Journal of Information Processing Systems, Vol.8, No.3, September 2012 http://dx.doi.org/10.3745/JIPS.2012.8.3.495

Designing an Efficient and Secure Credit Card-based Payment System with Web Services Based on the ANSI X9.59-2006

Chi Po Cheong*, Simon Fong**, Pouwan Lei***, Chris Chatwin* and Rupert Young*

Abstract—A secure Electronic Payment System (EPS) is essential for the booming online shopping market. A successful EPS supports the transfer of electronic money and sensitive information with security, accuracy, and integrity between the seller and buyer over the Internet. SET, CyberCash, Paypal, and iKP are the most popular Credit Card-Based EPSs (CCBEPSs). Some CCBEPSs only use SSL to provide a secure communication channel. Hence, they only prevent "Man in the Middle" fraud but do not protect the sensitive cardholder information such as the credit card number from being passed onto the merchant, who may be unscrupulous. Other CCBEPSs use complex mechanisms such as cryptography, certificate authorities, etc. to fulfill the security schemes. However, factors such as ease of use for the cardholder and the implementation costs for each party are frequently overlooked. In this paper, we propose a Web service based new payment system, based on ANSI X9.59-2006 with extra features added on top of this standard. X9.59 is an Account Based Digital Signature (ABDS) and consumer-oriented payment system. It utilizes the existing financial network and financial messages to complete the payment process. However, there are a number of limitations in this standard. This research provides a solution to solve the limitations of X9.59 by adding a merchant authentication feature during the payment cycle without any addenda records to be added in the existing financial messages. We have conducted performance testing on the proposed system via a comparison with SET and X9.59 using simulation to analyze their levels of performance and security.

Keywords—Payment Protocols, Electronic Commerce, SET, X9.59, Web Services

1. INTRODUCTION

Online credit card payments gained huge popularity since the boom of the e-commerce era. However, the Internet is an open platform and so the communication path between users is naturally insecure. This means that all communication is potentially vulnerable to eavesdroppers probing and modifying messages/instructions as they pass between the communicating end-points. Transmitting sensitive information between the consumer and merchant through the Internet is dangerous if there is no secure path. Unsecure communication aside, there are a number

Manuscript received January 31, 2012; accepted May 1, 2012.

Corresponding Author: Simon Fong
* University of Sussex, Brighton, BN1 9RH, United Kingdom (webster@macau.ctm.net, c.r.chatwin@sussex.ac.uk, r.c.d.young@sussex.ac.uk)
** Department of Computer and Information Science, University of Macau, Macau SAR (ccfong@umac.mo)
*** University of Bradford, Bradford, West Yorkshire, United Kingdom (pw_y2k@yahoo.com)

of factors that each participant must consider. For example, on the merchant's side, there is concern about whether the credit card or the cardholder is genuine. There is no easy way to know whether the consumer is a genuine cardholder over the Internet. In the past, merchants have been incurring increasing losses due to cardholder disputes and fraud. On the other hand, cardholders worry about the privacy of their sensitive information, such as their credit card number. They want to avoid unauthorized usage of their credit card and be sure that the transaction amount cannot be modified by the merchant. The security issues have deterred the majority of consumers from purchasing online.

To identify the success of an Internet payment system, a set of factors should be evaluated. Technically, the four sets of criteria that must be weighed [1]: security, cost, convenience and universality Security features consist of identification, authentication, confidentiality, integrity, and non-repudiation They should all be covered by a reliable payment system. Secure Socket Layer (SSL) is the most commonly used protocol in e-commerce, but it is not a payment protocol. It only protects the confidential data in the communication channel between the merchant and the consumer. The credit card number or information could be stolen on the merchant's side during the payment process. Two types of payment protocols are used in the credit card-based payment system. The first type uses a virtual number instead of the actual credit card number during the payment cycle [2]. The second one uses the actual credit card number with encryption mechanisms or PKI. Examples are CyberCash [3], SET [4], iKP [5], VbV [6], etc. However, most of them have not fully utilized and integrated with the existing financial payment infrastructure and they are not consumer-oriented payment protocol.

Many secure credit card-based payment systems have been proposed in recent years, mainly in two different directions. Some systems are focusing on convenience by compromising certain security requirements. They tend to rely more on SSL moving away from SET for instance. X9.59 is a payment protocol that can balance between security and simplicity. The X9.59 standard eliminates the requirement for a Certificate Authority (CA). It utilizes the existing financial payment network and uses less encryption mechanisms during the payment process. However, X9.59 is inefficient in the area that requires the transfer of the Signed Payment Object from the Merchant Financial Institution (MFI) to the Consumer Financial Institution (CFI) occupying the existing financial network. Moreover, it uses the existing payment network to exchange the signature on the merchant public key, which is signed by the CFI.

CONSEPP [7] is a payment protocol, which is also based on the X9.59 standard. It proposes a merchant authentication method with the input of X9.59 during the payment cycle and a lightweight method for the transfer of the merchant public key from the CFI to the consumer through the MFI and the merchant. However, the CONSEPP does not solve the limitations of X9.59, which requires some addenda records being added to the existing financial messages. In addition, more addenda records are required to be added in existing financial messages in order to transfer the signed merchant public key from the CFI to the consumer. This paper presents a novel method to solve the limitations of X9.59 and CONSEPP. We propose a system that is based on this standard with some added modifications and enhancements. In addition, the proposed system defines objects that are not included in the original X9.59 standard.

The organization of this paper is structured as follows: Section 2 gives the details of the background architecture of X9.59, which serves as the main vehicle for improving CONSEPP. A new credit card-based payment system that is supposed to be efficient and secure, which is called ESCCPS, is proposed and fully described in Section 3. Section 4 sheds light onto the design of ESCCPS in the aspects of UML class diagrams, operation flows, and message formats. The performance of ESCCPS is then evaluated via a simulation. Section 5 concludes this paper.

2. The Architecture of ANSI X9.59-2006

X9.59 is a multi-purpose account based electronic payment model used in e-commerce for the Financial Services Industry. It was first introduced by Lynn and Anne Wheeler [8] in 1997 and was approved on May 24, 2006. X9.59 [9] is an Account Based/Authority Digital Signature (ABDS) / (AADS) [10] payment system that is defined in ANSI. The digital signature is based on the Public Key Infrastructure (PKI) and provides the two major features of data integrity and non-repudiation. Digital signature is a unique digest of the message, which is encrypted with the sender's private key. The receiver uses the sender's public key in the signature verification process. In order to ensure that the public key is safe to trust, a third-party Certificate Authority (CA) must be involved in the payment process. This is called the Certificate Based/Authority Digital Signature (CBDS) / (CADS) model because it uses a certificate such as X.509 to transfer the public key between each participant. However, a complex certificate validation system is required in the CBDS. For example, the CA hierarchy model is used for SET. Moreover, the certificate revocation list (CRL) is a major issue in the CBDS model. It is required to guarantee that the certificate itself has not expired or been revoked in real-time. It is a very time consuming process because not all the CAs know each other. Multiple certificates are required to be validated even by the CA themselves. It is believed that a successful digital signature model would leverage the existing infrastructure and business processes to ensure trust in financial transactions. Otherwise, a large investment amount is required to establish a new signature infrastructure.

The X9.59 standard uses the ABDS model to tackle the limitations of the CBDS model. The ABDS model integrates digital signatures into the existing financial transaction process model. For example, the cardholder registers his/her public key with other information such as their telephone number, address, etc. in the bank account record. This means that the public key used for the verification of the digital signature is stored in the Financial Institutions (FI) account record. This model can eliminate the requirement for the CA in the verification process because the digital signature is verified by the FIs. X9.59 can support many payment methods such as e-Check payments, credit card payments, and other account-based payments. In addition, this standard can reduce the cost of the issue and maintenance of the digital certificate. Hence, this standard eliminates the use of other complex payment mechanisms and keeps it simple. It utilizes the existing infrastructures during payment cycles much more effectively than the other payment systems. It defines a collection of secure payment objects that offer authentication, integrity, and the prevention of replay during the payment process. However, this standard only focuses on the Payment Object delivery. For example, the Signed Payment Object, Payment Ack Object and Payment Consideration Ack Objects are transferred between the consumer and merchant. Other functions in electronic payment systems, such as: merchant authentication, shopping experience, payment negotiation, etc. are outside of the scope of this standard.

2.1 A Secure Payment Object in X9.59

The signed payment object shown in Table 1 is a core component in X9.59. This standard us-

Table 1. Data fields defined in the X9.59 Signed Payment Object

Field Name	Filed Description
StandardVersion	X9.59 protocol version
Object_Type	Type of object
Paycode	Payment instruction to merchant
PrcC	Customer account number
LUID	(Customer) locally unique identified
PrcM	Merchant account number/id
PaydataC	Currency type and transaction amount
DateS	Transaction date/time as generated by the consumer
DateE	Account expiration
H{OD}	Hash of order details
SigPayObject	Digital signature of X9.59 signed elements

es digital signature techniques to secure the payment object. The consumer generates a payment object during the payment cycle and signs this payment object using the consumer's private key. Then CFI uses the consumer's public key to verify the Signed Payment Object. The key pair is generated during the consumer registration phase and the public key is stored in the CFI account record like other pieces of consumer account information.

X9.59 uses a Payment Routing Code (PRC) in the payment object instead of the actual credit card number. The PRC is not a number that is used only one time. It is generated by the CFI in the account setup procedures. The Signed Payment Object is transferred from the consumer to the merchant in plain text format and is used to construct the authorization request message. Although the PRC can be seen or attacked by a third party, the PRC in the payment object cannot be used without the consumer's signature for each party. With the Signed Payment Object, the transaction can maintain the integrity and non-repudiation security schemes. The Local Unique Identifier (LUID) can be used to prevent a replay attack of the transaction. Typically it is a sequence number generated by the consumer for each payment transaction, like a payment or check number.

2.2 Basic Payment Flow

There are four participants in the X9.59 credit card-based payment process. The consumer, also called the cardholder, wants to buy products or services on the Internet. A merchant or credit card acceptor provides goods or services on their online store. The Consumer's Financial Institution (CFI) is referred to as the issuer bank, which issues the credit card to the consumer. In the X9.59 standard, the CFI generates the key-pairs for the consumer in the account setup phase. The Merchant's Financial Institution (MFI) is referred to as the acquirer bank, merchant bank, or acquirer. The MFI holds the merchant account and acts as the payment gateway between the merchant and the existing credit card payment system.

The X9.59 payment objects support different payment methods such as electronic checks, debit cards, credit cards, etc. This research shows that the most commonly used payment method is the credit card payment method, which must cooperate with X9.59 payment objects. A generic credit card-based payment flow is given in Fig. 1. The consumer selects the desired item before using some shopping experience protocols such as IOTP [11] or OFX [12]. Before the

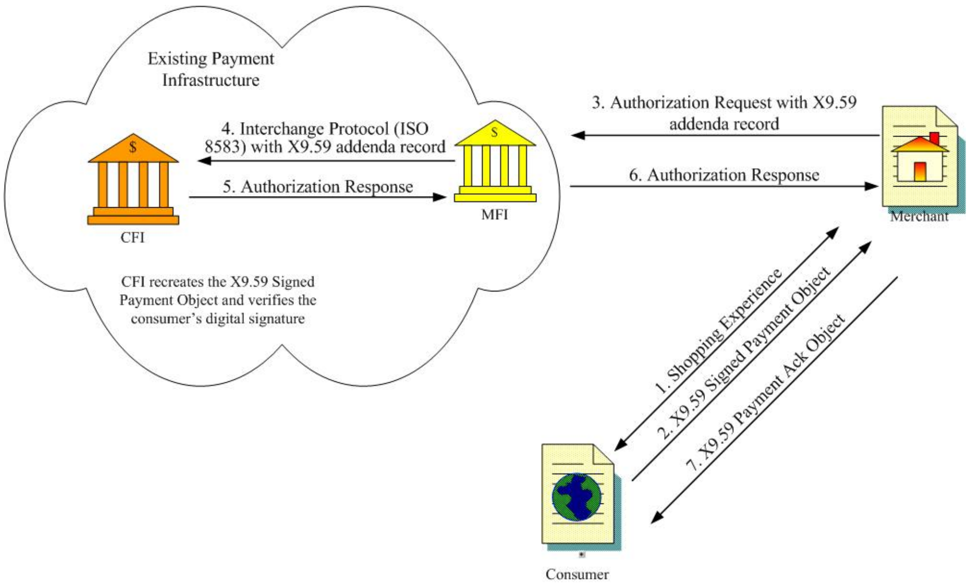
Existing Payment Infrastructure
4. Interchange Protocol (ISO 8583) with X9.59 addenda record
5. Authorization Response
CFI
MFI
CFI recreates the X9.59 Signed Payment Object and verifies the consumer's digital signature
3. Authorization Request with X9.59 addenda record
6. Authorization Response
Merchant
1. Shopping Experience
2. X9.59 Signed Payment Object
7. X9.59 Payment Ack Object
Consumer

Fig. 1. The basic payment flow in X9.59

purchasing activities occur, the consumer should authenticate the merchant by using an existing mechanism, such as X.509 [13]. The consumer forms the order detail document, computes the hash for the order detail, and builds the X9.59 payment object using a trusted electronic wallet. Then, the consumer transfers the order detail document and X9.59 Signed Payment Object to the merchant. The merchant verifies the object and creates the authorization request message with X9.59 addenda records and transfers to the MFI. The MFI creates an ISO 8583 [14] authorization request message with the copy of the X9.59 addenda records and sends it to the CFI. The ISO 8583 standard is an interchange message specification and is designed for the originated financial transaction card message. It is used for exchanging the electronic transactions made by the cardholder using the payment card. The CFI recreates the X9.59 payment object and verifies the consumer's signature using the consumer's public key, which is stored in the CFI via the consumer account setup procedure. The CFI returns the standard authorization response message to the MFI as being either approved or declined. It does so through the existing financial payment network. Then, the MFI forwards the standard authorization response message to the merchant; and the merchant sends the Signed Payment Ack Object to the consumer.

2.3 Mapping Between X9.59 and ISO 8583

One of the goals in X9.59 is to utilize the existing infrastructure as much as possible during the payment cycle. In the credit card payment process, this standard will work with other interchange protocols, such as ISO 8583, to complete the payment process. The data elements in X9.59 should be mapped into the standard authorization request message as defined in ISO 8583. Moreover, some addenda records are required to be added into this message to fulfill the standard. The mapping between X.59 and ISO8583 is shown in Table 2.

Those X9.59 addenda records are then transmitted along with the standard authorization re-

Table 2. The mapping between X9.59 and ISO8583

X9.59 Signed Payment Object	ISO 8583 Authorization Request Message			AddendaRecord
	BN		Length	
StandardVer. 16b				Y
Paycode 16b				Y
PrcC	2	Primary Account Number	19	N
LUID 16b				Y
PrcM	42	Card acceptor identification code	15	N
PaydataC	4 49	Amount, transaction Currency code, transaction	12 3	N
DateS 64b				Y
DateE	14	Date, expiration (YYMM)	4	N
H{OD} 16b				Y
SigPayObject 328b				Y

quest message and moved into the new ISO bit. Other X9.59 data elements such as PrcC, PrcM, PaydataC, and DateE use the existing fields to store the values. For example, the PRC is stored in the ISO bit number 2, which is used to store the primary account number in a traditional message.

2.4 The Advantages and Disadvantages of X9.59

The ideas or goals of the X9.59 standard are to simplify the payment flow and to utilize the existing infrastructure as much as possible in the electronic payment system. It avoids the use of complex mechanisms and reduces the amount of participants during the payment cycle. There are several characteristics or advantages in the X9.59 standard. It tries to eliminate the requirement for a Certificate Authority (CA) by using an account authority. For example, the public key, which is used to verify the consumer's signature, is stored in the CFI account record. Therefore, the problems with consumer certificate distribution and revocation, which are the drawbacks in certificate-based systems, are eliminated. In addition, the implementation or deployment cost will be reduced because a consumer certificate is no longer required. Using a Payment Routing Code (PRC) can protect the credit card number because it is not shown in the payment cycle. Although the PRC along with other data elements are sent to the merchant without encryption, unauthorized users cannot use the PRC without the consumer's accompanying digital signature. In X9.59, the encryption requirement is minimal during the online payment cycle. Less encryption processing reduces the transaction processing time and eliminates the key distribution problems.

However, some limitations can be found in the X9.59 standard. Modifications to the ISO 8583 are needed because the Signed Payment Object is verified by the CFI. Therefore, some addenda records are required to be added into the existing interchange protocol, which is transmitted between the MFI and CFI. However, it is difficult to make the modifications with some well branded credit card companies such as Visa and MasterCard. In addition, X9.59 does not provide any merchant authentication mechanism. It uses other methods such as X.509 to fulfill this security scheme. We decided to improve this shortcoming in this research project.

3. Efficient and Secure Credit Card-based Payment System

The proposed Efficient Secure Credit Card-based Payment System (ESCCPS) is based on ANSI X9.59-2006. It provides the solutions to solve the limitations of X9.59 and has some extra features on top of X9.59. The ESCCPS also defines additional functions that are not included in the X9.59 standard.

3.1 ESCCPS Payment Flow

The ESCCPS payment flow is based on X9.59 and has some modifications to eliminate the limitations of X9.59. The payment flow is shown in Fig. 2. The merchant sends a Signed Invoice Object (SIO) with its signature to the consumer after the consumer presses the check-out button. The merchant's public key is stored in the SIO and it can be retrieved by the consumer. The consumer creates a Signed Payment Object (SPO), in which the SIO is embedded, and sends it to the merchants. The merchant then constructs a Signed Authorization Request Object (SARO) and sends it to the MFI. In the X9.59 standard, the consumer's signature in the SPO is verified by the CFI. However in the ESCCPS, most of the verification process is done by the MFI. The MFI acquires the consumer's public key and requests for the signature service from the CFI Web service interface through the Simple Object Access Protocol (SOAP). Then, the MFI uses the consumer's public key to verify the consumer signature stored in the SPO. The MFI also verifies the SIO, which is embedded in the SPO, using the merchant's public key.

The merchant's public key is stored in the MFI during the merchant account setup phases. After the verification process, the merchant constructs a standard ISO 8583 authorization request message without addenda records and sends it to the CFI through the existing financial network.

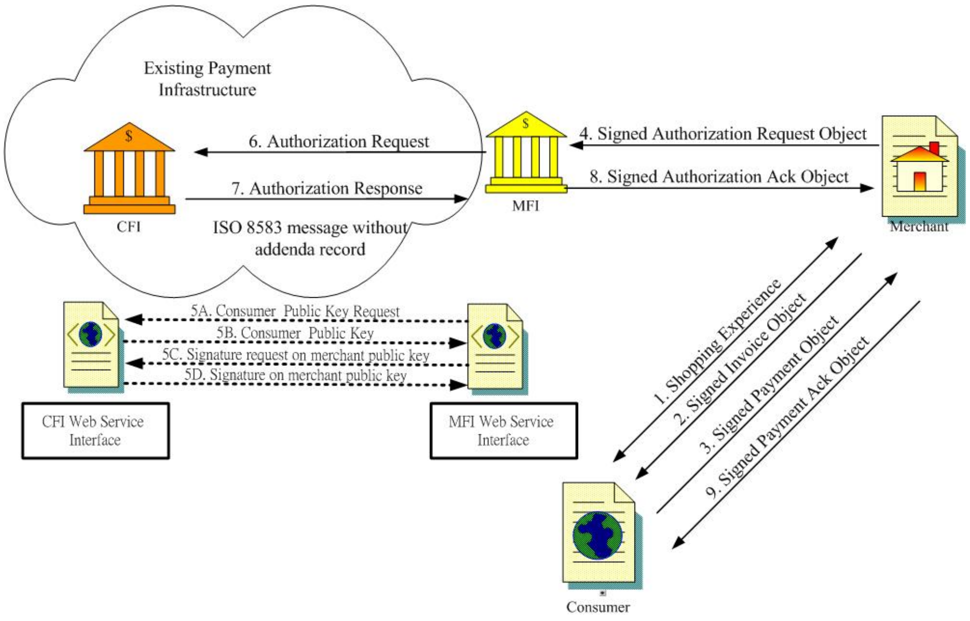
Existing Payment Infrastructure
CFI
6. Authorization Request
7. Authorization Response
ISO 8583 message without addenda record
MFI
4. Signed Authorization Request Object
8. Signed Authorization Ack Object
Merchant
5A. Consumer Public Key Request
5B. Consumer Public Key
5C. Signature request on merchant public key
5D. Signature on merchant public key
CFI Web Service Interface
MFI Web Service Interface
1. Shopping Experience
2. Signed Invoice Object
3. Signed Payment Object
9. Signed Payment Ack Object
Consumer

Fig. 2. ESCCPS payment flow

When the CFI receives the message, it maps the PRC to the actual credit card number and goes through the internal authorization policy rules. Then, the CFI returns the authorization response message to MFI as being either approved or declined. It does so through the existing financial network. The MFI creates a Signed Authorization Ack Object (SAAO) with the MFI's signature and sends it to the merchant. In addition, the SAAO also includes the signature on the merchant's public key, which is signed by the CFI. After the merchant verifies the signature in the SAAO by using the MFI's public key, the merchant creates a Signed Payment Ack Object (SPAO) with the CFI signature and relays it to the consumer. The consumer can use the CFI's signature to verify the merchant's public key. If the merchant's public key is valid, the consumer uses the merchant's public key, which is retrieved from the SIO, to verify the SPAO.

The owner signs all of the defined objects in the ESCCPS. Each object is verified by the recipient using the public key of the create owner The participants in the ESCCPS cannot alter the signed objects during the payment cycle, nor can they repudiate the transaction after completing the transaction. The ESCCPS provides a simple method for each participant to be able to exchange the public key between themselves. Therefore, the ESCCPS can keep the payment cycle simple and efficient.

3.2 Signed Objects in the ESCCPS

There are five signed objects, which are shown in Fig. 3, that are defined in the ESCCPS. The Signed Invoice Object (SIO) is used in the merchant authentication process. It includes the merchant's public key (MPK) for the consumer to verify the Signed Payment Ack Object (SPAO). The Signed Authorization Request Object (SARO) is used by the merchant to initiate the authorization request from the MFI. Then, the MFI sends the response code through the Signed Authorization Ack Object (SAAO) to the merchant. The SAAO also includes the CFI's signature on the MPK. The CFI signature can be used to verify the MPK by the consumer. Finally the merchant sends the Signed Payment Ack Object to the consumer to complete the transaction. The consumer uses the valid MPK to verify the SPAO. The Signed Invoice Object (SIO) is a new object defined in the ESCCPS. It includes merchant information, the hash of order details, and the merchant's public key (MPK). The SIO is sent to the consumer after being signed by the merchant using their private key. The consumer retrieves and stores the MPK from the SIO. At this stage, the MPK does not perform any verification. However, the MPK will be verified using the signature signed by the CFI when the consumer receives the SAAO. A binary copy of the SIO is embedded in the Signed Payment Object (SPO). The SPO is created and signed by the consumer and is returned to the merchant. The merchant does not need to verify the SPO. The SPO will be verified by the MFI when the MFI receives the SARO. The SARO is simply the SPO with the merchant signature. The merchant sends the SARO to the MFI to request the authorization of the transaction. The MFI retrieves the SPO from the SARO and retrieves the SIO from the SPO. It can use the MPK to verify the SIO and it can use the consumer's public key (CPK) to verify the SPO. The MPK is stored in the MFI account record like other merchant account information such as merchant ID, telephone number, address, etc. The CPK can be acquired from the CFI through the CFI's Web service interface. In order to prevent a lot of attacks (e.g., the Man in the Middle fraud aimed at providing a fake key) this type of communication should be secure. Subsequently, the MFI constructs the standard authorization request message and sends it to CFI through the existing financial network. When the MFI receives the au-

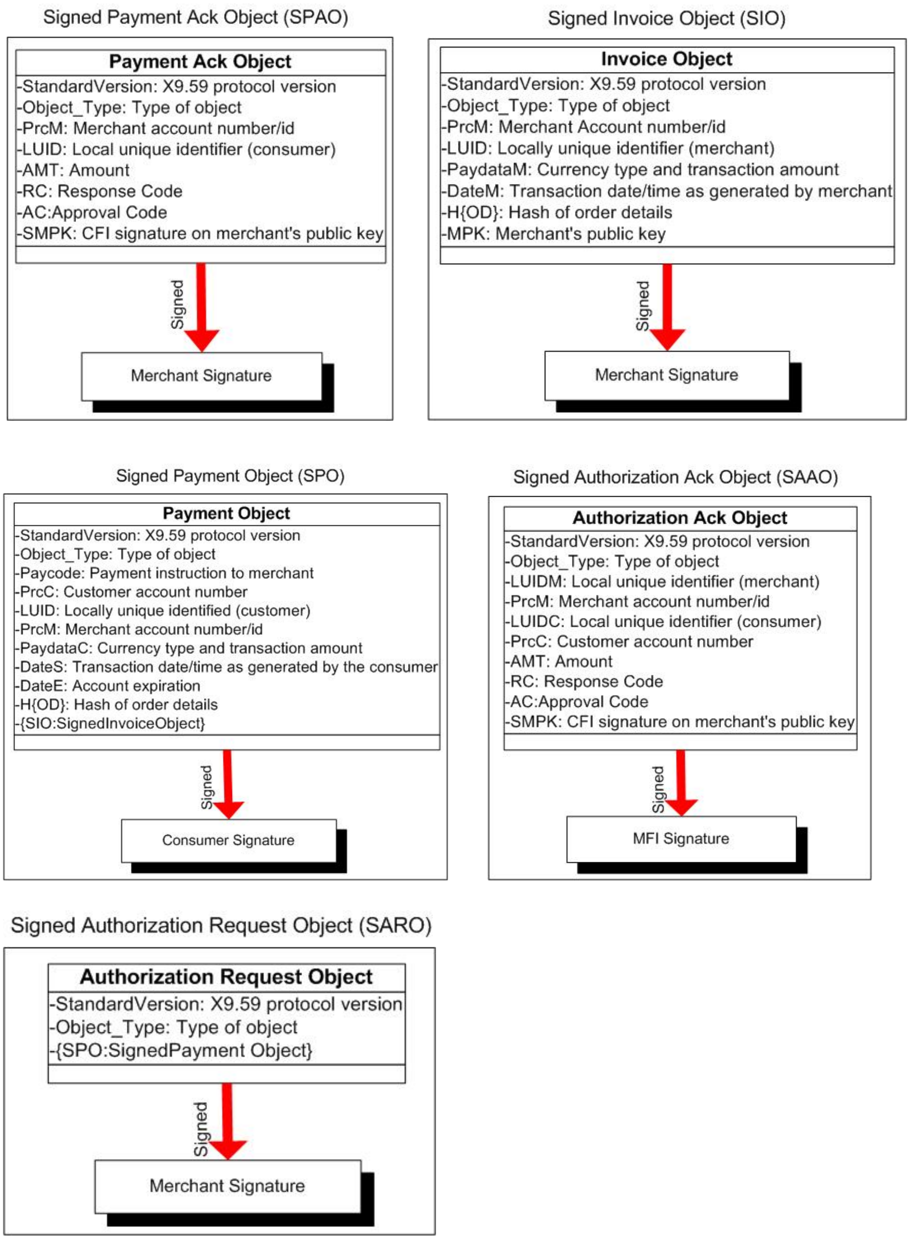
Signed Payment Ack Object (SPAO)
Payment Ack Object
-StandardVersion: X9.59 protocol version
-Object_Type: Type of object
-PrcM: Merchant account number/id
-LUID: Local unique identifier (consumer)
-AMT: Amount
-RC: Response Code
-AC:Approval Code
-SMPK: CFI signature on merchant's public key
Signed
Merchant Signature
Signed Invoice Object (SIO)
Invoice Object
-StandardVersion: X9.59 protocol version
-Object_Type: Type of object
-PrcM: Merchant Account number/id
-LUID: Locally unique identifier (merchant)
-PaydataM: Currency type and transaction amount
-DateM: Transaction date/time as generated by merchant
-H{OD}: Hash of order details
-MPK: Merchant's public key
Signed
Merchant Signature
Signed Payment Object (SPO)
Payment Object
-StandardVersion: X9.59 protocol version
-Object_Type: Type of object
-Paycode: Payment instruction to merchant
-PrcC: Customer account number
-LUID: Locally unique identified (customer)
-PrcM: Merchant account number/id
-PaydataC: Currency type and transaction amount
-DateS: Transaction date/time as generated by the consumer
-DateE: Account expiration
-H{OD}: Hash of order details
-{SIO:SignedInvoiceObject}
Signed
Consumer Signature
Signed Authorization Ack Object (SAAO)
Authorization Ack Object
-StandardVersion: X9.59 protocol version
-Object_Type: Type of object
-LUIDM: Local unique identifier (merchant)
-PrcM: Merchant account number/id
-LUIDC: Local unique identifier (consumer)
-PrcC: Customer account number
-AMT: Amount
-RC: Response Code
-AC:Approval Code
-SMPK: CFI signature on merchant's public key
Signed
MFI Signature
Signed Authorization Request Object (SARO)
Authorization Request Object
-StandardVersion: X9.59 protocol version
-Object_Type: Type of object
-{SPO:SignedPayment Object}
Signed
Merchant Signature

Fig. 3. The five signed objects that are defined in ESCCPS

thorization response message from the CFI, the MFI creates and signs the Signed Authorization Ack Object (SAAO). Before creating the SAAO, the MFI sends the MPK to the CFI's Web service interface and requests for the MPK signing service. The CFI signs the MPK using its private key and sends it back to the MFI. The SAAO includes the signature on the MPK and sends it to the merchant. The merchant creates and signs the Signed Payment Ack Object (SPAO),

Table 3. The mapping between X9.59 and ISO8583

Object	Created by	Verified by	Public Key	
			Stored in	Obtained Through
SIO	Merchant	MFI	MFI	Account Setup
SPO	Consumer	MFI	CFI	CFI's Web Service
SARO	Merchant	MFI	MFI	Account Setup
SAAO	MFI	Merchant	MFI	Account Setup
SPAO	Merchant	Consumer	Merchant	SIO/CFI

which includes the CFI's signature on the MPK, and sends it to the consumer. The consumer verifies the MPK, which is retrieved from the SIO, with the signature from the CFI. If the signature is identical, the consumer can use the MPK to verify the SPAO and complete the process. The contents summary of each object is shown as follows:

$SIO=\{[IO] + [H(IO)]_{SignM}\}$

$SPO=\{[SIO]+[PO] + [H(SIO+PO)]_{SignC}\}$

$SARO=\{[SPO] + [H(SPO)]_{SignM}\}$

$SAAO=\{[AAO] + [H(AAO)]_{SignMFI}\}$

$SPAO=\{[PAO] + [H(PAO)]_{SignM}\}$

The relationship between participants and objects is shown in Table 3. For example, the SPO is created by the consumer and is verified by the MFI. The consumer's public key is acquired from the CFI's Web service.

3.3 Integration Issues

ESCCPS is designed to be a standard application package that is integrated with the existing traditional credit card system using the Web service. ESCCPS will have the interface to access the service provided by the existing systems. The existing traditional credit card system, which is hosted in the MFI, should have the interface to expose the service that is used by the ESCCPS for online authorization. In the banking industry, X.25 is usually used as a communication protocol between the host system and the external system. The ESCCPS can use this interface to utilize the existing financial infrastructure. The Web services have to be included in the ESCCPS as it has features such as the dynamic discovery of services. The Web service broker is an important component in the ESCCPS. The CFI publishes their Web services through the Service Broker (SB). The MFI finds the corresponding Web service through the SB based on criteria and categories, such as the Payment Routing Code, and hence obtains the consumer's public key.

3.4 Advantages of the ESCCPS

The ESCCPS takes full advantage of X9.59 by providing two benefits: first, the ESCCPS does not require addenda records to be added in the ISO 8583 message. In X9.59, the consumer's signature is verified by the CFI so that it requires the Signed Payment Object to be sent to the CFI from the MFI. Thus six records are required to be added into the traditional message. In the ESCCPS, however, most of the verification is done by the MFI. No new records are required to be added into the existing message, such as an authorization request message between the MFI and the CFI. Second, the merchant authentication process is embedded in the payment cycle. In the beginning of the payment process, the merchant sends a Signed Invoice Object (SIO) to the consumer. The SIO is signed by the merchant and includes the merchant's public key. The consumer does not need to verify the SIO. The consumer only embeds the SIO in the Signed Payment Object (SPO) and sends it to the merchant. The merchant constructs the Signed Authorization Response Object (SARO) and sends it to the MFI. The MFI retrieves the SIO and SPO from the SARO and verifies the merchant signature by the merchant's public key, which is stored in the MFI. Moreover, the MFI uses the consumer's public key, which is obtained from the CFI's web service interface to verify the consumer's signature. If it is valid, the MFI creates a standard authorization message and sends it to the CFI. The consumer can authenticate the merchant by using the verified merchant's public key, which is retrieved from the SIO and authenticated by the CFI signature with the MPK, to verify the Signed Payment Ack Object (SPAO).

The ESCCPS is cost effective and interoperable compared to other electronic payment protocols because it utilizes the existing infrastructures and traditional financial messages during the payment cycle. In addition, the ESCCPS provides features that are not found in X9.59. For example, the Signed Invoice Object (SIO), the Signed Authorization Request Object (SARO), the Signed Authorization Ack Object (SAAO), etc. The ESCCPS also provides a method for the consumer's public key transfers from the CFI to the MFI and a method for the merchant's public key transfers from the merchant to the consumer.

4. THE SYSTEM DESIGN OF THE ESCCPS

The ESCCPS is to be designed and developed as a standard application package, which can be implemented by the consumer, merchant, and MFI computer without any modification. The ESCCPS software package can be implemented in both fat and thin e-wallets. The Unified Modeling Language (UML) [15] is used to model and design the proposed system.

4.1 A Typical Electronic Credit Card Payment Architecture

An electronic payment model can be classified into the following three payments models [16]: the electronic credit card payment model, the electronic cash payment model, and the electronic check payment mode. The proposed system of the ESCCPS can be applied to the electronic credit card payment model that involves the following four participants: the consumer, merchant, the consumer's bank, and the merchant's bank. The consumer uses their credit card information to pay the invoice by submitting his/her credit card information to the merchant in an electronic format via some secure payment gateway over a communication network. The merchant obtains the response code from the merchant's bank through the existing payment network or from the Internet depending upon the features of the different electronic payment systems. Processes such as settlement, cleaning, etc. are still based on the traditional credit card payment system.

The architecture of the electronic credit card-based payment system can be conceptually divided into the following three subsystems: the merchant subsystem, the consumer's wallet, and a payment gateway. Typical electronic credit card-based payment architecture is shown in Fig. 4. Generally, the merchant system provides an online shopping cart, payment function, and secu-

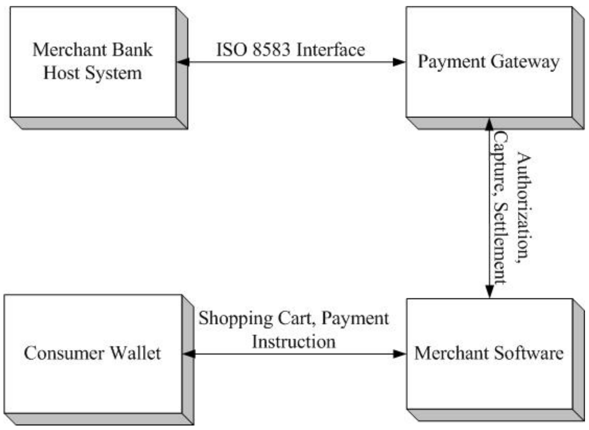
Merchant Bank Host System
ISO 8583 Interface
Payment Gateway
Authorization, Capture, Settlement
Consumer Wallet
Shopping Cart, Payment Instruction
Merchant Software

Fig. 4. Typical electronic credit card-based payment architecture

rity function.

The merchant software is usually hosted on the merchant's Web server. A consumer's e-wallet can be implemented by a fat e-wallet such as the SET e-wallet or thin e-wallet, or a server-based e-wallet such as the NetPay e-wallet [17]. The fat e-wallet stores consumer and payment information in the client's computer. An e-wallet application must be installed in the consumer's computer. The service-based e-wallet stores payment information in a vendor or an agent server. Consumers can access their e-wallets from any computer. The payment gateway is an application service provider that is used for the authorization of payments for e-commerce. The payment gateway is middleware between the merchant and merchant's bank. In general, the payment gateway formats an authorization message using the ISO 8583 financial message standard and sends it to the merchant's bank through an existing channel such as a lease line or by using the Internet.

4.2 System Architecture of the ESCCPS

The implementation of the ESCCPS software will involve the joint collaboration of three participants: the consumer, merchant, and the MFI. The system architecture of the ESCCPS is shown in Fig. 5. A consumer's primary interface in the ESCCPS is with the merchant system. The interface enables the consumer wallet to send and receive a signed object to or from the merchant system. Two interfaces are provided from the merchant system. One is to the card-holder to support electronic payments. The second one is to the MFI to receive the Signed Authorization Ack Object and to send the Signed Authorization Object. The MFI also has access to the Web Service Broker System to find out the Web service provided by the CFI. The MFI then obtains the consumer's public key and a signature on the merchant's public key by invoking the Web service from the CFI. The ESCCPS software application can be seamlessly integrated with the existing systems by using standard communication protocols such as TCP/IP, X.25, etc.

The ESCCPS application is composed of the following six modules: security, key, communication, information, payment, and host system. The standard ESCCPS application package is installed on the participant's computer or server with little changes to the existing system.

The participants such as consumer, merchant, or MFI can use the information module to make

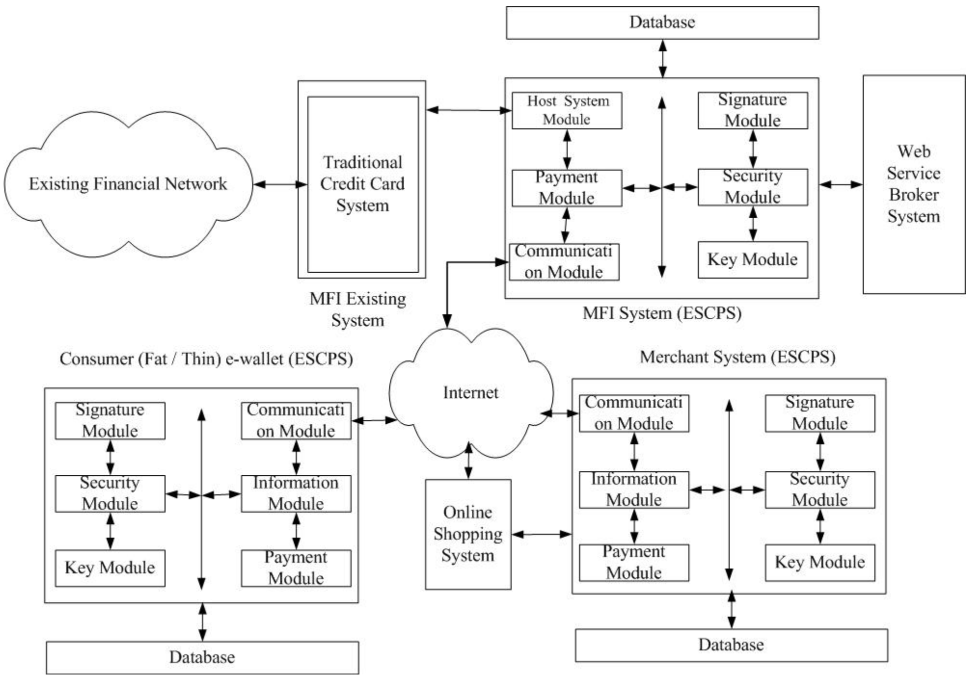

Fig. 5. System architecture of the ESCCPS

queries about transaction information. Participants can obtain the account information, order information, status of the Signed Payment Object, status of transaction, etc. from the participant's information module, such as the consumer information module. The modules of security, signature, and key provide the security services. The security module provides encryption, decryption, and access control functionalities and exchanges secure data between the signature module and the key module. The signature module and the key module cannot directly exchange data with each other. For example, the private key is retrieved from the key module by the security module. Then the security module passes the private key to the signature module for signing the payment object. The signature module cannot acquire the private key from the key module directly. Therefore the security level of the ESCCPS has tighter control. The communication module handles the communication functionality between the consumer's e-wallet, merchant system, MFI system, and Web Service Broker System. The five payment objects defined in the ESCCPS are generated in the payment module. All payment objects are signed or verified in the signature module. The payment module cannot use the signature module directly as it must go through the security module to authorize the signing request. Participants can maintain their existing database systems such as Oracle Database Server, Microsoft SQL Server, MySQL, etc. in the payment module.

The host system module is only implemented in the MFI system, which acts as an interface between the ESCCPS and the traditional credit card system. The existing credit card system can generate an ISO 8583 financial message by using the details of the transaction from the host system module. On the other hand, the existing traditional credit card system, which is hosted in the MFI, should have the interface to expose the service that is used by the ESCCPS for online authorization. In the banking industry, X.25 is usually used as a communication protocol between the host system and the external system. The ESCCPS can use this interface to utilize the existing financial infrastructure. ESCCPS supports two different approaches with the implementation of Web services provided by the CFI, which are loosely coupling and tightly coupling. In the tightly coupling approach, the consumer can use the Uniform Resource Identifier (URI) for the service description directly. In the loosely coupling approach, the consumer can find the service description in the Web Service Broker. In this research, the loosely coupling approach is recommended. Service-Oriented Architecture (SOA) is adopted to achieve loose coupling between interacting software agents. In the SOA, the Web Service broker system serves as a repository or yellow pages that are published by the CFI, which is the service provider. The CFI publishes their Web services to the Web Service Broker. The MFI finds the corresponding Web service through the Service Broker based on criteria and categories, such as the Payment Routing Code, and hence obtains the consumer's public key. The MFI binds and then invokes the Web services and obtains results. The communication module in the ESCCPS uses the Web service interface to find the Universal Description Discovery Integration (UDDI) from the Service Broker. The MFI binds and invokes the CFI's Web service using the UDDI [18].

4.3 The Design of ESCCPS Classes

The ESCCPS is designed with object-oriented methodology. Five core classes are defined in

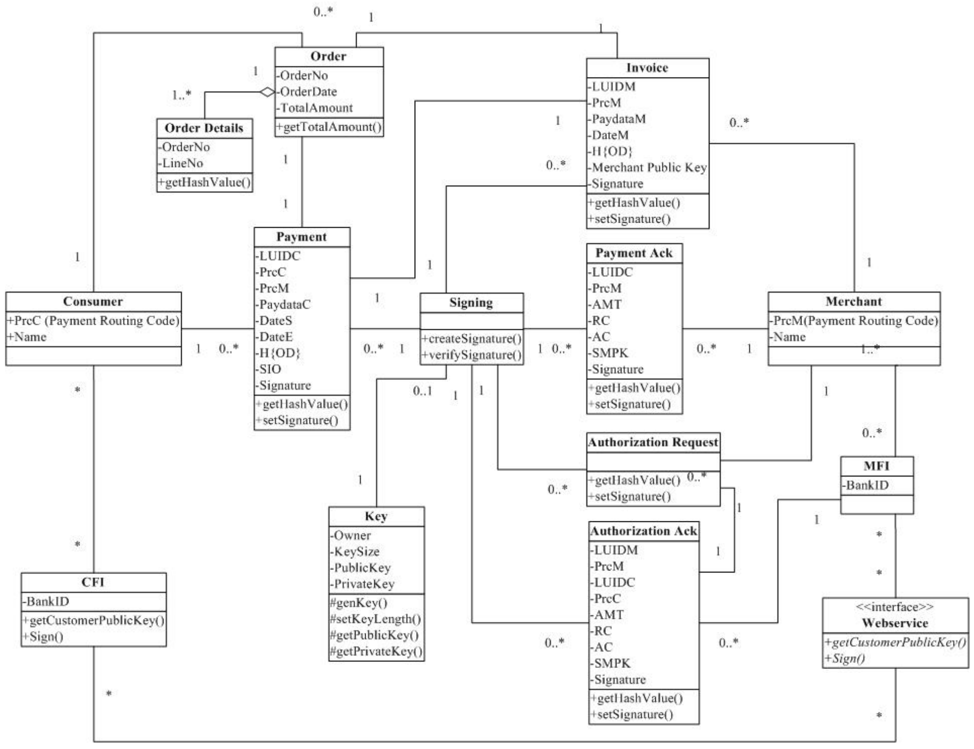

Fig. 6. Class diagram of the ESCCPS in UML

ESCCPS including the Invoice Class, Payment Class, Authorization Request Class, Authorization Ack Class, and the Payment Ack Class The class diagram is shown in Fig. 6. In order to protect the system, those objects must be signed by the signing object before being used in the payment cycle. In addition, each class must be authenticated before using the methods in the signing class. In the ESCCPS, an unsigned object will be changed to a signed object by assigning a value to the signature attribute. For example, the unsigned invoice object uses the getHashValue method to get a hash value, which is calculated by all the attributes of the invoice object excluding the signature attribute. The signing object then generates a signature by giving the hash value of the invoice. After that, the unsigned invoice object is transformed to the Signed Invoice Object (SIO) by assigning a signature. The class diagram is shown in Fig. 6.

4.4 The Event Flow of the ESCCPS

The sequence of actions and the flow of messages or events among the objects or components of the system are shown in Fig. 7. The system consists of the following five participants: the consumer, merchant, MFI, CFI, and traditional payment system. The Web Service Broker keeps the service description in a service registry and allows for the looking up of service provider interfaces and server locations. The MFI looks up Web services in the Service Broker and invokes the Web service through the UDDI. The CFI's Web service has two major functions. First, it obtains the consumer's public key, which is used for verifying the SPO from the CFI, and delivers it to the MFI. Second, it provides a signing function to the MFI for signing the merchant's public key in which a consumer can verify the merchant's public key that was acquired in the SIO. The event starts at the top left, with the consumer sending an initial buy request to the merchant object. The merchant generates a Signed Invoice Object (SIO) and sends it to the

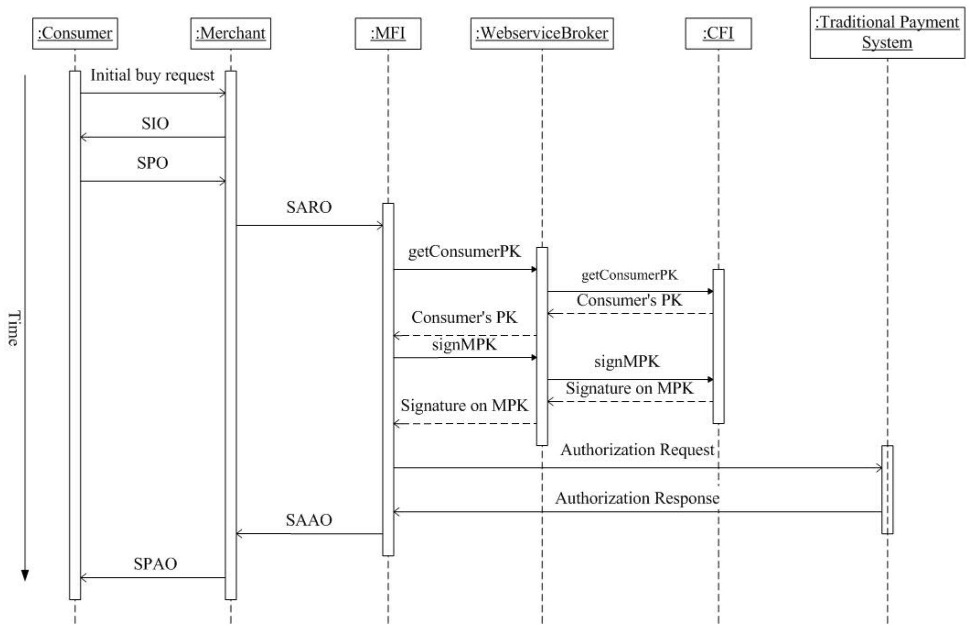

Fig. 7. Diagram showing the sequences of the payment flow in the ESCCPS

consumer. At that point, the consumer does not verify the SIO. The consumer generates a Signed Payment Object (SPO) and sends it to the merchant. The Signed Authorization Request Object (SARO) is generated by the merchant by the use of the SIO and SPO and sends it to the MFI. The MFI verifies the SARO using the consumer's public key which is obtained from the CFI's Web Service. After all the verification processes, the MFI sends the Signed Authorization Ack Object (SAAO) to the merchant. At this point, the merchant generates the Signed Payment Ack Object (SPAO) and sends it to the consumer as having been either approved or declined.

4.5 ESCCPS Messages

The ESCCPS messages are based on ISO 8583 and X9.59. A message consists of a Message Header and Signed Object. The message structures are shown in Fig. 8. The message file length can be fixed or variable. The Message Header is divided into the two elements of Header and Object Type. The second component is the Signed Object, which is divided into three parts: Field Maps, Data Fields, and Signature. The Field Maps specify which data fields are present. The second part is the Data Fields, which is the stored payment information, such as the PRC, transaction amount, etc. It also describes the length of each variable-length data field. A subfield is defined before the content of the variable-length data field. The last part is the Signature, which is the message digest of the Field Maps and Data Fields, and is signed by the owner.

All of the data types in the ESCCPS message are encoded using an 8-bit UCS/Unicode Transformation Format (UTF-8). UTF-8 is a variable-length character encoder for Unicode. It uses one to four bytes per character depending on the Unicode symbol. One byte is used to encode the first 128 ASCII (American Standard Code for Information Interchange) characters. Two to four bytes are used to support international languages such as Chinese, Arabic, Greek, etc.

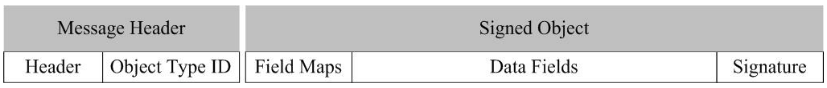

Fig. 8. Message structures of ESCCPS

4.5.1 Message Header

The Header contains the header length, the total length of the message, communication information, and other system-related processing information. Table 4 shows an example of a message header. The Header is located at the beginning of the ESCCPS message.

The Object Type describes the types of objects. It is located after the message header. Five signed objects are defined in the ESCCPS. Table 5 shows the value of the five types of objects defined in an ESCCPS message. The Object Type ID uses 1 byte to store the value with a range of 1 to 255. Therefore, the maximum number of objects is 255. The Object Type ID identifies

Table 4. Typical structure of a message header

Field No	Field Description	Length	Example
1	Version number	1 byte	1
2	Total length of header	1 byte	200
3	Total message length	2 bytes	1200
….	System-Related Processing Information	….	….

Table 5. The five types of objects defined in the ESCCPS

Object Type ID	Object Name	Length (Bytes)
1	Signed Invoice Object	1
2	Signed Payment Object	1
3	Signed Authorization Request Object	1
4	Signed Authorization Ack Object	1
5	Signed Payment Ack Object	1

the processing requirement and indicates the content of the message.

4.5.2 Signed Object

The signed object has the following three components: Field Maps, Data Field, and Signature. The Field Maps specify which fields are present in the message and which fields are not. Fig. 9 shows a typical example of Field Maps. It uses one byte to represent eight fields whether the data field is present or not. The Field Map is a binary representation of data presence in an 8-bit byte.

If a bit is “1,” it means the corresponding data field is present. Otherwise, it is not present. In this example, the first bit is 1. It means that field 1 is present. The second bit is 0, meaning that field 2 is not present. The third, fourth, and fifth are 1, meaning the data fields 3, 4, and 5 are present. The last three bits are 0, meaning the fields 6, 7, and 8 are not present in the message. 3 bytes or 24 bits are used for the Field Maps to indicate whether 24 data fields defined in the ESCSP message are present or not.

The second component of the Signed Object is the Data Field. It is used to store the payment-related information. The field content may be fixed-length or variable-length, depending on each field attribute. Many of the fields are fixed-length. A subfield is required for a variable-length field. It is located at the beginning of the variable-length field and uses two bytes to describe the length of the variable-length field. No fields can exceed 65,535 positions. The key data field attribute is shown in Table 6. F and V in the Length Type column are used to indicate which data field is fixed-length (F) or variable-length (V). If the data field is variable-length, a subfield is required and located before the data field.

Some data fields are defined as variable-length types, such as fields 6, 7, 15, 16, etc. A 2 bytes subfield is used to indicate the length of the data field. The value in the subfield does not include its own length. One of the advantages of using a variable-length field is that it provides flexibility for participants. For example, “field 15,” is used to store the merchant’s public key in the

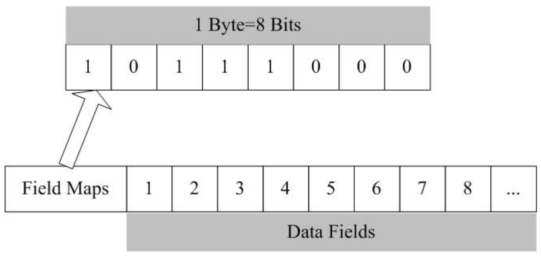

Fig. 9. Field mapping example

Table 6. Key data field attributes

Field No.	Field Name	Length Type	Length (Byte)	Format
1	Type of Object	F	1	1N
2	Payment Routing Code (PRC)	F	19	19N
3	PRC Expiration Date	F	4	YY/MM
4	Acquirer Locally Unique Identifier	F	12	12N
5	Merchant Account Number	F	6	6N
6	Merchant Locally Unique Identifier	V	<=20	<=20N
7	Consumer Locally Unique Identifier	V	<20	<=20N
8	Currency Type	F	2	2N
9	Conversion Rate	F	4	6AN
10	Transaction Amount	F	12	12AN
11	Local Date	F	8	YYYY/MM/DD
12	Local Time	F	6	hhmmss
13	Response Code	F	2	2N
14	Approval Code	F	6	6N
15	Merchant’s Public Key	V	<=512	<512AN
16	CFI Signature on Merchant’s Public Key	V	<=512	<512AN
17	Hash of Order Details	F	16	16AN
18	Signed Invoice Object (SIO)	V	<=1,000	<=1,000AN
19	Signed Payment Object (SPO)	V	<=1,000	<=1,000AN
20	Reserve	V		
21	Reserve	V		
22	Reserve	V		
23	Reserve	V		
24	Signature	V	<=512	<512AN

Legends: N = Numeric digits 0-9, AN = Alphabetic and numeric char.

Signed Invoice Object. The maximum key length to be supported in field 15 is 4,096 bits (512 bytes). In addition, it can be used to store the different key length of the merchant’s public key such as 1,024 bits, 2,048 bits, etc. It can be integrated with the participants' existing infrastructure without a lot of changes. The advantage of a variable-length is that it can eliminate the transmission of unnecessary filled characters. The last component of a Signed Object is a signature. The system uses a cryptographic hash function to calculate the hash value or message digest of the Field Maps and Data Fields together. A private key is used to encrypt the hash value. Then, the cipher text of the hash value is created. This signature is located at the end of the ESCCPS message.

Three message examples are shown in Fig. 10. The first example is a Signed Invoice Object (SIO). The object ID in the message header is set to “1,” indicating that it is the SIO. The value of the Field Maps is used to specify which data fields are present. In this example, 11 bits are set to 1. This means that there are 11 data fields present. The positions of 1, 5, 6, 8, 9, 10, 11, 12, 15, 17, and 24 in the Field Maps are set to 1. The corresponding data fields are shown in the message. In the second example, the SIO is embedded in the Signed Payment Object (SPO). The consumer signs the SPO before sending it to the merchant. The SIO is stored in field 18 of the SPO. The content of field 18 includes Field Maps, Data Fields and the SIO signature. The last

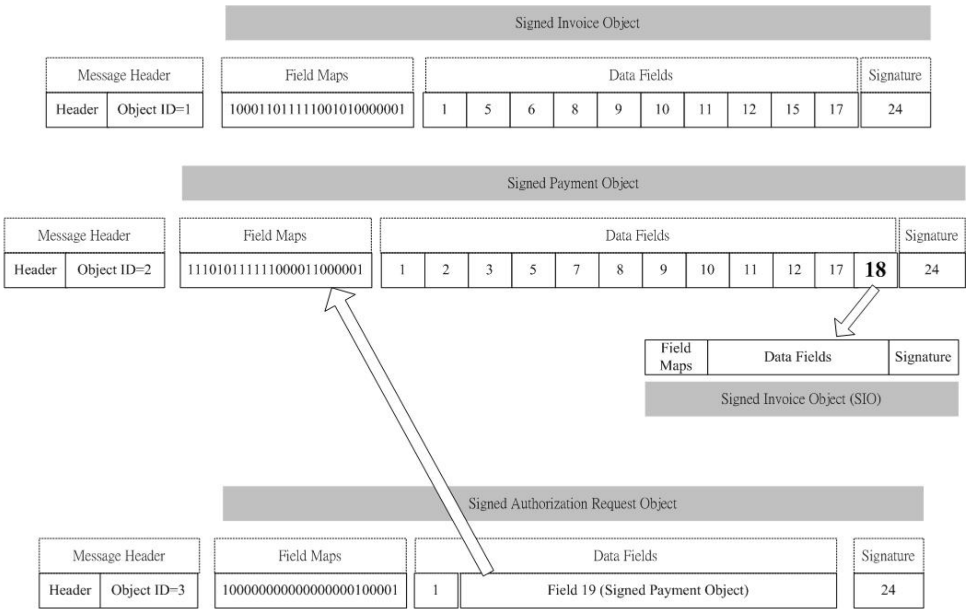

Fig. 10. Examples of an ESCCPS message

example also embeds the SPO in the Signed Authorization Request Object (SARO). The SARO is sent to the MFI through the merchant. Therefore, the MFI can obtain the content of the SPO and also the SIO.

5. PERFORMANCE EVALUATION OF THE ESCCPS

Performance is one of the vital factors for evaluating an electronic payment system. The processing time is a key indicator for measuring the performance of an electronic payment system. In the evaluation, the simulation tool, Holosofx Workflow BPR, was chosen to develop three simulation models. One is the typical payment system that uses CBDS, which is the SET. The others are the ABDS payment system, the X9.59, and the proposed system--the ESCCPS. For simplicity, each model assumes that only one merchant and one consumer is involved in the payment cycle. The consumer makes a payment to the merchant. The merchant payment system negotiates with the consumer e-wallet using the selected payment protocol. The merchant generates an authorization request message and sends it to the MFI or payment gateway through the existing payment infrastructure. The authorization response message will be sent back to the MIF as having been either approved or declined. The merchant receives the response from the MFI and sends it to the consumer through one of the payment systems' specifications.

Each simulation model will be tested by a different workload. The workload is the consumer who generates a payment request that is measured in average arrival rates or the number of requests per unit time, which are randomly selected at the time based on an exponential distribution. The exponential distributions are a class of continuous probability distribution. It is often used to model the time between independent events that happen at a constant average rate. The average cycle time and average process time are selected for the performance evaluation in each simulation model. The process time is the amount of system working time spent on the process during a job within a simulation. The Cycle time is the total time from the beginning to the end of the entire process. It includes the process time and the delay time.

5.1 System Parameters and Assumptions

The tasks defined in each simulation model are based on each payment protocol specification. Examples are the message flow, the number of participants, the number of object signing processes, and the number of signature verification processes. Each model has processed 10,000 transactions for different workload situations (e.g., from an average of 15 requests per minute to 23 requests per minute). The average cycle time and average process time are selected for the performance evaluation in each simulation model.

Input parameters are a vital component in the simulation model. Many parameters are used to build simulation models. Each task will consume a specific amount of time to complete the process, such as the time consumed in the generation of a digital signature, signature verification, etc. A Java Cryptographic Extension (JCE) [19] package was used to estimate the time consumed for each task. The experiments were implemented on an Intel Core2Dual E6300 computer. The length of the key, which is used for all PKI services, is 4,096 bits. Secure cryptographic hash functions (e.g., SHA-*, RIPEMD-*, etc.) can be used in the message digest. In the symmetric encryption / decryption, the Triple DES algorithm was used and the key length was 128 bits.

There are some assumptions made in the simulation modes. The network traffic and the communication time between each participant are not included. The communication time within the existing financial network is not considered. All the tasks between each party defined in the three models are processed automatically. The starting point of the three models is at the moment right after the cardholder initiated the check-out function. Therefore, the onsite shopping processes are not included in the simulation models.

5.2 The Simulation Models: SET, X9.59, and ESCCPS

The SET simulation model involves three participants and two external processes in the payment cycle. All the tasks provided from other participants who are interacting with the SET system such as the acquirer, issuer, and certificate authority are grouped into the two external processes of the existing credit card payment process and the certificate verification process. The simulation model begins with the consumer generating the Payment Initial Request (PinitReq) message and sending it to the merchant. The SET has seven messages.

The results, which are shown in Fig. 11, were collected after processing 10,000 transactions. 9 workloads were simulated from an average of 15 requests per minute to 23 requests per minute. The cycle time and process time rapidly increase when the consumer's e-wallet generates 23 payment requests per minute to the merchant. In this workload, the merchant software utilization is increased to 99%. The SET simulation model requires about 7 hours and generates 10,000 transactions in the workload at 23 payment requests per minute.

Unlike SET, X9.59 has fewer participants. It only has the cardholder, merchant, and MFI. In addition, an external system is involved in the simulation model (i.e., the traditional payment system.) The traditional payment system is the existing credit card-based system that is for face-to-face retail transactions and it is initiated by the merchant or the acquirer. Two signed payment

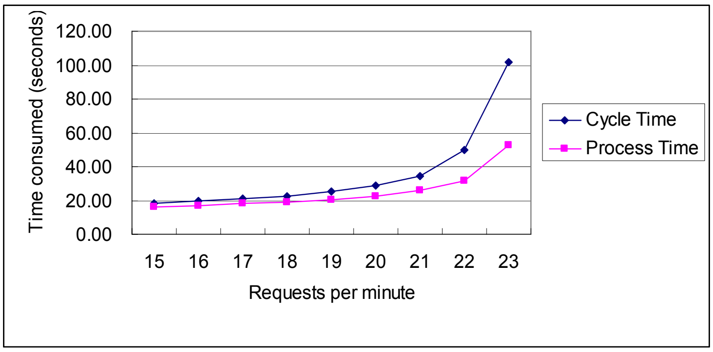

Fig. 11. Time taken in the SET simulation model

objects and one external process are involved in the X9.59 simulation model. Payment objects include the Signed Payment Object (SPO) and the Signed Ack Object (SAO), which are used between the consumer and merchant. The functions of the external process include handling the authorization request message sent by the merchant, verifying the transaction, and returning a response code to the merchant to either approve or decline it. (See Fig. 12 to see the results of the X9.59 simulation model.) X9.59 has fewer messages compared with SET and ESCCPS. Therefore, the time consumed for the payment process is less than the others. However, it does not provide sufficient security schemes to protect the payment process.

5 signed payment objects, 3 participants, and 2 external processes are involved in the ESCCPS simulation model. The consumer, merchant, and MFI are the participants in the ESCCPS. The system begins with the Signed Invoice Object (SIO), which is sent to the consumer from the merchant. Two external processes, which are the traditional payment system and the web service system, cooperate with the ESCCPS. The traditional payment system is the existing credit card-based payment system that is used for physical card payment authorization. The web service system is used to obtain the consumer's public key from the CFI and for signing merchant's public key. The minimum process time to compete one payment request from the ESCCPS is about 6.03 seconds. The simulation results of the ESCCPS are shown in Fig. 13.

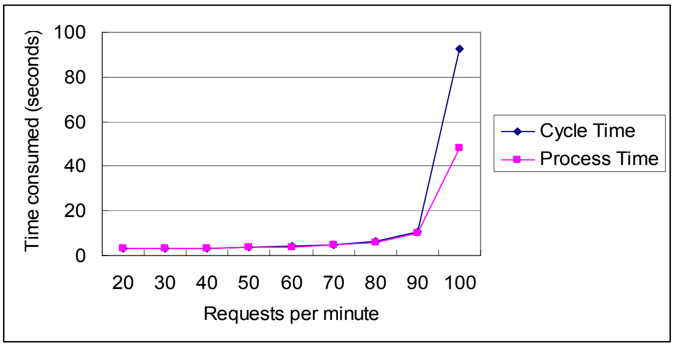

Fig. 12. Time taken in the X9.59 simulation model

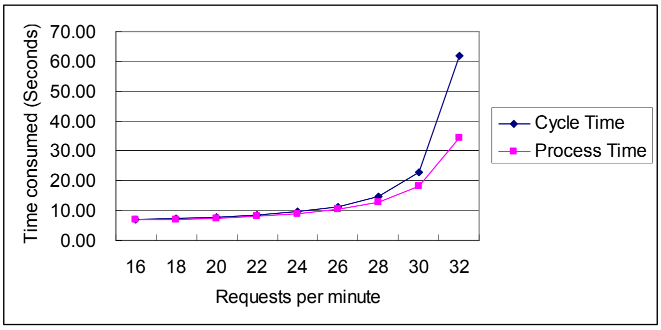

Fig. 13. Time taken in the ESCCPS simulation model

5.3 Performance Comparison

The statistics are collected after 10,000 transactions have been processed in each model and in each workload situation. Fig. 14 summarizes the average cycle time and process time comparison using the 3 payment protocols with 9 kinds of workloads. The ESCCPS is about 200% faster than SET with a workload of 18 requests per minute. Although X9.59 is the fastest payment protocol from among the three simulation models, it does not provide all of the basic security schemes such as merchant authentication, etc.

Fig. 15 shows the percentage faster of the ESCCPS compared to SET in cycle time and process time. The ESCCPS shows a performance gain of 166% to 1,026% as compared to SET for 15 to 23 requests per minutes, respectively. X9.59 is the fastest payment protocol, but it has fewer desirable features than the other three payment protocols. An evaluation of the three simulation models is shown in Table 7. The ESCCPS provides a similar functionality to SET but the processing time of the ESCCPS is faster than SET. The cost of the ESCCPS for system implementation, system maintenance, and operation is less than SET because the CA is not involved.

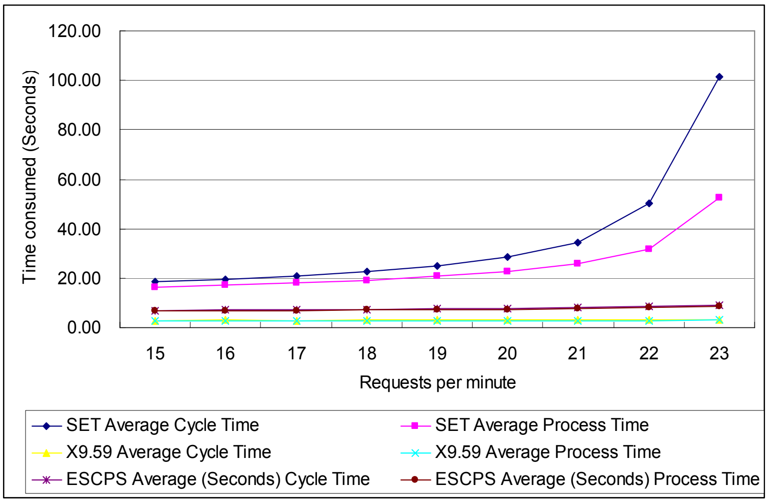

Fig. 14. Comparison of the amount of time taken for SET, X9.59, and ESCCPS

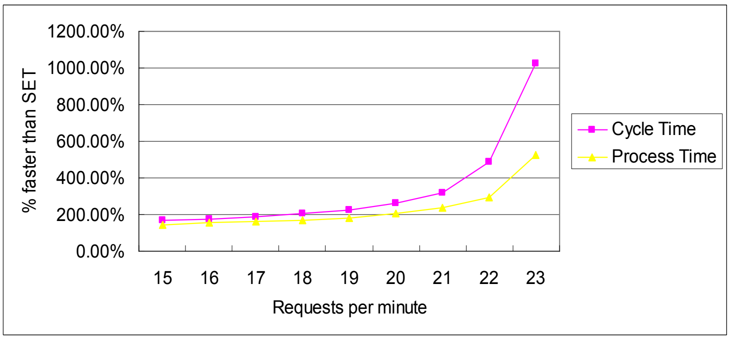

Fig. 15. The percentage faster of ESCCPS as compared to SET

Table 7. Evaluation of the three payment protocols

	SET	**X9.59**	**ESCCPS**
Identification	Yes	Yes	Yes
Confidentially	Encryption	Using PRC	Using PRC
Authentication:			
Cardholder	Yes	Yes	Yes
Merchant	Yes	No	Yes
Integrity:			
Cardholder	Yes	Yes	Yes
Merchant	Yes	No	Yes
Non-repudiation:			
Cardholder	Yes	Yes	Yes
Merchant	Yes	No	Yes
CA involved	Yes	No	No
New fields added to the existing financial message	No	Yes	No
Cost	High	Low	Low

Hence the ESCCPS is an efficient payment system. It balances security, simplicity, and cost for each party. The ESCCPS becomes faster and faster than SET as the number of payment requests are increased.

6. Conclusion

This paper proposed a new credit card-based payment system called the Efficient and Secure Credit Card based Payment System (ESCCPS). It is based on the ANSI X9.59-2006 standard. The ESCCPS supports many security schemes such as identification, confidentiality, authentication, integrity, and non-repudiation. The ESCCPS is also an effective payment system because it fully utilizes the existing payment infrastructures and provides security schemes that are on par with SET. Unlike the X9.59 standard, it does not require any changes to be made in the existing payment infrastructure and it also provides a way for the MFI and consumer to authenticate each participant during the payment cycle. Although the ESCCPS is designed for the B2C (Business to Consumer) payment model, it can be used for the B2B (Business to Business) and B2G (Business to Government) payment models with some modifications. Moreover, the following three different payment methods: SET - secure but slow; X9.59 – not secure but fast; and the ESCCPS - both relatively secure and fast - are compared by simulation. The results demonstrate that the performance of the ESCCPS is almost the same as X9.59 and outperforms SET. In conclusion, the ESCCPS, which offers both a reasonable level of security and efficiency, is believed to be a good payment protocol to support and facilitate the exponential growth of e-commerce and m-commerce.

REFERENCES

[1] Jean-Michel SAHUT, "*Internet Payment Solutions: Comparative evaluation and key factors of success*", in Proceedings of the *2005 Symposium on Applications and the Internet Workshops (SAINT-W'05), 2005 IEEE.*

[2] Francesco Buccafurri, Gianluca Lax, "Implementing disposable credit card numbers by mobile phones", *Journal of Electronic Commerce Research*, Volume 11 Issue 3, September, 2011, pp.271-296.

[3] *PayPal Merchant Services*. Available at *"http://www.cybercash.com/"* retrieved on 17 Sepetmeber, 2007.

[4] *Visa and MasterCard. SET Secure Electronic Transaction Specification Book 1*: Business Description, May, 31, 1997.

[5] Mihir Bellare, Juan A. Garary, Ralf Hauser, Amir Herzberg, Hugo Krawczyk, Michael Steinerm Gene Tsudik, Michael Waidner, "iKP – A Family of Secure Electronic Payment Protocols", July 12, 1995.

[6] *Visa Company*. Verified by Visa. Available at "*http://www.visaasia.com/ap/sea/merchants/productstech/vbv_implementvbv.shtml"* retrieved on 17 September, 2007.

[7] Albert Levi and Cetin Kaya Koc, *"CONSEPP: Convenient and Secure Electronic Payment Protocol Based on X9.59"*, in Proceedings 17th the *Computer Security Application Conference, 2001, (ACSAC 2001) IEEE*

[8] Anne & Lynn Wheeler, *X9 Financial Standard Information*, *"http://www.garlic.com/~lynn/"* retrieved on 17 September, 2007.

[9] American National Standard X9.59-2006, Electronic Commerce for the financial Services Industry: Account Based Secure Payment Objects, May, 24, 2006.

[10] Anne & Lynn Wheeler. *Account Authority Digital Signature Model* at *"http://www.garlic.com/~lynn/aadsover.html"* retrieved on 17 September, 2007.

[11] Burdett, D. Request for Comments: *2801- Internet Open Trading Protocol- IOTP Version 1.0., Networking Working Group, The Internet Society, 2000*. Available at *"http://www.rfc-editor.org/rfc/rfc2801.txt"* retrieved on 17 September, 2007.

[12] *Open Financial Exchange*. Available at *"http://www.ofx.net/ofx"* retrieved on 17 September, 2007.

[13] R. Housley, W. Ford, W. Polk, D. Solo. Request for Comments: *2459 - Internet X.509 Public Key Infrastructure Certificate and CRL Profile, Networking Working Group, The Internet Society 1999.* Available at *"http://www.ietf.org/rfc/rfc2459.txt"* retrieved on 17 September, 2007.

[14] ISO 8583-1:2003 Financial transaction card originated messages – Interchange message specifications – Part 1: message, data elements and code values.

[15] *Object Management Group OMG. Unified Modeling Language*, UML Resource Page. Available at *"www.uml.org"* retrieved on 17 September, 2007.

[16] Bo Meng, Qianxing Xiong. *"Research on Electronic Payment Model"*, in Proceedings of the *IEEE International Conference on Computer Supported Cooperative Work in Design*, Vol.1, 26-28 May, 2004, pp.597-602.

[17] Xiaoling Dai1, John Grundy, *"Three Kinds of E-wallets for a NetPay Micro-Payment System"*, in

Proceedings of *5th International Conference on Web Information Systems Engineering*, Lecture Notes in Computer Science, Springer-Verlag, Brisbane, Australia, November 22-24, 2004, pp. 66-67

[18] M.Gudgin, M.Hadley, N.Mendelsohn, J.Moreau, H.F.Nielsen, A.Karmarkar & Y.Lafon, *SOAP Version 1.2 Part 1: Message Framework W3C, 2007*. Available at *"http://www.w3.org/TR/soap12-part1/"* retrieved on 17 September, 2007.

[19] *Sun Microsystems, Inc.. Java(TM) Cryptography Extension (JCE) in J2SE [Online]*. Available: *"http://java.sun.com/products/jce/ retrieved on 17 September 2007"*.

Chi Po Cheong

Mr. Cheong is currently pursuing a PhD degree from the School of Science and Technology at the University of Sussex in Brighton, UK. In 2007, Mr. Cheong graduated with a Master of Science with a major in E-Commerce Technology from the University of Macau.

Simon Fong

He graduated from La Trobe University in Australia, with a First Class Honours BEng, a Computer Systems degree, and a PhD. Computer Science degree in 1993 and 1998 respectively. Simon is now working as an Assistant Professor in the Computer and Information Science Department at the University of Macau. He is also one of the founding members of the Data Analytics and Collaborative Computing Research Group in the Faculty of Science and Technology. Prior to joining the University of Macau, he worked as an Assistant Professor in the School of Computer Engineering at Nanyang Technological University in Singapore. Before his academic career, Simon took up various managerial and engineering posts, such as being a systems engineer, IT consultant, integrated network specialist, and e-commerce director in Melbourne, Hong Kong, and Singapore. Some companies that he worked at before include Hong Kong Telecom, Singapore Network Services, AES Pro-Data, and the United Overseas Bank in Singapore. Dr. Fong has published over 150 peer-reviewed international conference and journal papers, mostly in the area of e-commerce technology, business Intelligence, and datamining.

Pouwan Lei

Dr. Pouwan Lei is a Research scientist, working at the Information Technology Group in the School of Engineering, Design, and Technology at the University of Bradford in West Yorkshire, UK.

Chris Chatwin

Professor C. R. Chatwin holds the Chair of Engineering, University of Sussex, UK; where, inter alia, he is Research Director of the "iims Research Centre" and the Laser and Photonics Systems Engineering Group. At Sussex he is a member of the University: Senate, Council and Court. He has published two research monographs: one on numerical methods, the other on hybrid optical/digital computing - and more than two hundred international papers. Professor Chatwin is on the editorial board of the International Journal "Lasers in Engineering." He is also a member of the Institution of Electrical and Electronic Engineers, the British Computer Society, and the Association of Industrial Laser Users. He is a Chartered Engineer, Euro-Engineer, International Professional Engineer, Chartered Physicist, Chartered Scientist, and a Fellow of The Institution of Electrical Engineers, The Institution of Mechanical Engineers, The Institute of Physics, and The Royal Society for Arts, Manufacture, and Commerce.

Rupert Young

Dr. Rupert Young currently is serving as a Reader in Engineering (Engineering and Design) and Optical and Medical Imaging (Biomedical Engineering) at the University of Sussex in Brighton, United Kingdom. Dr. Young obtained his BSc degree and PhD degree from the University of Glasgow, UK.