation. The bank removes the coupon with the serial number SN from its database. Then, the bank transfers the amount CL_T to the client's account and sends the response of the client's request to the client as follows:

$$\mathbf{B}\rightarrow\mathbf{C}: \quad CancelOK, h(CancelOK, SN, T_{CR}, Y_i) \tag{9.12}$$

After receiving the message, the client is notified that the coupon with the serial number SN is no longer valid in the system. To make a payment to a merchant, the client needs to purchase a new bank coupon from the bank.

9.2.8 Coin Return Protocol

In some situation, the client may want to end the transaction with the merchant after spending some credits and request the merchant to refund the un-spending credits. This process can be performed as follows:

$$\mathbf{C{\rightarrow}M:} \quad c_{j_{max}}, T_G, h(ID_M, c_0, T_G, Y_i) \tag{9.13}$$

where $c_{j_{max}}$ is the highest-valued coin currently spent with the merchant. The merchant checks whether the received $c_{j_{max}}$ is equal to the $c_{j_{max}}$ that she has. If they are matched, the merchant forwards the following message to the bank:

$$\mathbf{M{\rightarrow}B:} \quad ID_M, c_{j_{max}}, T_G, h(ID_M, c_0, T_G, Y_i) \tag{9.14}$$

The bank retrieves $c_{j_{max}}$ and c_0 and calculates returned amount, where $returnedAmount = CL_M - j_{max}$. The bank then transfers the returned amount to the client's account and updates the client's bank coupon with the new CL_{TR}, where $updatedCL_{TR} = currentCL_{TR} + returnedAmount$. The bank updates the entry in the list containing T_G and c_0, and then sends the acknowledgement to the merchant.

$$\begin{aligned}\mathbf{B{\rightarrow}M:} \quad & h(returnedAmount, ID_M, c_0, T_G, CL_{TR}, T_{TR}, Y_i), \\ & h(returnedAmount, ID_C, c_0, T_G, T_{TR}, Z_j), T_{TR}\end{aligned} \tag{9.15}$$

The merchant is notified that the returned amount *returnAmount* has been withdrawn and transferred to the client's account from $h(returnedAmount, ID_C, c_0, T_G, T_{TR}, Z_j)$. Also, the merchant is notified that the set of coins starting with c_0 is no longer valid. The merchant then sends the following message to the client.

$$\mathbf{M{\rightarrow}C:} \quad h(returnedAmount, c_0, T_G, CL_{TR}, T_{TR}, Y_i), T_{TR} \tag{9.16}$$

The client expects to receive the updated CL_{TR}, where $updatedCL_{TR} = currentCL_T + returnedAmount$, and $returnedAmount = CL_M - j_{max}$. The client compares CL_{TR} with the received one. If they are matched, she can infer the updated CL_{TR}. Later on, the client can use the bank coupon with the updated CL_{TR} to make a payment to another merchant.

9.3 Extending the Proposed Protocol to Postpaid Micropayment

In this section, we show that the proposed protocol can be extended to operate as a postpaid micropayment protocol. Considering PayWord [RS96], it is based on the assumption that the bank can accept the risk due to the overspending problem because most of its clients maintain good spending behaviors or it may want to maintain high transaction performance. Based on this assumption, we introduce an extension of the proposed protocol for postpaid micropayment transactions by presenting *Payment Initialization Protocol* and *Payment Clearing Protocol*. Sections 9.3.1 and 9.3.2 presents the protocols in details.

9.3.1 Payment Initialization Protocol

To make a payment to the merchant, the client runs the *Setup Protocol* to request for authorized total credits CL_T. However, under the same assumption as that of PayWord [RS96] which states that the client is allowed to spend up to the credit limit specified to any merchant, CL_T issued by the bank in the proposed protocol stands for the credit limit that the client is allowed to spend to each merchant instead.

The client, receiving the random number c from the *Setup Protocol*, generates a set of coins $\{c_0, ..., c_m\}$, where $m = CL_T$. The client then sends the following message to the merchant:

$$\mathbf{C}\rightarrow\mathbf{M}: \quad \{c_0, CL_T, T_G, h(c_0, ID_M, T_G, Y_i)\}_{X_i} \tag{9.17}$$

It can be seen that the merchant cannot generate this message by herself because she does not have Y_i which is shared between the client and the bank. After retrieving c_0 and CL_T, the client can make the payment to the merchant by following the *Payment Protocol* previously presented in section 9.2.5.

9.3.2 Payment Clearing Protocol

At the end of a specified period, the merchant collects necessary information and sends the following message to the bank.

$$\mathbf{M}\rightarrow\mathbf{B}: \quad ID_M, c_{j_{max}}, T_G, h(c_0, ID_M, T_G, Y_i) \tag{9.18}$$

After receiving the message, the bank calculates c_{CL_T}, where $c_{CL_T} = \{c, T_G\}$ and calculates c_0 from c_{CL_T}. The bank verifies the received c_0 with the one that it has. If they are matched, the bank calculates j_{max} and transfers the amount j_{max} to the merchant's account.

9.4 Analysis and Discussions

9.4.1 Transaction Security Properties

In this section, we show that the cryptographic technique applied to KSLv1 and KSLv2 protocols can be applied to our micropayment protocol which makes our protocol satisfy the transaction security properties stated in the Definition 3.10

of the proposed mobile payment model. The message 9.4 demonstrates how the technique works:

$$\begin{aligned}\mathbf{B{\rightarrow}M:}\quad & \{c_0, T_G, SN, CL_M, h(ID_M, SN, CL_{TR}, T_{TR}, Y_i)\}_{Z_j},\\ & h(c_0, T_G, CL_M, s, X_i), s, t\end{aligned}$$

It can seen that all transaction security properties stated in the proposed mobile payment model are satisfied as follows:

- **Party Authentication** is ensured by the symmetric encryption with the secret Z_j and by the secret Y_i shared between the client and the bank. The encryption ensures that either the bank or the merchant has originated the message, and the secret Y_i ensures that the bank is the originator of the message.

- **Transaction Privacy** is guaranteed by the symmetric encryption with the secret Z_j.

- **Transaction Integrity** is guaranteed by $h(c_0, T_G, CL_M, s, X_i)$ which is forwarded from the client. It is to ensure that c_0 will not be modified during the transaction.

- **Non-repudiation of Transactions** is ensured by $h(ID_M, SN, CL_{TR}, T_{TR}, Y_i)$ in that the bank cannot deny that it did not generate the message $\{c_0, T_G, SN, CL_M, h(ID_M, SN, CL_{TR}, T_{TR}, Y_i)\}_{Z_j}$ since the bank is the only party that holds both Z_j and Y_i.

- **Transaction Authorization** is ensured by the secrets X_i and Z_j which show that the client and the bank, respectively, have authorization on performing the transactions to the merchant.

9.4.2 Dispute Resolution

The proposed protocol offers the ability to resolve disputes among engaging parties in both direct and indirect manners. On one hand, the proposed protocol provides direct dispute resolution. Considering the message 9.5 in the *Payment Initialization Protocol*, we can prove that the bank is the originator of this message since $h(ID_M, SN, CL_{TR}, T_{TR}, Y_i)$ can be retrieved by only the client and the bank, but the client does not have the secret Z_j. Thus, the client is not the originator of the message. On the other hand, some protocol messages provide indirect dispute resolution. Considering the message 9.10 sent from the merchant to the client in the *Extra Credit Request Protocol*, although the client can generate this message by herself, she cannot modify the content of the message since it will later be detected by the bank.

9.4.3 Private Information

In any payment system, the information that is known only by relevant parties, such as secret keys, bank account information, price, or goods descriptions, is considered as *private information* [KP02]. Revealing such information offers an opportunity to perform various kinds of attacks or to trace the client's spending behavior.

In our proposed protocol, c_0 and $c_{j_{max}}$ are transmitted in encrypted forms compared to signed messages in PayWord [RS96] and cleartext in PayFair [Yen01]. Moreover, only c_j is sent from the client to the bank over the air interface. The bank can infer c_0 from $c_0 = h^n(c, T_G)$, where n stands for the current CL_{TR}, and later sends c_0 to the merchant in the message 9.4. Therefore, the secrecy of the requested amount is preserved.

9.4.4 Secret Keys

The proposed micropayment protocol deploys the set of three different secrets $\{X, Y, Z\}$ shared among engaging parties. These secrets are used as encrypting keys and the keys for keyed-hash functions that must be updated periodically or upon the requests by engaging parties. It is suggested that, to increase performance of the system and to reduce the frequency of performing key distribution process, after the secrets X, Y, and Z have been distributed, each engaging party applies the key generation technique presented in section 8.3 in order to generate the sets of session keys X_i, Y_i, and Z_j, where $i, j = 1, ..., n$, and then uses these session keys in transactions. The security analysis of the proposed key generation technique has been provided in section 8.4.

9.4.5 Trust Relationships among Engaging Parties

In PayFair [Yen01], the client is assumed to fully trust the bank in that the bank will not impersonate as the client to perform transactions to any merchants since all secrets of the client are known to the bank. Such an assumption is too strong because, practically, the bank in the protocol may be a payment gateway which is a company monitoring payment traffic and acts as a medium between the client and the merchant at the Internet side and between the client's and the merchant's financial institutions at the banking private network side. It may possibly have a conspiracy with an attacker or generate a fake the client's request by itself.

Based on the trust relationships among engaging parties stated in the the Definition 3.11 of the proposed formal model, in the proposed protocol, we state partial trust relationship between the client and the bank instead. The bank cannot generate any fake request since each message sent from the client is composed of the components or the secrets which cannot be generated or known to the bank. Therefore, the above problem is solved.

9.4.6 Performance Analysis

To demonstrate the practicability of the proposed protocol, we compare our protocol with PayWord [RS96], PayFair [Yen01], and Park *et al.*'s protocol [PBD01], in terms of transaction performance by focusing on the computation at engaging parties and the number of message passes in the protocol. Our comparison criteria is based on the proposed formal mobile payment model presented in section 3.1.11.

Party's Computation

Considering the party's computation, we mainly focus on the number of cryptographic operations applied to each engaging party. We compare our post-paid micropayment protocol with PayWord [RS96] and Park *et al.*'s protocol [PBD01] and our prepaid micropayment protocol with PayFair [Yen01]. Tables 9.1-9.4 demonstrate the comparison stated above. Note that, from the tables, n stands for the computation for generating a set of coins, p and q stand for public-key parameters which are large prime numbers, H stands for hash function, and KH stands for keyed-hash function (or MAC).

Moreover, considering the following situation, we assume that the client receives a bank coupon and performs transactions with 10 different merchants. There are 20 times that the client performs the transactions which exceed the limit specified to each merchant. Note that in PayFair [Yen01], when the requested payment order exceeds the limit specified to each merchant, the client has to repeat the entire protocol steps to get a new coupon, whereas, in the proposed protocol, the client can run the *ECR Protocol* to request the bank for extra credits. Table 9.2 demonstrates the computation at the client of each protocol under the above situation.

From Tables 9.1 and 9.2, it can be seen that, in 1 transaction, the computation at the client of PayFair is slightly lower than that of the proposed

Cryptographic Operations		Our Prepaid Protocol	PayFair
Signature generation	C	-	-
	M	-	-
	B	-	-
Signature verification	C	-	-
	M	-	-
	B	-	-
Symmetric operation	C	1	1
	M	1	-
	B	2	2
Hash function (H)	C	n	n
	M	n	n
	B	n	n
Keyed-hash function (KH)	C	4	3
	M	1	4
	B	3	5
Session-key generation	C	$2KH$	-
	M	$2KH$	-
	B	$2KH$	-
Communication overhead	C	-	-
	M	-	-
	B	-	-

Table 9.1: The number of cryptographic operations applied to our proposed prepaid protocol and PayFair [Yen01] in 1 transaction

Cryptographic Operations	Our Prepaid Protocol	PayFair
1. Signature generation	-	-
2. Signature verification	-	-
3. Symmetric operation	10	200
4. Hash function (H)	$10n$	$200n$
5. Keyed-hash function (KH)	430	600

Table 9.2: The number of cryptographic operations at client applied to the proposed prepaid protocol and PayFair, respectively, when the client performs transactions with 10 different merchants and there are 20 times that she spends the coins over the limit specified to each merchant

prepaid protocol. This is because our protocol deploys the session key generation technique (presented in section 8.2) that produces limited-used secret keys in order to enhance the security of the system, whereas PayFair deploys long-term, reusable, shared secrets among parties. The concern about security in the deployment of reusable, long-term, shared secrets has been discussed in section 2.6. Moreover, the proposed protocol satisfies non-repudiation property, whereas PayFair does not. In higher number of transactions, it is clear that our prepaid protocol has lower number of client's computation than PayFair due to the feature of our proposed *ECR Protocol*.

Cryptographic Operations		Our Postpaid Protocol	PayWord	Park *et al.*'s Protocol
Signature generation	C	-	1	-
	M	-	-	-
	B	-	1	-
Signature verification	C	-	1	-
	M	-	2	-
	B	-	2	-
Symmetric operation	C	2	-	-
	M	1	-	-
	B	1	-	-
Hash function (H)	C	n	n	n
	M	n	n	n
	B	n	n	n
Keyed-hash function (KH)	C	2	-	3
	M	-	-	2
	B	2	-	2
Session-key generation	C	$2KH$	-	-
	M	$2KH$	-	-
	B	$2KH$	-	-
Communication overhead	C	-	$2\|p\|+2\|q\|$	-
	M	-	$2\|p\|+2\|q\|$	-
	B	-	$3\|p\|+3\|q\|$	-

Table 9.3: The number of cryptographic operations applied to the proposed postpaid protocol, PayWord [RS96], and Park *et al.*'s protocol [PBD01] in 1 transaction

Comparing the proposed postpaid micropayment protocol to the existing protocols, it can be seen in Table 9.3 that our protocol has lower client's computation than PayWord [RS96]. This is because, in our protocol, only symmetric-key operations and hash functions are applied, whereas PayWord deploys public-key operations. Also, the communication load of PayWord is higher than that of our protocol. This infers that the proposed postpaid protocol has better performance than PayWord. However, as shown in Table 9.3, the client's computation of Park *et al.*'s protocol [PBD01] is slightly lower than that of our protocol because Park *et al.*'s protocol does not deploy any symmetric-key operations whereas our protocol does. As previously discussed in section 2.4.3, the lack of symmetric-key operations in Park *et al.*'s protocol leads to the problem of revealing payment information (c_0 and c_j). Moreover, our protocol deploys limited-used session keys generated from the proposed session key generation technique presented in section 8.2 that enhances security of the system, whereas Park *et al.*'s protocol deploys reusable, long-term secret keys. It can be noted that the deployment of slightly higher computation at the client in our protocol enhances the security of the system considerably. Also, the difference in terms of the number of keyed-hash functions does not greatly affect performance of the system.

Cryptographic Operations	Our Postpaid Protocol	PayWord	Park *et al.*'s Protocol
1. Signature generation	-	200	-
2. Signature verification	-	200	-
3. Symmetric operation	200	-	-
4. Hash function (H)	$200n$	$200n$	$200n$
5. Keyed-hash function (KH)	200	-	600

Table 9.4: The number of cryptographic operations at client applied to the proposed protocols, PayWord, and Park *et al.*'s protocol, respectively, when the client performs transactions with 10 different merchants and there are 20 times that she spends the coins over the limit specified to each merchant

In addition, it can be seen in Table 9.4 that, at higher number of transactions, the client's computation of the proposed protocol is lower than that of PayWord and Park *et al.*'s protocol. As a result, comparing in terms of the party's computation, our proposed protocols have better performance than the existing micropayment protocols (referred to the proposed mobile payment model in section 3.1.11).

The Number of Message Passes

The number of message passes affects the performance of payment transactions. Higher number of messages passes results in longer time consumption to complete the transaction. Moreover, longer processing time causes a mobile user to stay online for longer period which costs higher connection cost. This may not be acceptable by users. Figure 9.2 demonstrates the number of message passes in the *Payment Initialization Protocol* of the proposed protocols and the existing protocols.

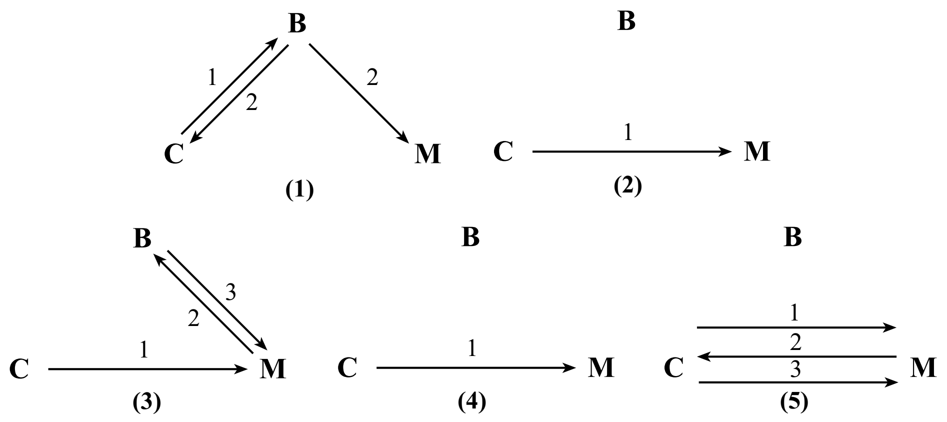


Figure 9.2: The number of messages passes in *Payment Initialization Protocol* of the proposed (1) prepaid and (2) postpaid protocols, (3) PayFair, (4) PayWord, and (5) Park *et al.*'s protocol, respectively

Compared to PayWord [RS96], although the number of message passes of the proposed postpaid protocol is equal to that of PayWord [RS96] as shown in Figure 9.2 and Table 9.5, from Table 9.3, the proposed protocol has lower party's computation at engaging parties than that of PayWord. Therefore, the proposed protocol has better performance than PayWord.

Phases	Our Protocols		PayWord	PayFair	Park's Protocol
	prepaid	postpaid			
Setup	2	2	2	2	NS
Payment Initialization	2	1	1	3	3
Payment Clearing	-	1	1	-	1

Table 9.5: The number of message passes per transaction of our protocols, PayWord, and PayFair, respectively

Compared to Park *et al.*'s protocol [PBD01], from Figure 9.2, it can be seen that the proposed postpaid protocol has less number of message passes in the *Payment Initialization Protocol*. Although Table 9.5 shows that the number of message passes per transaction of Park *et al.*'s protocol is equal to ours, Park *et al.* did not mention the details of the *Setup Protocol* which describes how the client requests the bank to perform transactions in the protocol (NS in Table 9.5 stands for not specified.). It can be argued that this process takes at least two message passes: one is that the client requests the bank for authorization and the other is that the client receives the response of the request from the bank. Therefore, our protocol has less number of message passes per transaction than that of Park *et al.*'s protocol which results in higher transaction performance.

Compared to PayFair [Yen01], the proposed prepaid protocol has less number of message passes. Moreover, in PayFair, the client needs to request the bank for issuing a new coupon and generate a new set of coins every time she runs out of credits, whereas the coupon in the proposed protocol is issued only once and can be used to make payment with many merchants. This greatly reduces the computational load at the client side. These features result in better performance than PayFair.

9.5 Summary

In this chapter, we have pointed out the problems occurred in existing micropayment protocols when applying them to wireless environments due to poor performance and security flaws. To enable a secure and practical wireless micropayment, we have proposed a prepaid micropayment protocol that applies the symmetric cryptographic technique previously applied to KSLv1 and KSLv2 protocols to solve the above problems. Our proposed protocol offers full non-repudiation in that a merchant and a bank cannot impersonate as a client to perform a payment transaction in the protocol. Achieving this property provides higher security to the payment system. Moreover, we reduce the level of trust relationship between the client and the bank in that the bank is partially trusted by the client. In addition, the private information of all engaging parties is well-protected. These offer more practicability and provide higher security to our protocol compared to the existing protocols.

Moreover, we have shown that the proposed protocol is able to perform postpaid micropayment transactions by introducing its extension which does not affect the existing protocol structure. This results in a general micropayment protocol that engaging parties are able to perform both kinds of micropayment under the same framework.

We also performed performance analysis to demonstrate that the proposed protocol has better performance than PayWord [RS96], Park *et al.*'s protocol, and PayFair [Yen01] which results in more applicable to limited capability mobile devices.

Chapter 10

Conclusion

Mobile payment is generally known as interactions among engaging parties regarding fund transfer over a wireless network through mobile devices. Mobile payment is not only about purchasing goods or services over a wireless network, but also includes fund transfer among individuals. When we started our research, although a number of literature about mobile payment have been discussed, none of them has modelled it formally. Moreover, several mobile payment frameworks had tried to reduce the computational load at engaging parties in order to overcome limitations of wireless environments and to enable payment transactions through limited capability mobile devices, but they did not fully consider transaction security properties which are important factors for payment transactions. This thesis provides a formal model for mobile payment and proposes practical and secure mobile payment solutions for both account-based and token-based payment transactions. In this chapter, we highlight the essential contributions of the thesis by pointing out existing problems and how they have been solved.

To provide concrete explanation for mobile payment, we proposed a formal model for a practical and secure mobile payment system. In this model, we defined engaging parties in the system and the interactions among them. We also defined goals ($Goals$) and requirements to complete a payment transaction in each party's point of view (PR). Achieving such goals and requirements ensure that all parties have completed the transactions related to them. In terms of the security of the system, we stated transaction security properties ($TSec$) and trust relationships among engaging parties ($Trust$) to be satisfied by a mobile payment system. In particular, a payment system that satisfies such properties can have secure payment transactions.

To enable practical mobile payment transactions, we focused on transaction performance. In the model, we defined the term "*acceptable transaction performance*" (ATP) as the transaction performance that is acceptable by all engaging parties in the payment system. A payment system that satisfies acceptable transaction performance is likely to be chosen as a real world application. As a result, the proposed model can be used as a major guideline to evaluate security and practicability of mobile payment systems. That is, to be considered as a practical and secure mobile payment system, a payment system must satisfy $Goals$, PR, $TSec$, $Trust$, and ATP mentioned above.

We have discussed existing frameworks to enable mobile payment, namely proxy-based, agent-based, and non proxy-based frameworks, in terms of transaction performance and security and found that the payment systems based on each existing framework has their own advantages and disadvantages. In chapter 4, we proposed a mobile payment framework which not only incorporates advantages of the existing frameworks, but also solves their existing problems. In our framework, we deployed a mobile agent to reduce the connection time which leads to the reduction of connection cost. Moreover, a proxy server is operated by a partially trusted party as the environment where the mobile agent can perform high computational operations on behalf of the client. We applied the proposed framework to SET protocol [Mas97] and compared the SET system based on our framework with the SET systems based on the existing frameworks in terms of security and performance. The results obtained from our analysis have shown that our SET system offers higher security than existing agent-based mobile SET systems and has less computation at the client side compared to the proxy-based SET systems. Thus, in terms of transaction performance, the proposed SET system is more likely to satisfy ATP stated in the proposed model than the SET system based on the existing frameworks. As the SET protocol itself does not satisfy $Trust$ stated in the proposed formal model, in section 4.4.2, we provided a suggestion on how to modify the SET protocol to achieve $Trust$. Note that, with our framework, SET, a high computational fixed-network payment protocol, can have practical and secure mobile payment transactions. Therefore, applying a non proxy-based mobile payment protocol which deploys lightweight operations to the proposed framework will result in the mobile payment transactions with higher transaction performance.

As previously discussed in chapter 2, existing non proxy-based mobile payment protocols lack transaction security properties due to the fact that some lightweight cryptographic operations lack the important security properties viz., non-repudiation. To overcome this limitation, we proposed a cryptographic technique that deploys lightweight symmetric-key operations, yet satisfies all transaction security properties ($TSec$) stated in the proposed formal model. Note that normally, all transaction security properties are satisfied only by public-key operations. Applying our technique to a protocol results in not only security enhancement, but also increased transaction performance. We applied the proposed technique to secure both account-based and token-based mobile payment. Particularly, in this thesis, we mainly focused our consideration on account-based mobile payment, yet discussed the usability of the proposed technique to secure token-based mobile payment.

To enable account-based mobile payment, we proposed two non proxy-based credit-card payment protocols called KSLv1 and KSLv2. In KSLv1 protocol, we minimized the client's computation by deploying the proposed cryptographic technique only at the client side, whereas the merchant and the payment gateway perform public-key operations. Applying the proposed technique to the protocol makes it satisfy transaction security properties stated in our formal model. We implemented a system that follows KSLv1 protocol descriptions where a client performs payment transactions on a PDA over a wireless LAN network. The results have shown that KSLv1 is capable of being implemented as a real world application. We extended the possible use of the proposed cryptographic technique by proposing KSLv2 protocol, whereby all parties are not required to perform public-key operations, based on the proposed technique. We compared KSLv1 and KSLv2 with existing non proxy-based payment protocols in terms of security and performance. The results obtained from our analysis have shown that our protocols satisfy all necessary security properties including the trust relationships among engaging parties ($Trust$) stated in the proposed model and have higher transaction performance than existing protocols. In other words, the proposed protocols are more likely to achieve ATP than the existing protocols. As a result KSLv1 and KSLv2 operate well without any proxy required, yet have transaction performance enhancement when they operate under our proposed framework.

To show that the proposed framework and protocols satisfy the proposed mobile payment model, we analyzed them on accountability. Generally, accountability in electronic commerce protocols is the property that all engaging parties must by able to prove their responsibility regarding the payment transaction relevant to them. Accountability is considered as a high-level security property that covers all transaction security properties ($TSec$) stated in the proposed formal model in that, to satisfy the accountability, all transaction security properties must be satisfied. In addition, from the Definitions 3.7 and 3.1.8 of the proposed model, we have shown that the goals of engaging parties ($Goals$) and the party's requirements for payment transactions (PR) can be formalized by the accountability. Therefore, we can prove that a payment system satisfies $TSec$, $Goals$, and PR stated in the proposed model by analyzing it on the accountability property.

We examined existing accountability logics and found that they are inadequate to analyze mobile payment protocols where the accountability can be satisfied by not only asymmetric-key messages, but also symmetric-key ones. We then proposed an accountability logic which is able to analyze mobile payment protocols efficiently. We presented the analyses of the proposed framework and protocols on accountability. The results from the analysis have shown that they satisfy the accountability property. This infers that they satisfy $TSec$, $Goals$, and PR, stated in the proposed mobile payment model. As previously shown that the proposed framework and protocols satisfy $Trust$ and ATP, it is obvious that the payment systems based on the proposed framework and protocols are considered as practical and secure mobile payment systems according to the Definition 3.15 of the proposed formal model.

We examined the deployment of shared secrets in symmetric-key cryptosystems, especially in the payment system scenario. We found that the deployment of long-term, reusable, shared secrets, such as credit-card information, during transactions is vulnerable to attacks. Even though such secrets are enciphered, this offers an opportunity for an attacker to intercept the ciphered message and try to decipher it. Moreover, we found that existing techniques to enhance the security of shared secret keys still rely on long-term shared keys. We then introduced a technique to generate session keys that does not rely on any long-term secrets, but by a dynamically chosen set of preference keys. Our technique has been proven to be secure against various kinds of attacks. Furthermore, we have demonstrated the practical usefulness of our technique be applying it to our mobile payment protocols, KSLv1 and KSLv2.

In addition to account-based mobile payment protocols, we investigated existing token-based mobile payment protocols by focusing on micropayment protocols. Most micropayment protocols are applicable for wireless environments due to their low computation at mobile users. Applying expensive cryptographic operations may not considered worthwhile. Therefore, several micropayment protocols were designed to deploy small number of lightweight operations. However, the nature of such operations leads to the lack of transaction security properties and the exposure of private information of engaging parties. We then introduced a prepaid micropayment protocol that solves the above problems. Our protocol deploys the cryptographic technique applied to KSLv1 and KSLv2 protocols so that it satisfies transaction security properties stated in our mobile payment model. We then presented an extension of the proposed protocol to perform postpaid micropayment transactions. This results in a general micropayment protocol in that users can perform both prepaid and postpaid micropayment under the same framework.

In summary, this thesis makes the following contributions:

1. A formal model for a practical and secure mobile payment system;
2. A framework for mobile payment that enables mobile payment capability to any kinds of payment protocols;
3. Non proxy-based mobile payment protocols for both account-based and token-based payment that operate well in wireless environments;
4. A formal logic that is able to analyze accountability of electronic commerce protocols;
5. A session key generation technique that enhances security of shared secret transfer in Internet transactions.

Each of the proposed research above suggests future research directions. Particularly, the development of lightweight cryptographic algorithms might be able to increase transaction security and performance while applying them to the proposed mobile payment protocols. The proposed accountability logic not only can analyze mobile payment protocols, but also any kind of electronic commerce protocols. We can also see the potential contribution of the proposed key generation technique in applying it to other kinds of Internet transactions.

Bibliography

[AG03] L. Antovski and M. Gusev. M-payments. In *Proceedings of the 25th International Conference on Information Technology Interfaces 2003*, pages 95–100, Cavtat, Croatia, June 16-19, 2003.

[AHS98] N. Asokan, E.V. Herreweghen, and M. Steiner. Towards a framework for handling disputes in payment systems. In *Proceedings of the Third USENIX Workshop on Electronic Commerce 1998*, pages 187–202, 1998.

[Ahu96] V. Ahuja. *Secure Commerce on the Internet.* Academic Press, 1996.

[AM00] R. Sai Anand and C.E. Veni Madhavan. An online, transferable e-cash payment system. *Lecture Notes in Computer Science*, 1977:93–103, 2000.

[AMS97] R. Anderson, C. Manifavas, and C. Sutherland. Netcard - a practical electronic cash system. *Lecture Notes in Computer Science*, 1189:49–57, 1997.

[And98] M.M. Anderson. The electronic check architecture version 1.0.2. *Financial Services Technology Consortium*, 1998.

[APASW98] J.L. Abad-Peiro, N. Asokan, M. Steiner, and M. Waidner. Designing a generic payment service. *IBM Systems Journal*, 37(1):72–88, 1998.

[BAN90] M. Burrows, M. Abadi, and R. Needham. A logic of authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, February, 1990.

[BGH$^+$00] M. Bellare, J.A. Garay, R. Hauser, A. Herzberg, H. Krawczyk, M. Steiner, G. Tsudik, E.V. Herreweghen, and M. Waidner. Design, implementation, and deployment of the iKP secure electronic payment system. *IEEE Journal of Selected Areas in Communications*, 18(4):611–627, April, 2000.

[BMN01] C. Boyd, P. Montague, and K. Nguyen. Elliptic curve based password authenticated key exchange protocols. *Lecture Notes in Computer Science*, 2119:487–501, 2001.

[BP98] C. Boyd and D.G. Park. Public key protocols for wireless communications. In *Proceedings of the International Conference on Information Security and Cryptology 1998*, pages 47–57, 1998.

[Cer03] Certicom.com. The next generation of cryptography: Public key sizes for AES. *Code and Cipher: Certicom's Bulletin of Security and Cryptography*, 1(1), 2003. `http://www.certicom.com/`.

[Cim02] S. Cimato. Design of an authentication protocol for GSM javacards. *Lecture Notes in Computer Science*, 2288:355–368, 2002.

[Com04] Commonwealth Bank Group. *EFTPOS*, 2004. `http://www.commbank.com.au/` Last accessed January 30, 2005.

[Dat04] Dataquest.com. *Gartner Press Room: PDAs Statistics*, 2004. `http://www.dataquest.com/press_gartner/quickstats/pe-rsonal.html` Last accessed January 30, 2005.

[DK01] A.R. Dani and P.R. Krishna. An e-check framework for electronic payment systems in the web based environment. *Lecture Notes in Computer Science*, 2115:91–100, 2001.

[DZ03] Y. Dai and L. Zhang. A security payment scheme of mobile e-commerce. In *Proceedings of the International Conference on Communication Technology 2003*, volume 2, pages 949–952, April 9-11, 2003.

[ePa04a] ePaymentnews. *Forrester's mCommerce Sales Predictions 2001-2005*, 2004. `http://www.epaynews.com/statistics/mcommstats.html#42` Last accessed January 30, 2005.

[ePa04b] ePaymentnews. *WAP Handset Share of All Handset Sales, 2000-2006*, 2004. `http://www.epaynews.com/statistics/mcommstats.html#16` Last accessed January 30, 2005.

[FAB02] A. Fourati, H. Ayed, and A. Benzekri. A SET based approach to secure the payment in mobile commerce. In *Proceedings of the 27th Annual IEEE Conference on Local Computer Networks 2002*, pages 136–137, November 6-8, 2002.

[FKK96] A.O. Freier, P. Karlton, and P. Kocher. *The SSL Protocol Version 3.0: Internet Draft*, March 1996. `http://wp.netscape.com/eng/ssl3/ssl-toc.html`.

[GCW01] H. Gao, W. Changjie, and Y. Wang. A secure electronic payment protocol based on bank account. In *Proceedings of the International Conference on Info-tech and Info-net 2001*, volume 6, pages 30–34. IEEE Computer Society, October 29 - November 1, 2001.

[Gho98] A.K. Ghosh. *E-Commerce Security: Weak Links, Best Defenses.* John Wiley and Sons, 1998.

[GMAG95] S. Glassman, M. Manasse, M. Abadi, and P. Gauthier. The Millicent protocol for inexpensive electronic commerce. In *Proceedings of the Fourth International World Wide Web Conference 1995*, pages 603–618, 1995.

[Her01] E.V. Herreweghen. Non-repudiation in SET: Open issues. *Lecture Notes in Computer Science*, 1962:140–156, 2001.

[HMM02] G. Horn, K.M. Martin, and C.J. Mitchell. Authentication protocols for mobile network environment value-added services. *IEEE Transactions on Vehicular Technology*, 51(2):383–392, 2002.

[HP98] G. Horn and B. Preneel. Authentication and payment in future mobile systems. In *Proceedings of the Fifth European Symposium on Research in Computer Security 1998*, pages 277–293, Belgium, 1998.

[HSW96] R. Hauser, M. Steiner, and M. Waidner. Micro-payments based on iKP. IBM Research Report RZ 2791, IBM Research Division, December 1996.

[IDC] IDC. US mobile commerce revenues, 2001-2007. `http://www.epaynews.com/statistics/mcommstats.html`, Last accessed January 30, 2005.

[ITF] ITFacts.biz. *Smartphones account for 3% of wireless handset market.* `http://www.itfacts.biz/index.php?id=C0_9_1` Last accessed January 30, 2005.

[JRFH02] L. Jin, S. Ren, L. Feng, and G.Z. Hua. Research on WAP clients supports SET payment protocol. *IEEE Communications*, 9(1):90–95, February, 2002.

[Kai96] R. Kailar. Accountability in electronic commerce protocols. *IEEE Transactions on Software Engineering*, 22(5):313–328, May, 1996.

[KBC97] H. Krawczyk, M. Bellare, and R. Canetti. HMAC: Keyed-hashing for message authentication. *RFC 2104*, 1997.

[KLK$^+$02] M.A. Kim, H.K. Lee, S.W. Kim, W.H. Lee, and E.K. Kang. Implementation of anonymity-based e-payment system for m-commerce. In *Proceedings of the International Conference on Communications, Circuits and Systems and West Sino Exposition 2002*, pages 363–366, 2002.

[KLS05] S. Kungpisdan, P.D. Le, and B. Srinivasan. A limited-used key generation scheme for internet transactions. *Lecture Notes in Computer Science*, 3325:302–316, 2005.

[KN98] V. Kessler and H. Neumann. A sound logic for analyzing electronic commerce protocols. In *Proceedings of the European Symposium on Research in Computer Security 1998*, pages 345–360, 1998.

[KP02] S. Kungpisdan and Y. Permpoontanalarp. Practical reasoning about accountability in electronic commerce protocols. *Lecture Notes in Computer Science*, 2288:268–284, 2002.

[Kra99] H. Krawczyk. Blinding of credit card numbers in the SET protocol. *Lecture Notes in Computer Science*, 1648:17–28, 1999.

[KSL03a] S. Kungpisdan, B. Srinivasan, and P.D. Le. Lightweight mobile credit-card payment protocol. *Lecture Notes in Computer Science*, 2904:295–308, 2003.

[KSL03b] S. Kungpisdan, B. Srinivasan, and P.D. Le. A practical framework for mobile SET payment. In *Proceedings of the IADIS International E-society Conference 2003*, pages 321–328, Lisbon, Portugal, June 3-6, 2003.

[KSL04a] S. Kungpisdan, B. Srinivasan, and P.D. Le. Accountability logic for mobile payment protocols. In P. K. Srimani, editor, *Proceedings of the International Conference on Information Technology: Coding and Computing 2004*, volume 1, pages 40–44, Las Vegas, USA, April 5-7, 2004. IEEE Computer Society.

[KSL04b] S. Kungpisdan, B. Srinivasan, and P.D. Le. An integrated framework for payment transactions in wireless environments. In *Proceedings of the Second International Conference on Information and Communication Technologies 2004*, pages 158–168, November 18-19, 2004.

[KSL04c] S. Kungpisdan, B. Srinivasan, and P.D. Le. A secure account-based mobile payment protocol. In P. K. Srimani, editor, *Proceedings of the International Conference on Information Technology: Coding and Computing 2004*, volume 1, pages 35–39, Las Vegas, USA, April 5-7, 2004. IEEE Computer Society.

[KSL04d] S. Kungpisdan, B. Srinivasan, and P.D. Le. A secure prepaid wireless micropayment protocol. In *Proceedings of the Second International Workshop on Security in Information Systems 2004*, pages 104–113, Porto, Portugal, April 13-14, 2004.

[KSL04e] S. Kungpisdan, B. Srinivasan, and P.D. Le. A secure wireless prepaid micropayment protocol with extension to postpaid micropayment. In *Proceedings of the Sixth International Conference on Information Integration and Web Based Applications and Services 2004*, pages 517–526, Jakarta, Indonesia, September 27-29, 2004.

[Lam81] L. Lamport. Password authentication with insecure communication. *Communications of the ACM*, 24(11):770–772, 1981.

[LCGS03] K. Lam, S. Chung, M. Gu, and J. Sun. Lightweight security for mobile commerce transactions. *Computer Communications*, 26:2052–2060, 2003.

[Leg04] Legion of the Bouncy Castle. *Bouncy Castle Crypto APIs*, 2004. `http://www.bouncycastle.org/index.html` Last accessed January 30, 2005.

[LZ04] Y. Li and X. Zhang. A security-enhanced one-time payment scheme for credit card. In *Proceedings of the International Workshop on Research Issues on data Engineering: Web Services for E-Commerce and E-Government Applications 2004*, pages 40–47, 2004.

[Mas97] Mastercard and Visa. *SET Protocol Specifications Book 1-3*, 1997.

[MB01] L.M. Marvel and C.G. Boncelet. Authentication for low power systems. In *Proceedings of the IEEE Military Communications Conference 2001*, volume 1, pages 135–138, 2001.

[MS98] C. Meadows and P. Syverson. A formal specification of requirements for payment transactions in the SET protocol. *Lecture Notes in Computer Science*, 1465:122–140, 1998.

[MV02] Y. Mu and V. Varadharajan. An efficient internet credit card scheme from the weil pairing. In *Proceedings of the Third IEEE International Symposium on Electronic Commerce 2002*, pages 58–62, October 18-19, 2002.

[MvOV97] A.J. Menezes, P.C. van Oorschot, and S.A. Vanstone. *Handbook of Applied Cryptography.* CRC Press Inc., 1997.

[Pay01] Paybox.net. *Paybox Security Whitepaper: Business and Technical Information Regarding the Security at Paybox Version 1.4*, November, 2001. `http://www.paybox.net`.

[PBD01] D.G. Park, C. Boyd, and E. Dawson. Micropayments for wireless communications. *Lecture Notes in Computer Science*, 2015:192–205, 2001.

[PRRJ03] N. Potlapally, S. Ravi, A. Raghunathan, and N. Jha. Analyzing the energy consumption of security protocols. In *Proceedings of the 2003 International Symposium on Low Power Electronics and Design 2003*, pages 30–35, 2003.

[RdS98] A. Romao and M. da Silva. An agent-based secure internet payment system. *Lecture Notes in Computer Science*, 1402:80–93, 1998.

[Rea89] R.J. Read. EFTPOS: electronic funds transfer at point of sale. *Electronics and Commmunication Engineering Journal*, 1(6):263–270, November/December, 1989.

[RRP02] S. Ravi, A. Raghunathan, and N. Potlapally. Securing wireless data: System architecture challenges. In *Proceedings of the 15th International Symposium on System Synthesis 2002*, pages 195–200, 2002.

[RS96] R.L. Rivest and A. Shamir. PayWord and Micromint: Two simple micropayment schemes. *Cryptobytes*, 2(1):7–11, 1996.

[RSA93a] RSA Laboratories. *PKCS#1: RSA Encryption Standard. Version 1.5*, November, 1993.

[RSA93b] RSA Laboratories. *PKCS#7: Cryptographic Message Syntax Standard. Version 1.5*, November, 1993.

[RW02] A.D. Rubin and R.N. Wright. Off-line generation of limited-use credit card numbers. *Lecture Notes in Computer Science*, 2339:196–209, 2002.

[SASR01] C. Sanchez-Avila and R. Sanchez-Reillol. The Rijndael block cipher (AES proposal): a comparison with DES. In *Proceedings of the 35th IEEE International Carnahan Conference on Security Technology 2001*, pages 229–234. IEEE Computer Society, October, 2001.

[SGK02] S. Schwiderski-Grosche and H. Knopse. Secure mobile commerce. *Electronics and Communication Engineering Journal*, pages 228–238, October, 2002.

[Sha02] A. Shamir. SecureClick: A web payment system with disposable credit card numbers. *Lecture Notes in Computer Science*, 2339:232–242, 2002.

[Sun05] Sun Microsystems. *Java 2 Platform Micro Edition*, Last accessed January 30, 2005 2005. `http://java.sun.com/j2me/index.jsp`.

[SV97] J. Stern and S. Vaudenay. SVP: a flexible micropayment scheme. In *Proceedings of the Financial Cryptography 1997*, pages 161–171, 1997.

[Tha00] D.V. Thanh. Security issues in mobile e-commerce. *Lecture Notes in Computer Science*, 1875:467–476, 2000.

[TKL04] B.T.S. Toh, S. Kungpisdan, and P.D. Le. KSL protocol: Design and implementation. In *Proceedings of the 2004 IEEE Conference on Cybernetics and Intelligent Systems*, pages 544–549, Singapore, December 1-3, 2004.

[TR02] R. Temple and J. Regnault. *Internet and Wireless Security.* The Institution of Electrical Engineers, London, UK, 2002.

[Tsu92] G. Tsudik. Message authentication with one-way hash function. *ACM Communications Review*, 22(5):29–38, 1992.

[VIS02] VISA International Service Association. *3-D Secure version 1.0.2*, September 26, 2002.

[VIS04] VISA International. *Visa Electron*, 2004. `http://international.visa.com/ps/products/electron/`.

[WAP01] WAP Forum. *Wireless Transport Layer Security Version 06-Apr-2001*, April, 2001. `http://www.wapforum.org`.

[WAP02] WAP Forum. *Wireless Application Protocol Version 2.0*, January, 2002. `http://www.wapforum.org`.

[WC01] D.S. Wong and A.H. Chan. Efficient and mutually authentication key exchange for low power computing devices. *Lecture Notes in Computer Science*, 2248:272–289, 2001.

[WLY99] X.F. Wang, K.Y. Lam, and X. Yi. Secure agent-mediated mobile payment. *Lecture Notes in Artificial Intelligence*, 1599:162–173, 1999.

[WSZ01] K. Wrona, M. Schuba, and G. Zavagli. Mobile payments - state of the art and open problems. *Lecture Notes in Computer Science*, 2232:88–100, 2001.

[Yee94] B.S. Yee. *Using Secure Coprocessor*. PhD thesis, Carnegie Mellon University, 1994.

[Yen01] S. Yen. PayFair: A prepaid internet micropayment scheme ensuring customer fairness. *Proceedings of the IEE Computers and Digital Techniques 2001*, 148(6):207–213, 2001.

[YL01] P.L. Yu and C.L. Lei. An user efficient fair e-cash scheme with anonymous certificates. In *Proceedings of IEEE Region 10th International Conference on Electrical and Electronic Technology 2001*, volume 1, pages 74–77, August 19-22, 2001.

[YSWO00] X. Yi, C.K. Siew, X.F. Wang, and E. Okamoto. A secure agent-based framework for internet trading in mobile computing environments. *Distributed and Parallel Databases*, 8:85–117, 2000.

[ZK02] H. Zheng and C. KeFei. Electronic payment in mobile environment. In *Proceedings of the 13th International Workshop on Database and Expert Systems Applications 2002*, pages 413–417, September 2-6, 2002.

[ZWCY02] F. Zhu, D.S. Wong, A.H. Chan, and R. Ye. Password authenticated key exchange based on RSA for imbalanced wireless networks. *Lecture Notes in Computer Science*, 2433:150–161, 2002.

Appendix A

Detailed Analysis of KSLv1 and KSLv2 on Accountability

A.1 Analyzing KSLv1 on Accountability

We analyze KSLv1 protocol on accountability by using the KSL logic introduced in chapter 7. As the goals **G1-G3** have been analyzed and discussed in section 7.6.3, in this section, the analysis of the goals **G4-G6** of KSLv1 is presented.

A.1.1 Proving the Goal G4 of KSLv1

G4: *C believes C CanProve* (
I authorized debit$(I, C, Price, Date)$) *to V*

Consider **Step 6** of KSLv1,

M→C: $\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$

It can be transformed into:

C *sees* $\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$

Details of the Proof

C *sees* $\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$ *(1)*

1, C1, M: C *believes* C *sees* $\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$ *(2)*

2, H1, M: C *believes* C *has* $\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}$ *(3)*

3, A8, H2, H4, M:

C *believes* C *has* ($\{\{h(OI), Yes/No\}_{Y_i}\}_{X_{i+1}}, X_{i+1}$) $\wedge$

C *believes* C *believes* ($C \overset{X_{i+1}}{\longleftrightarrow} M$)

$\rightarrow$ C *believes* C *has* $\{h(OI), Yes/No\}_{Y_i}$ *(4)*

A10, 4, A8, H2, A5, P2, M:

C *believes* V*-is-external-party* $\wedge$

C *believes* C *has* ($\{h(OI), Yes/No\}_{Y_i}, Y_i$) $\wedge$

C *believes* (

V *has* ($\{h(OI), Yes/No\}_{Y_i}, Y_i$) $\wedge$

$(h(OI), Yes/No) = \{\{h(OI), Yes/No\}_{Y_i}\}_{Y_i}$

$\rightarrow$ V *believes* Y_i*-is-decrypting-key-for-*$\{h(OI), Yes/No\}_{Y_i}$)

$\rightarrow$ C *believes* C *CanProve* (

Y_i*-is-decrypting-key-for-*$\{h(OI), Yes/No\}_{Y_i}$) *to* V *(5)*

A10, 4, A1, P2, M:

C *believes* V*-is-external-party* $\wedge$

C *believes* C *has* $(h(OI), Yes/No)$ $\wedge$

C *believes* (

V believes ($\boldsymbol{C} \overset{(h(OI),Yes/No)}{\longleftrightarrow} \boldsymbol{I}$) $\wedge$ − − − − − − − − − − − − − − − − − ***(b)***

V *has* $(h(OI), Yes/No)$ $\rightarrow$ V *believes* ($C \overset{(h(OI),Yes/No)}{\longleftrightarrow} I$))

$\rightarrow$ C *believes* C *CanProve* ($C \overset{(h(OI),Yes/No)}{\longleftrightarrow} I$) *to* V *(6)*

We can see that the statement *(b)* contradicts the assumption **A10** that '$\neg C$ *believes* V *believes* ($C \overset{(h(OI),Yes/No)}{\longleftrightarrow} I$)'. Thus, the proof fails.

4, H2, A8, 5, 6, P3', M:

C *believes* C *has* $(\{h(OI), Yes/No\}_{Y_i}, Y_i) \wedge$

C *believes* C *believes* $(C \xleftrightarrow{Y_i} I) \wedge$

C *believes* C *CanProve* (

Y_i*-is-decrypting-key-for-*$\{h(OI), Yes/No\}_{Y_i}$) *to* $V \wedge$

C believes C CanProve $(C \xleftrightarrow{(h(OI), Yes/No)} I)$ ***to V*** $-----$ ***from (6)***

$\rightarrow$ C *believes* C *CanProve* (I *says* $(h(OI), Yes/No, ID_C)$) *to* V *(7)*

Although the failure of proving the goal **G4** can lead to a dispute between the client and the issuer, KSLv1 provides indirect dispute resolution property with the same reason as that of proving the goal **G2** of KSLv1 previously presented in section 7.6.3.

A.1.2 Proving the Goal G5 of KSLv1

G5: *PG believes PG CanProve* (

M authorized credit$(M, PG, Price, Date)$) *to* V

Consider **Step 3** of KSLv1,

M→PG: $\{\{MAC[(Price, h(OI), ID_M), Y_i], Price\}_{K_{PG}}, h(OI), i,$
$TID, ID_C, ID_I\}_{K_M^{-1}}$

It can be transformed into:

PG sees $\{\{MAC[(Price, h(OI), ID_M), Y_i], Price\}_{K_{PG}}, h(OI), i,$
$TID, ID_C, ID_I\}_{K_M^{-1}}$

Details of the Proof

$$PG \text{ sees } \{\{MAC[(Price, h(OI), ID_M), Y_i], Price\}_{K_{PG}}, h(OI), i, TID, ID_C, ID_I\}_{K_M^{-1}} \quad (1)$$

1, C1, M:

$$PG \text{ believes } PG \text{ sees } \{\{MAC[(Price, h(OI), ID_M), Y_i], Price\}_{K_{PG}}, h(OI), i, TID, ID_C, ID_I\}_{K_M^{-1}} \quad (2)$$

2, H1, M:

$$PG \text{ believes } PG \text{ has } \{\{MAC[(Price, h(OI), ID_M), Y_i], Price\}_{K_{PG}}, h(OI), i, TID, ID_C, ID_I\}_{K_M^{-1}} \quad (3)$$

A1, H2, M: $PG \text{ believes } PG \text{ has } K_M$ *(4)*

A7, A3, K: $PG \text{ believes } PG \text{ believes } (\xrightarrow{K_M} M)$ *(5)*

3, 4, 5, H6, M:

$$PG \text{ believes } PG \text{ has } (\{MAC[(Price, h(OI), ID_M), Y_i], Price\}_{K_{PG}}, h(OI), i, TID, ID_C, ID_I) \quad (6)$$

4, 6, H2, M:

$$PG \text{ believes } PG \text{ has } (\{MAC[(Price, h(OI), ID_M), Y_i], Price\}_{K_{PG}}, h(OI), i, TID, ID_C, ID_I, K_M) \quad (7)$$

A7, A3, P2, M: $PG \text{ believes } PG \text{ CanProve } (\xrightarrow{K_M} M) \text{ to } V$ *(8)*

3, 7, 8, P3, M:

$$PG \text{ believes } PG \text{ CanProve } (M \text{ says } (\{MAC[(Price, h(OI), ID_M), Y_i], Price\}_{K_{PG}}, h(OI), i, TID, ID_C, ID_I) \text{ to } V \quad (9)$$

9, P6, M:

$$PG \text{ believes } PG \text{ CanProve } (M \text{ says } \{MAC[(Price, h(OI), ID_M), Y_i], Price\}_{K_{PG}}) \text{ to } V \quad (10)$$

A7, A3, P2, M: $PG \text{ believes } PG \text{ CanProve } (\xrightarrow{K_{PG}} PG) \text{ to } V$ *(11)*

A10, 6, A7, H2, A5, P2, M:

$PG \text{ believes } V\text{-is-external-party} \wedge$

$PG \text{ believes } PG \text{ has } (\{MAC[(Price, h(OI), ID_M), Y_i], Price\}_{K_{PG}}, K_{PG}^{-1}) \wedge$

PG believes (

V has ($\{MAC[(Price, h(OI), ID_M), Y_i], Price\}_{K_{PG}}, K_{PG}^{-1}$) $\wedge$

$(MAC[(Price, h(OI), ID_M), Y_i], Price) =$

$\{\{MAC[(Price, h(OI), ID_M), Y_i], Price\}_{K_{PG}}\}_{K_{PG}^{-1}}$

$\rightarrow$ *V believes* K_{PG}^{-1}*-is-decrypting-key-for-*

$\{MAC[(Price, h(OI), ID_M), Y_i]\}_{K_{PG}}$)

$\rightarrow$ *PG believes PG CanProve* (

K_{PG}^{-1}*-is-decrypting-key-for-*

$\{MAC[(Price, h(OI), ID_M), Y_i], Price\}_{K_{PG}}$) *to V* *(12)*

10, 11, 12, P5', M:

PG believes PG CanProve (

M says $(MAC[(Price, h(OI), ID_M), Y_i], Price, ID_{PG})$) *to V* *(13)*

9, 13, P6, M:

PG believes PG CanProve (*M says* $(Price, Date, ID_{PG})$) *to V* *(14)*

14, A12, K:

PG believes PG CanProve (

M authorized credit$(M, PG, Price, Date)$) *to V* *(15)*

The proof for the goal **G5** is successful.

A.1.3 Proving the Goal G6 of KSLv1

G6: *M believes M CanProve* (

PG authorized credit$(PG, M, Price, Date)$) *to V*

Consider **Step 5** of KSLv1,

PG→M: $\{\{h(OI), Yes/No\}_{Y_i}, \{h(OI), Yes/No\}_{K_M}\}_{K_{PG}^{-1}}$

It can be transformed into:

M *sees* $\{\{h(OI), Yes/No\}_{Y_i}, \{h(OI), Yes/No\}_{K_M}\}_{K_{PG}^{-1}}$

Details of the Proof

M *sees* $\{\{h(OI), Yes/No\}_{Y_i}, \{h(OI), Yes/No\}_{K_M}\}_{K_{PG}^{-1}}$ *(1)*

1, C1:

M believes M sees $\{\{h(OI), Yes/No\}_{Y_i}, \{h(OI), Yes/No\}_{K_M}\}_{K_{PG}^{-1}}$ *(2)*

2, H1, M:

M believes M has $\{\{h(OI), Yes/No\}_{Y_i}, \{h(OI), Yes/No\}_{K_M}\}_{K_{PG}^{-1}}$ *(3)*

A1, H2, M: *M believes M has* K_{PG} *(4)*

A7, A3, K: *M believes M believes* $(\xrightarrow{K_{PG}} PG)$ *(5)*

3, 4, 5, H6, M:

M believes M has $(\ \{h(OI), Yes/No\}_{Y_i}, \{h(OI), Yes/No\}_{K_M}\ )$ *(6)*

4, 6, H2, M:

M believes M has $(\ \{h(OI), Yes/No\}_{Y_i}, \{h(OI), Yes/No\}_{K_M}, K_{PG}\ )$ *(7)*

A7, A3, P2, M: *M believes M CanProve* $(\xrightarrow{K_{PG}} PG)$ *to V* *(8)*

3, 7, 8, P3, M:

M believes M CanProve (

PG says $(\ \{h(OI), Yes/No\}_{Y_i}, \{h(OI), Yes/No\}_{K_M}\ )$ *to V* *(9)*

9, P6, M:

M believes M CanProve (PG says $\{h(OI), Yes/No\}_{K_M}$ *) to V* *(10)*

A7, A3, P2, M: *M believes M CanProve* $(\xrightarrow{K_M} M)$ *to V* *(11)*

A10, 6, A7, H2, A5, P2, M:

M believes V-is-external-party $\wedge$

M believes M has $(\ \{h(OI), Yes/No\}_{K_M}, K_M^{-1}\ ) \wedge$

M believes (

V has $(\{h(OI), Yes/No\}_{K_M}, K_M^{-1}) \wedge$

$(h(OI), Yes/No) = \{\{h(OI), Yes/No\}_{K_M}\}_{K_M^{-1}}$

$\rightarrow$ *V believes* K_M^{-1}*-is-decrypting-key-for-*$\{h(OI), Yes/No\}_{K_M}$)

$\rightarrow$ *M believes M CanProve* (

K_M^{-1}*-is-decrypting-key-for-*$\{h(OI), Yes/No\}_{K_M}$) *to V* *(12)*

10, 11, 12, P5', M:

M believes M CanProve (*PG says* $(h(OI), Yes/No, ID_M)$) *to V* *(13)*

13, P6, M: *M believes M CanProve* (*PG says* $h(OI)$) *to V* *(14)*

A7, A3, K: *M believes M believes* $(\xrightarrow{K_M} M)$ *(15)*

6, H2, 15, A7, H5, M: M *believes M has* $(h(OI), Yes/No)$ *(16)*

A10, 16, A11, H2, A4, P2, M:

M believes V-is-external-party $\wedge$

M believes M has $(h(OI), OI)$ $\wedge$

M believes (

V has $(h(OI), OI) \wedge (h(OI) = h(OI))$

$\rightarrow$ *V believes h(OI)-is-fingerprint-of-OI*)

$\rightarrow$ *M believes M CanProve* (*h(OI)-is-fingerprint-of-OI*) *to V* *(17)*

14, 17, P4, M:

M believes M CanProve (*PG says* $h(OI)$) *to V* $\wedge$

M believes M CanProve ($h(OI)$*-is-fingerprint-of-OI*) *to V*

$\rightarrow$ *M believes M CanProve* (*PG says OI*) *to V* *(18)*

18, P6, M:

M believes M CanProve (*PG says* $h(OD, Price)$) *to V* *(19)*

A10, A11, A4, P2, M:

M believes V-is-external-party $\wedge$

M believes M has $(h(OD, Price), OD, Price)$

M believes (

V has $(h(OD, Price), OD, Price)$ $\wedge$

($h(OD, Price) = h(OD, Price)$)

$\rightarrow$ *V believes* $h(OD, Price)$*-is-fingerprint-of-*$(OD, Price)$)

$\rightarrow$ *M believes M CanProve* (

$h(OD, Price)$*-is-fingerprint-of-*$(OD, Price)$) *to V* (20)

19, 20, P4, M:

M believes M CanProve (*PG says* $h(OD, Price)$) *to V* $\wedge$

M believes M CanProve (

$h(OD, Price)$*-is-fingerprint-of-*$(OD, Price)$) *to V*

$\rightarrow$ *M believes M CanProve* (*PG says* $(OD, Price)$) *to V* (21)

13, 18, 21, P6, M:

M believes M CanProve (*PG says* $(Price, Date, ID_M)$) *to V* (22)

22, A12, K:

M believes M CanProve (

PG authorized credit$(PG, M, Price, Date)$) *to V* (23)

The proof of the goal **G6** is successful.

A.2 Analyzing KSLv2 on Accountability

As the goals **G1** of KSLv2 protocol has been analyzed by the proposed KSL logic in section 7.6.4, in this section, the analyses of the goals **G2**-**G6** of KSLv2 are presented.

A.2.1 Proving the Goal G2 of KSLv2

G2: *C believes C CanProve* (

M authorized payment-order$(M, C, Price, Date)$) *to V*

Consider **Step 6** of KSLv2,

M→C: $\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}$

It can be transformed into:

C *sees* $\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}$

Details of the Proof

C *sees* $\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}$ *(1)*

1, C1, M: C *believes* C *sees* $\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}$ *(2)*

2, H1, M: C *believes* C *has* $\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}$ *(3)*

3, A8, H2, M:

C *believes* C *has* ($\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}, X_{i+1}$) *(4)*

A10, 4, A5, P2, M:

C *believes* *V-is-external-party* $\wedge$

C *believes* C *has* ($\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}, X_{i+1}$) $\wedge$

C *believes* (

V *has* ($\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}, X_{i+1}$) $\wedge$

$\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i} = \{\{\{h(OI), Yes/No, h(Z_{i+1})\}_{Y_i}\}_{X_{i+1}}\}_{X_{i+1}}$

$\rightarrow$ V *believes* (

X_{i+1}*-is-decrypting-key-for-*$\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}$))

$\rightarrow$ C *believes* C *CanProve* (

X_{i+1}*-is-decrypting-key-for-*$\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}$) *to* V *(5)*

3, A8, H2, H4, M:

C *believes* C *has* ($\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}, X_{i+1}$) $\wedge$

C *believes* C *believes* ($C \overset{X_{i+1}}{\longleftrightarrow} M$)

$\rightarrow$ C *believes* C *has* $\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}$ *(6)*

6, A8, H2, M: C *believes* C *has* ($\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}, Y_i$) *(7)*

A10, 7, A1, P2, M:

C *believes* *V-is-external-party* $\wedge$

C *believes* C *has* ($\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}, Y_i$) $\wedge$

C *believes* (

V *believes* $(C \overset{Y_i}{\longleftrightarrow} I) \wedge$

V *has* $(\ \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}, Y_i\ ) \wedge$

$(h(OI), Yes/No, h(Z_{j+1})) = \{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{Y_i}$

$\rightarrow V$ *believes* $(C \overset{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}}{\longleftrightarrow} I)\ )$

$\rightarrow C$ *believes* C *CanProve* $(C \overset{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}}{\longleftrightarrow} I)$ *to* V *(8)*

4, A8, 5, 8, P3', M:

C *believes* C *has* $(\ \{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}, X_{i+1}\ ) \wedge$

C *believes* C *believes* $(C \overset{X_{i+1}}{\longleftrightarrow} M) \wedge$

C *believes* C *CanProve* (

X_{i+1}*-is-decrypting-key-for-*$\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}$) *to* $V \wedge$

C *believes* C *CanProve* $(C \overset{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}}{\longleftrightarrow} I)$ *to* V

$\rightarrow C$ *believes* C *CanProve* (

M *says* $(\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}, ID_C)$) *to* V *(9)*

9, P6, M:

C *believes* C *CanProve* (M *says* $\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}$) *to* V *(10)*

A10, 7, A5, P2, M:

C *believes* V*-is-external-party* $\wedge$

C *believes* C *has* $(\ \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}, Y_i\ ) \wedge$

C *believes* (

V *has* $(\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}, Y_i) \wedge$

$(h(OI), Yes/No, h(Z_{j+1})) = \{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{Y_i}$

$\rightarrow V$ *believes* Y_i*-is-decrypting-key-for-*$\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}$)

$\rightarrow C$ *believes* C *CanProve* (

Y_i*-is-decrypting-key-for-*$\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}$) *to* V *(11)*

10, 11, P5, M:

C *believes* C *CanProve* (M *says* $\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}$) *to* $V \wedge$

C *believes* C *CanProve* (

Y_i*-is-decrypting-key-for-*$\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}$) *to* V

$\rightarrow C$ *believes* C *CanProve* (

$$M \textit{ says } (h(OI), Yes/No, h(Z_{j+1}))\ )\ \textit{to } V \qquad (12)$$

12, P6, M: C *believes* C *CanProve* (M *says* $h(OI)$) *to* V *(13)*

7, A8, H4, M:

C *believes* C *has* ($\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}, Y_i$) $\wedge$

C *believes* C *believes* $(C \xleftrightarrow{Y_i} I)$

$\rightarrow$ C *believes* C *has* ($h(OI), Yes/No, h(Z_{j+1})$) *(14)*

A10, 14, A11, H2, A4, P2, M:

C *believes* V*-is-external-party* $\wedge$

C *believes* C *has* $(h(OI), OI)$ $\wedge$

C *believes* (

V *has* $(h(OI), OI)$ $\wedge$ ($h(OI) = h(OI)$)

$\rightarrow$ V *believes* ($h(OI)$*-is-fingerprint-of-OI*))

$\rightarrow$ C *believes* C *CanProve* ($h(OI)$*-is-fingerprint-of-OI*) *to* V *(15)*

13, 15, P4, M:

C *believes* C *CanProve* (M *says* $h(OI)$) *to* V $\wedge$

C *believes* C *CanProve* ($h(OI)$*-is-fingerprint-of-OI*) *to* V

$\rightarrow$ C *believes* C *CanProve* (M *says* OI) *to* V *(16)*

16, P6, M: C *believes* C *CanProve* (M *says* $h(OD, Price)$) *to* V *(17)*

A10, A11, A4, P2, M:

C *believes* V*-is-external-party* $\wedge$

C *believes* C *has* $(h(OD, Price), OD, Price)$ $\wedge$

C *believes* (

V *has* $(h(OD, Price), OD, Price)$ $\wedge$

$h(OD, Price) = h(OD, Price)$

$\rightarrow$ V *believes* $h(OD, Price)$*-is-fingerprint-of-*$(OD, Price)$)

$\rightarrow$ C *believes* C *CanProve* (

$h(OD, Price)$*-is-fingerprint-of-*$(OD, Price)$) *to* V *(18)*

17, 18, P4, M:

C *believes* C *CanProve* (M *says* $h(OD, Price)$) *to* V $\wedge$

C *believes* C *CanProve* (

$h(OD, Price)$*-is-fingerprint-of-*$(OD, Price)$) *to* V

$\rightarrow$ C *believes* C *CanProve* (M *says* $(OD, Price)$) *to* V *(19)*

9, 16, 19, P6, M:

C *believes* C *CanProve* (M *says* $(ID_C, Price, Date)$) *to* V *(20)*

20, A12, K:

C *believes* C *CanProve* (

M *authorized payment-order*$(M, C, Price, Date)$) *to* V *(21)*

The proof for the goal **G2** is successful.

A.2.2 Proving the Goal G3 of KSLv2

G3: I *believes* I *CanProve* (

C *authorized debit*$(C, I, Price, Date)$) *to* V

Consider **Step 4.1** of KSLv2,

PG→I: $MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price,$
$ID_C, ID_M, h(Z_{j+1})$

It can be transformed into:

I *sees* $(MAC[(Price, h(OI), ID_M), Y_i], h(OI), i,$
$TID, Price, ID_C, ID_M, h(Z_{j+1}))$

Details of the Proof

I *sees* $(MAC[(Price, h(OI), ID_M), Y_i], h(OI), i,$
$TID, Price, ID_C, ID_M, h(Z_{j+1}))$ *(1)*

1, C1, M: *I believes I sees* $(MAC[(Price, h(OI), ID_M), Y_i], h(OI), i,$
$TID, Price, ID_C, ID_M, h(Z_{j+1}))$ *(2)*

2, H1, H2, M: *I believes I has* $(MAC[(Price, h(OI), ID_M), Y_i],$
$h(OI), i, TID, Price, ID_C, ID_M, h(Z_{j+1}))$ *(3)*

3, A8, H2, M: *I believes I has* $(MAC[(Price, h(OI), ID_M), Y_i],$
$h(OI), i, TID, Price, ID_C, ID_M, Y_i)$ *(4)*

A10, 4, A2, P2, M:

I believes V-is-external-party $\wedge$

I believes I has (

$MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price, ID_C, ID_M, Y_i) \wedge$

I believes (

V believes $(C \xleftrightarrow{Y_i} I) \wedge$

V believes I sees $MAC[(Price, h(OI), ID_M), Y_i] \wedge$

V has $(MAC[(Price, h(OI), ID_M), Y_i], Price, h(OI), ID_M, Y_i) \wedge$

$MAC[(Price, h(OI), ID_M), Y_i] = MAC[(Price, h(OI), ID_M), Y_i]$

$\rightarrow$ *V believes C says* $(Price, h(OI), ID_M, ID_I)$

$\rightarrow$ *I believes I CanProve (C says* $(Price, h(OI), ID_M, ID_I)$ *) to V* *(5)*

5, P6, M: *I believes I CanProve (C says* $(Price, ID_I)$ *) to V* *(6)*

The issuer cannot prove that the client has sent the date of transaction *Date* to her, but it has already been contained in the message sent to the issuer under the banking private network. Thus, it does not compromise the security of the system.

A.2.3 Proving the Goal G4 of KSLv2

G4: *C believes C CanProve (*
I authorized debit$(I, C, Price, Date)$ *) to V*

Consider **Step 6** of KSLv2,

M→C: $\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}$

It can be transformed into:

C sees $\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}$

Details of the Proof

C sees $\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}$ *(1)*

1, C1, M: *C believes C sees* $\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}$ *(2)*

2, H1, M: *C believes C has* $\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}$ *(3)*

3, A8, H2, H4, M:

C believes C has ($\{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{X_{i+1}}, X_{i+1}$) $\wedge$

C believes C believes ($C \overset{X_{i+1}}{\longleftrightarrow} M$)

$\rightarrow$ *C believes C has* $\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}$ *(4)*

A10, 4, A8, H2, A5, P2, M:

C believes V-is-external-party $\wedge$

C believes C has ($\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}, Y_i$) $\wedge$

C believes (

V has ($\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}, Y_i$) $\wedge$

$(h(OI), Yes/No, h(Z_{j+1})) = \{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{Y_i}$

$\rightarrow$ *V believes* Y_i*-is-decrypting-key-for-*$\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}$)

$\rightarrow$ *C believes C CanProve* (

Y_i*-is-decrypting-key-for-*$\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}$) *to V* (5)

4, A8, H4, M:

C believes C has $\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}$ $\wedge$

C believes C believes ($C \overset{Y_i}{\longleftrightarrow} I$) $\wedge$

C believes C has Y_i

$\rightarrow C\ believes\ C\ has\ (h(OI), Yes/No, h(Z_{j+1}))$ *(6)*

6, H2, M: $C\ believes\ C\ has\ h(Z_{j+1})$ *(7)*

A10, 7, A1, P2, M:

$C\ believes\ V\text{-}is\text{-}external\text{-}party\ \wedge$

$C\ believes\ C\ has\ h(Z_{j+1})\ \wedge$

$C\ believes\ ($

$V\ believes\ (M \overset{Z_{i+1}}{\longleftrightarrow} PG) \wedge V\ has\ (h(Z_{j+1}), Z_{j+1})$

$\rightarrow V\ believes\ (M \overset{Z_{i+1}}{\longleftrightarrow} PG)\ )$

$\rightarrow C\ believes\ C\ CanProve\ (M \overset{Z_{i+1}}{\longleftrightarrow} PG)\ to\ V$ *(8)*

A8, P2, M:

$C\ believes\ C\ has\ Y_i\ \wedge$

$C\ believes\ C\ believes\ (C \overset{Y_i}{\longleftrightarrow} I)\ \wedge$

$C\ believes\ ($

$(\ V\ has\ Y_i \wedge V\ believes\ (C \overset{Y_i}{\longleftrightarrow} I)) \rightarrow V\ believes\ (C \overset{Y_i}{\longleftrightarrow} I)\ )$

$\rightarrow C\ believes\ C\ CanProve\ (C \overset{Y_i}{\longleftrightarrow} I)\ to\ V$ *(9)*

4, 9, H2, 5, 6, P3', M:

$C\ believes\ C\ has\ (\ \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}, Y_i\ )\ \wedge$

$C\ believes\ C\ believes\ (C \overset{Y_i}{\longleftrightarrow} I)\ \wedge$

$C\ believes\ C\ CanProve\ ($

$Y_i\text{-}is\text{-}decrypting\text{-}key\text{-}for\text{-}\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\ )\ to\ V\ \wedge$

$C\ believes\ C\ CanProve\ (M \overset{Z_{i+1}}{\longleftrightarrow} PG)\ to\ V$

$\rightarrow C\ believes\ C\ CanProve\ ($

$I\ says\ (h(OI), Yes/No, h(Z_{j+1}), ID_C)\ )\ to\ V$ *(10)*

10, P6, M: $C\ believes\ C\ CanProve\ (\ I\ says\ h(OI)\ )\ to\ V$ *(11)*

A10, 6, A11, H2, A4, P2, M:

$C\ believes\ V\text{-}is\text{-}external\text{-}party\ \wedge$

$C\ believes\ C\ has\ (h(OI), OI)\ \wedge$

$C\ believes\ ($

$V\ has\ (h(OI), OI) \wedge (h(OI) = h(OI))$

$\rightarrow$ *V believes* $h(OI)$*-is-fingerprint-of-OI*)

$\rightarrow$ *C believes C CanProve* ($h(OI)$*-is-fingerprint-of-OI*) *to V* *(12)*

11, 12, P4, M:

C believes C CanProve (*I says* $h(OI)$) *to V* $\wedge$

C believes C CanProve ($h(OI)$*-is-fingerprint-of-OI*) *to V*

$\rightarrow$ *C believes C CanProve* (*I says OI*) *to V* *(13)*

13, P6, M: *C believes C CanProve* (*I says* $h(OD, Price)$) *to V* *(14)*

A10, A11, A4, P2, M:

C believes V-is-external-party $\wedge$

C believes C has $(h(OD, Price), OD, Price)$ $\wedge$

C believes (

V has $(h(OD, Price), OD, Price)$ $\wedge$

$(h(OD, Price) = h(OD, Price))$

$\rightarrow$ *V believes* $h(OD, Price)$*-is-fingerprint-of-*$(OD, Price)$)

$\rightarrow$ *C believes C CanProve* (

$h(OD, Price)$*-is-fingerprint-of-*$(OD, Price)$) *to V* *(15)*

14, 15, P4, M:

C believes C CanProve (*I says* $h(OD, Price)$) *to V* $\wedge$

C believes C CanProve (

$h(OD, Price)$*-is-fingerprint-of-*$(OD, Price)$) *to V*

$\rightarrow$ *C believes C CanProve* (*I says* $(OD, Price)$) *to V* *(16)*

10, 16, P6, M:

C believes C CanProve (*I says* $(ID_C, Price, Date)$) *to V* *(17)*

17, A12, K:

C believes C CanProve (

I authorized debit$(I, C, Price, Date)$) *to V* *(18)*

The proof for the goal **G4** is successful.

A.2.4 Proving the Goal G5 of KSLv2

G5: *PG believes PG CanProve (*

M authorized credit$(M, PG, Price, Date)$ *) to V*

Consider **Step 3** of KSLv2,

M→PG: $\{MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price,$
$ID_C, ID_I\}_{Z_j}, j, ID_M, MAC[(h(OI), i, TID, ID_C, ID_I), Z_{j+1}]$

It can be transformed into:

PG sees $\{MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price,$
$ID_C, ID_I\}_{Z_j}, j, ID_M, MAC[(h(OI), i, TID, ID_C, ID_I), Z_{j+1}]$

Details of the Proof

PG sees $\{MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price,$
$ID_C, ID_I\}_{Z_j}, j, ID_M, MAC[(h(OI), i, TID, ID_C, ID_I, Z_{j+1}]$ *(1)*

1, C1, M:

PG believes PG sees

$\{MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price, ID_C, ID_I\}_{Z_j},$
$j, ID_M, MAC[(h(OI), i, TID, ID_C, ID_I), Z_{j+1}]$ *(2)*

2, H1, M:

PG believes PG has

$\{MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price, ID_C, ID_I\}_{Z_j},$
$j, ID_M, MAC[(h(OI), i, TID, ID_C, ID_I), Z_{j+1}]$ *(3)*

3, H2, M:

PG believes PG has

$\{MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price, ID_C, ID_I\}_{Z_j}$ *(4)*

A10, 4, A8, H2, A5, P2, M:

PG believes V-is-external-party $\wedge$

PG believes PG has $(\{MAC[(Price, h(OI), ID_M), Y_i], h(OI), i,$
$TID, Price, ID_C, ID_I\}_{Z_j}, Z_j) \wedge$

PG believes (

V has $(\{MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price,$
$ID_C, ID_I\}_{Z_j}, Z_j) \wedge$

$(MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price, ID_C, ID_I) =$
$\{\{MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price, ID_C, ID_I\}_{Z_j}\}_{Z_j}$

$\rightarrow$ *V believes* Z_j*-is-decrypting-key-for-*
$\{MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID, Price, ID_C, ID_I\}_{Z_j}$)

$\rightarrow$ *PG believes PG CanProve* (
Z_j*-is-decrypting-key-for-*$\{MAC[(Price, h(OI), ID_M), Y_i], h(OI),$
$i, TID, Price, ID_C, ID_I\}_{Z_j}$) *to V* *(5)*

4, A8, H2, H4, M:

PG believes PG has $(\{MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID,$
$Price, ID_C, ID_I\}_{Z_j}, Z_j) \wedge$

PG believes PG believes $(M \overset{Z_j}{\longleftrightarrow} PG) \wedge$

$\rightarrow$ *PG believes PG has* $(MAC[(Price, h(OI), ID_M), Y_i], h(OI), i,$
$TID, Price, ID_C, ID_I$) *(6)*

A10, 6, H2, A1, P2, M:

PG believes V-is-external-party $\wedge$

PG believes PG has $MAC[(Price, h(OI), ID_M), Y_i] \wedge$

PG believes (

V believes $(C \overset{Y_i}{\longleftrightarrow} I) \wedge$

V has ($MAC[(Price, h(OI), ID_M), Y_i], Price, h(OI), ID_M, Y_i$) $\wedge$

$MAC[(Price, h(OI), ID_M), Y_i] = MAC[(Price, h(OI), ID_M), Y_i]$

$\rightarrow$ *V believes* $(C \overset{MAC[(Price, h(OI), ID_M), Y_i]}{\longleftrightarrow} I)$)

$\rightarrow$ *PG believes PG CanProve* $(C \overset{MAC[(Price,h(OI),ID_M),Y_i]}{\longleftrightarrow} I)$ *to V* *(7)*

A8, P2, M:

PG believes PG has $Z_j \wedge$

PG believes PG believes $(M \overset{Z_j}{\longleftrightarrow} PG) \wedge$

PG believes (

(*V has* $Z_j \wedge$ *V believes* $(M \overset{Z_j}{\longleftrightarrow} PG)) \rightarrow$ *V believes* $(M \overset{Z_j}{\longleftrightarrow} PG)$)

$\rightarrow$ *PG believes PG CanProve* $(M \overset{Z_j}{\longleftrightarrow} PG)$ *to V* *(8)*

4, 8, H2, 5, 6, P3', M:

PG believes PG has (

$\{MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID,$

$Price, ID_C, ID_I\}_{Z_j}, Z_j) \wedge$

PG believes PG CanProve $(M \overset{Z_j}{\longleftrightarrow} PG)$ *to V* $\wedge$

PG believes PG CanProve (

Z_j*-is-decrypting-key-for-*$\{MAC[(Price, h(OI), ID_M), Y_i],$

$h(OI), i, TID, Price, ID_C, ID_I\}_{Z_j})$ *to V* $\wedge$

PG believes PG CanProve $(C \overset{MAC[(Price,h(OI),ID_M),Y_i]}{\longleftrightarrow} I)$ *to V*

$\rightarrow$ *PG believes PG CanProve* (

M says $(MAC[(Price, h(OI), ID_M), Y_i], h(OI), i, TID,$

$Price, ID_C, ID_I, ID_{PG})$) *to V* *(9)*

9, P6, M:

PG believes PG CanProve (*M says* $(ID_{PG}, Price, Date)$) *to V* *(10)*

10, A12, K:

PG believes PG CanProve (

M authorized credit$(M, PG, Price, Date)$) *to V* *(11)*

The proof of the goal **G5** is successful.

A.2.5 Proving the Goal G6 of KSLv2

G6: *M believes M CanProve (*

PG authorized $credit(PG, M, Price, Date)$ *) to V*

Consider **Step 5** of KSLv2,

PG→M: $\{Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$
$h(Yes/No, h(OI), h(Y_i))\}_{Z_{j+1}}$

It can be transformed into:

M sees $\{Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$
$h(Yes/No, h(OI), h(Y_i))\}_{Z_{j+1}}$

Details of the Proof

M sees $\{Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$
$h(Yes/No, h(OI), h(Y_i))\}_{Z_{j+1}}$ *(1)*

1, C1: *M believes M sees* $\{Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$
$h(Yes/No, h(OI), h(Y_i))\}_{Z_{j+1}}$ *(2)*

2, H1, M: *M believes M has* $\{Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$
$h(Yes/No, h(OI), h(Y_i))\}_{Z_{j+1}}$ *(3)*

A10, 3, A8, H2, A5, P2, M:

M believes V-is-external-party $\wedge$

M believes M has ($\{Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$
$h(Yes/No, h(OI), h(Y_i))\}_{Z_{j+1}}, Z_{j+1}$ *)* $\wedge$

M believes (

V has ($(\{Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$
$h(Yes/No, h(OI), h(Y_i))\}_{Z_{j+1}}, Z_{j+1})$ $\wedge$

$$(Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$$
$$h(Yes/No, h(OI), h(Y_i)) \ ) =$$
$$\{ \ \{Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$$
$$h(Yes/No, h(OI), h(Y_i))\}_{Z_{j+1}} \ \}_{Z_{j+1}} \ )$$
$\rightarrow$ *V believes* Z_{j+1}*-is-decrypting-key-for-*
$$\{Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$$
$$h(Yes/No, h(OI), h(Y_i))\}_{Z_{j+1}} \ )$$
$\rightarrow$ *M believes M CanProve (*

Z_{j+1}*-is-decrypting-key-for-*
$$\{Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$$
$h(Yes/No, h(OI), h(Y_i))\}_{Z_{j+1}}$ *) to V* *(4)*

3, A8, H2, H4, M:

M believes M has ($\{Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$
$h(Yes/No, h(OI), h(Y_i))\}_{Z_{j+1}}, Z_{j+1}$ *)* $\wedge$

M believes M believes $(M \overset{Z_{j+1}}{\longleftrightarrow} PG)$

$\rightarrow$ *M believes M has (* $Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$
$h(Yes/No, h(OI), h(Y_i))$ *)* *(5)*

A10, 5, H2, A1, P2, M:

M believes V-is-external-party $\wedge$

M believes M has $\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}$ $\wedge$

M believes (

V believes $(C \overset{Y_i}{\longleftrightarrow} I)$ $\wedge$ *V has* $(\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}, Y_i \ )$ $\wedge$

$(h(OI), Yes/No, h(Z_{j+1})) = \{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}\}_{Y_i}$

$\rightarrow$ *V believes* $(C \overset{\{h(OI),Yes/No,h(Z_{j+1})\}_{Y_i}}{\longleftrightarrow} I)$ *)*

$\rightarrow$ *M believes M CanProve* $(C \overset{\{h(OI),Yes/No,h(Z_{j+1})\}_{Y_i}}{\longleftrightarrow} I)$ *to V* *(6)*

A8, P2, M:

M believes M has Z_{j+1} $\wedge$

M believes M believes $(M \overset{Z_{j+1}}{\longleftrightarrow} PG)$ $\wedge$

M believes (

$(\; V \; has \; Z_{j+1} \wedge V \; believes \; (M \overset{Z_{j+1}}{\longleftrightarrow} PG) \;)$

$\rightarrow V \; believes \; (M \overset{Z_{j+1}}{\longleftrightarrow} PG) \;)$

$\rightarrow M \; believes \; M \; CanProve \; (M \overset{Z_{j+1}}{\longleftrightarrow} PG) \; to \; V$ *(7)*

3, A8, H2, 7, 4, 6, P3', M:

$M \; believes \; M \; has \; (\; \{Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$

$h(Yes/No, h(OI), h(Y_i))\}_{Z_{j+1}}, Z_{i+1} \;) \wedge$

$M \; believes \; M \; CanProve \; (M \overset{Z_{j+1}}{\longleftrightarrow} PG) \; to \; V \wedge$

$M \; believes \; M \; CanProve \; (\; Z_{j+1}\text{-}is\text{-}decrypting\text{-}key\text{-}for\text{-}$

$\{Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$

$h(Yes/No, h(OI), h(Y_i))\}_{Z_{j+1}} \;) \; to \; V$

$M \; believes \; M \; CanProve \; (C \overset{\{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i}}{\longleftrightarrow} I) \; to \; V$

$M \; believes \; M \; CanProve \; ($

$PG \; says \; (Yes/No, \{h(OI), Yes/No, h(Z_{j+1})\}_{Y_i},$

$h(Yes/No, h(OI), h(Y_i)), ID_M) \;) \; to \; V$ *(8)*

8, P6, M: $M \; believes \; PG \; says \; h(Yes/No, h(OI), h(Y_i)) \; to \; V$ *(9)*

From Step 2 of KSLv2:

$M \; believes \; M \; sees \; ($

$\{OI, h(Y_i), Price, ID_C, ID_I, MAC[(Price, h(OI), ID_M), Y_i]\}_{X_i},$

$MAC[(OI, Price, ID_C, ID_I), X_{i+1}] \;)$ *(10)*

10, H1, H2, M: $M \; believes \; M \; has \; (h(Y_i), OI)$ *(11)*

11, H3, M: $M \; believes \; M \; has \; (h(Y_i), h(OI))$ *(12)*

5, 12, H2, M: $M \; believes \; M \; has \; (h(Y_i), h(OI), Yes/No)$ *(13)*

5, H2, M: $M \; believes \; M \; has \; h(Yes/No, h(OI), h(Y_i))$ *(14)*

A10, 13, 14, H2, A4, P2, M:

$M \; believes \; V\text{-}is\text{-}external\text{-}party \wedge$

$M \; believes \; M \; has \; (h(Yes/No, h(OI), h(Y_i)), h(Y_i), h(OI), Yes/No) \wedge$

$M \; believes \; ($

$V \; has \; (h(Yes/No, h(OI), h(Y_i)), h(Y_i), h(OI), Yes/No) \wedge$

$(\; h(Yes/No, h(OI), h(Y_i)) = h(Yes/No, h(OI), h(Y_i)) \;)$

$\rightarrow$ *V believes*

$h(Yes/No, h(OI), h(Y_i))$*-is-fingerprint-of-*$(Yes/No, h(OI), h(Y_i))$)

$\rightarrow$ *M believes M CanProve* (

$h(Yes/No, h(OI), h(Y_i))$*-is-fingerprint-of-*$(Yes/No, h(OI), h(Y_i)))$ *to V*

(15)

9, 15, P4, M:

M believes PG says $h(Yes/No, h(OI), h(Y_i))$ *to V* $\wedge$

M believes M CanProve

$h(Yes/No, h(OI), h(Y_i))$*-is-fingerprint-of-*$(Yes/No, h(OI), h(Y_i))$ *to V*

$\rightarrow$ *M believes M CanProve* (*PG says* $(Yes/No, h(OI), h(Y_i)$) *to V* *(16)*

16, P6, M: *M believes M CanProve* (*PG says* $h(OI)$) *to V* *(17)*

A10, 12, A11, H2, A4, P2, M:

M believes V-is-external-party $\wedge$

M believes M has $(h(OI), OI)$ $\wedge$

M believes (*V has* $(h(OI), OI)$ $\wedge$ ($h(OI) = h(OI)$)

$\rightarrow$ *V believes h(OI)-is-fingerprint-of-OI*)

$\rightarrow$ *M believes M CanProve* (*h(OI)-is-fingerprint-of-OI*) *to V* *(18)*

17, 18, P4, M:

M believes M CanProve (*PG says* $h(OI)$) *to V* $\wedge$

M believes M CanProve ($h(OI)$*-is-fingerprint-of-OI*) *to V*

$\rightarrow$ *M believes M CanProve* (*PG says OI*) *to V* *(19)*

19, P6, M: *M believes M CanProve* (*PG says* $h(OD, Price)$) *to V* *(20)*

A10, A11, A4, P2, M:

M believes V-is-external-party $\wedge$

M believes M has $(h(OD, Price), OD, Price)$ $\wedge$

M believes (

V has $(h(OD, Price), OD, Price)$ $\wedge$ ($h(OD, Price) = h(OD, Price)$)

$\rightarrow$ *V believes* $h(OD, Price)$*-is-fingerprint-of-*$(OD, Price)$)

$\rightarrow$ *M believes M CanProve* (

$$h(OD, Price)\text{-}is\text{-}fingerprint\text{-}of\text{-}(OD, Price)\ )\ to\ V \qquad (21)$$

20, 21, P4, M:

$$M\ believes\ M\ CanProve\ (\ PG\ says\ h(OD, Price)\ )\ to\ V\ \wedge$$
$$M\ believes\ M\ CanProve\ ($$
$$h(OD, Price)\text{-}is\text{-}fingerprint\text{-}of\text{-}(OD, Price)\ )\ to\ V$$
$$\rightarrow M\ believes\ M\ CanProve\ (\ PG\ says\ (OD, Price)\ )\ to\ V \qquad (22)$$

8, 19, 22, P6, M:

$$M\ believes\ M\ CanProve\ (\ PG\ says\ (ID_M, Price, Date)\ )\ to\ V \qquad (23)$$

23, A12, K:

$$M\ believes\ M\ CanProve\ ($$
$$PG\ authorized\ credit(PG, M, Price, Date))\ to\ V \qquad (24)$$

The proof of the goal **G6** is successful.

Appendix B

Publications

Parts of the research in this thesis have been published:

- Chapter 4 includes the proposed SET framework, presented at the 2003 IADIS International E-Society Conference [KSL03b], and its generalization reported at the 2nd International Conference on Information and Communication Technologies (ICT2004) [KSL04b].
- Chapter 5 incorporates two account-based mobile payment protocols, KSLv1 protocol presented at the 2003 International Conference on Cryptology in India (Indocrypt2003) [KSL03a] and KSLv2 protocol presented at the 2004 International Conference on Information Technology: Coding and Computing (ITCC2004) [KSL04c], respectively.
- Chapter 6 presents the results of implementation of KSLv1 protocol reported at the 2004 IEEE Conference on Cybernetics and Intelligent Systems [TKL04].
- The proposed accountability logic in Chapter 7 was presented at the ITCC2004 conference [KSL04a].
- Chapter 8 includes the proposed limited-used key generation technique for Internet transactions presented at the 5th International Workshop on Information Security Applications (WISA2004) [KLS05].

- Chapter 9 presents the proposed prepaid micropayment protocol which was reported at the 2$^{\text{nd}}$ International Workshop on Security in Information Systems (WOSIS2004) [KSL04d]. Its extension to postpaid micropayment was presented at the 6$^{\text{th}}$ International Conference on Information Integration and Web Based Applications and Services (iiWAS2004) [KSL04e].