Copyright © 2012 KSII

A Fair-Exchange E-Payment Protocol For Digital Products With Customer Unlinkability

Yi-Chung Yen[1], Tzong-Chen Wu[1], Nai-Wei Lo[1] and Kuo-Yu Tsai[2]
[1]Department of Information Management, National Taiwan University of Science and Technology
No.43, Sec. 4, Keelung Rd., Da'an Dist., Taipei 106, Taiwan (R.O.C.)
[2]Department of Management Information Systems, Hwa Hsia Institute of Technology
No.111, Gongzhuan Rd., Zhonghe Dist., New Taipei City 235, Taiwan (R.O.C.)
[e-mail: d9609303@mail.ntust.edu.tw, {tcwu, nwlo}@cs.ntust.edu.tw, kytsai@cc.hwh.edu.tw]
*Corresponding author: Nai-Wei Lo

Received April 9, 2012; revised October 19, 2012; accepted November 7, 2012; published November 30, 2012

Abstract

Along with the development of Information Technology, online transactions through Internet have become more popular for the reasons of convenience and efficiency. In order to provide secure and reliable online transactions, an effective electronic payment protocol is crucial. In this paper, we propose a novel electronic payment protocol for digital product transactions with an offline arbiter to achieve fair exchange, automated dispute resolution, customer anonymity, and customer unlinkability. In our protocol a product token is adopted to eliminate the need of key management for digital product decryption in the offline arbiter. In addition, Elliptic Curve Cryptography (ECC)-based self-certified public key is utilized to further reduce computing overheads. According to our analysis, the efficiency of our protocol can be greatly increased in comparison with previous literatures.

Keywords: Fair-exchange, E-payment, anonymity, customer unlinkability, self-certified public keys

The authors gratefully acknowledge the support from National Science Council, Taiwan, under the Grants No. NSC 100-2219-E-011-002. The authors also grateful acknowledge the helpful comments and suggestions of the reviewers.

http://dx.doi.org/10.3837/tiis.2012.10.012

1. Introduction

In general, participants of an electronic payment (e-payment for short) protocol for transacting digital products consist of a customer, a merchant, a bank, and an arbiter, whose role is responsible to resolve a transaction dispute. It is crucial for an e-payment protocol to achieve fair exchange, dispute resolution, customer anonymity, and customer unlinkability. The property of fair exchange ensures that the customer does not get the product or cannot use it unless he/she pays for it, and meanwhile, the merchant does not have any remuneration unless the customer actually receives the product as expected [1][2][3][4][5][6]. True fairness allows the arbiter to undo a transfer of the item or produce a replacement to the participant (customer or merchant) [1]. Dispute resolution assures the enforcement of fair exchange if a dispute occurs, such that none of the participants will get any unfair advantage using an e-payment protocol. Various proposed fair exchange e-payment protocols [1][2][3][5][7][31][32] are designed to provide automated dispute resolution without manual intervention. The purpose of customer anonymity is to conceal customer's identity and his personal information from being maliciously disclosed by an eavesdropper or revealed by a merchant [8][9][10][11][12]. Nevertheless, a customer can still be traced based on fixed individual-related value used in online transactions, such as his encrypted bank account or his public key. Therefore, customer unlinkability does not offer in previous works. Designing an efficient and secure e-payment protocol with these four properties of fairness, dispute resolution, customer anonymity, and customer unlinkability is a very challenging task.

Elaborated from Tsaur's ECC-based self-certified public key cryptosystem [13], this study proposes an e-payment protocol for digital products with fair exchange, automated dispute resolution, and customer anonymity. As a result, the customer is assured to get decryption keys of purchased digital products through an automated dispute resolution process. In terms of privacy protection, our protocol preserves the customer's identity within each transaction to achieve customer unlinkability. We utilize a product token to eliminate the overhead of decryption key management on the arbiter. Therefore, our protocol avoids the storage scalability issue and consequently reduces the chance for an arbiter to be the attack target by an adversary. By using the self-certified public keys in our protocol, computing overhead on e-payment protocol is further decreased to improve execution efficiency. We organize the rest of this paper as follows. In Section 2 and 3, literature review and cryptographic techniques applied in our protocol are introduced. Our protocol is described in Section 4. Finally, the discussion and conclusion are given in Section 5 and 6, respectively.

2. Literature Review

Early proposed e-payment protocols for digital products designed an online arbiter [2][7][14]. In these protocols, all transactions are forwarded and verified through the online arbiter so that the fair-exchange property is easily achieved. However, the online arbiter must be involved in the entire exchange process and often becomes the bottleneck of communication traffic. Asokan *et al.* [1] first proposed an optimistic protocol for fair exchange that adopts an offline arbiter. Since the offline arbiter is only involved in the presence of network or system failures or if a dispute occurs, such an optimistic approach can avoid the forementioned bottleneck problem. However, Asokan *et al.*'s protocol did not evaluate the integrity of the received product. A customer may get a product which is not what he expects.

Ray and Ray [3][8] proposed two enhanced fair-exchange e-payment protocols that ensure confidentiality and integrity of sold products by using an asymmetric cryptosystem. The Ray-Ray protocol allows the customer to verify whether the received product is what he paid for. However, the customer has to download encrypted digital products from both the arbiter and the merchant side. Therefore, communication overhead is much heavier than other e-payment protocols. Another problem inherent in Ray-Ray protocol is that the arbiter also serves as a warehouse for storing selling products registered by the merchant. The arbiter will suffer from storage management issue as the volume of the registered products increases rapidly. Lately, Nenadic *et al.* [4] proposed another fair-exchange e-payment protocol to enhance the feature of product validation by using a non-interactive verifiable and recoverable signature encryption scheme, such that product validation can be performed efficiently in an offline manner. However, in Nenadic *et al.*'s protocol, a certificate based on X509 standard [15] is associated with each product and each customer. As a result, the extra cost of certificate management and computing resources for certificate validation are required.

Previously proposed fair-exchange e-payment protocols adopted different approaches to reduce the storage space for decryption keys with respect to the digital products offered by the merchants [5][7][9][16][17]. In these previous works, the arbiter must generate and store the decryption key for each digital product to resolve any dispute between the merchant and the customer. This implies that the required storage space for decryption keys in arbiter is proportional to the number of digital products offered by the merchants. Oniz *et al.* [9] introduced the concept of chain keys to reduce the storage space in arbiter, in which the arbiter only stores and uses a root key and a single HMAC key to generate multiple decryption keys for the offered products with respect to each merchant. However, computation overhead in arbiter is proportional to the number of digital products registered. If one merchant registers a lot of digital products in the arbiter, the arbiter in average needs to spend more computing resource to generate a corresponding decryption key per digital product for a customer while a dispute occurs. Alaraj and Munro [5] also proposed a fair exchange protocol using shared key pairs between the arbiter and the merchant to encrypt/decrypt digital products. This enables the arbiter to have the same ability as the merchant, which is to provide a decryption key for the customer. However, the customer needs to download two certificates: one is a product certificate, and the other is shared between the merchant and the arbiter during online transactions. This will increase both communication overhead during certificate delivery and computing overhead in certificate validation. In addition, the digital product is encrypted by an asymmetric key. Therefore, computing overhead is heavy, especially for large-volume digital products. Recently, more fair exchange protocols [10][16][17] were presented; however, they all require extra cost to manage decryption keys, such as consuming more storage space and executing more computation on either the bank side [10][16] or the arbiter side [16][17].

Nowadays, customer anonymity has become a critical property of a fair exchange protocol because online stores keep track of customer's activities to analyze consumer behaviors to provide customized services and promotions. These activities violate customer's privacy and can trace a customer's purchase history. Lin and Liu's [10], and Ray *et al.*'s protocols [11] and Kupcu-Lysyanskaya's protocols [18] use electronic cash with blind signature [19] to achieve customer anonymity. This mechanism requires an electronic cash system for a customer to withdraw electronic cash before purchasing any product and for merchant to validate electronic cash. The Ray-Ray protocol [8] provides both merchant anonymity and customer anonymity with ephemeral key pairs to avoid eavesdropping during message transmission. However, the merchant can still be traced by his corresponding fixed and encrypted account value. In addition, the ephemeral key pairs are generated by the merchant and the customer themselves, so non-repudiation of the transaction cannot be achieved. Similarly, in Oniz *et al.*'s protocol [9], a customer can be traced by merchants or eavesdroppers based on the customer's public key. In Zhang *et al.*'s protocol [12], public keys of customer's/merchant's banks are applied to encrypt customer's/merchant's account for privacy protection. However, the customer/merchant can be traced by the forementioned manner.

2.1 Review of Oniz *et al.*'s Protocol

The notations depicted in **Table 1** are used in Oniz *et al.*'s protocol [9]. **Fig. 1** shows the process diagram of Oniz *et al*'s protocol. In the message 1, the protocol starts with the merchant sending encrypted e-good $\mathrm{E}_{\mathrm{KS}_i}$(e-good) and certificate of the product $\mathrm{CERT}_{\mathrm{TP}^i}$. The customer checks the certificate for the price, the description, and the hash value of the e-good. If all of them are valid, the customer sends the token (payment) and his/her public key KU_C to the merchant in the message 2. Then, the merchant checks the token. If the token is valid, the merchant sends the product decryption key KS_i encrypted with the public key of customer KU_C in the message 3. Finally, the customer can decrypt the encrypted product decryption key $\mathrm{E}_{\mathrm{KU}_\mathrm{C}}(\mathrm{KS}_i)$ with his/her pirvate key KR_C to obtain the product decryption key KS_i.

If a dispute occurs, the customer will send the token, the certificate index i, the product identifier PID, and his/her public key KU_C to the trusted third party TP in the message 4. Then, TP checks the token based on the PID. If the token is valid, TP will send the product decryption key KS_i encrypted with the public key of customer KU_C in the message 5. The customer can decrypt the encrypted product decryption key $\mathrm{E}_{\mathrm{KU}_\mathrm{C}}(\mathrm{KS}_i)$ with his/her pirvate key KR_C to obtain the product decryption key KS_i. Finally, TP will forward the token, the certificate index i, the product identifier PID, and his/her public key KU_C to the merchant in the message 6. Upon receiving message 6, the merchant will process the token.

Table 1. Notations of Oniz *et al.*'s protocol

Symbol	Description
$\mathrm{CERT}_{\mathrm{TP}^i}$	ith Certificate of a product signed by trusted third party
H(X)	Hash of X
E_K(data)	Encryption of data with key K
e-good	E-product or electronic item such as, database or multimedia file
Price	Price of e-good
Description	String describing contents of product
KU_X	Public key of identity X
KR_X	Private key of identity X
TP	Trusted third party
M	Merchant
C	Customer
$\|\|$	Concatenation Operation
PID	Product identifier

$\mathrm{CERT}_{\mathrm{TP}^i} = \mathrm{H}(\mathrm{E}_{\mathrm{KS}_i}(\text{e-good}))||\mathrm{Price}||\mathrm{Description}||\mathrm{PID}||i||\mathrm{Signature}$

1) M-->C: $\mathrm{E}_{\mathrm{KS}_i}(\text{e-good})\,||\,\mathrm{CERT}_{\mathrm{TP}^i}$

2) C-->M: $\mathrm{token}||\mathrm{KU}_\mathrm{C}$

3) M-->C: $\mathrm{E}_{\mathrm{KU}_\mathrm{C}}(\mathrm{KS}_i)$

<u>Dispute Resolution Phase</u>

Customer claims that he/she has not received Message 3.

4) C-->TP: $\mathrm{token}||i||\mathrm{PID}||\mathrm{KU}_\mathrm{C}$

5) TP-->C: $\mathrm{E}_{\mathrm{KU}_\mathrm{C}}(\mathrm{KS}_i)$

6) TP-->M: $\mathrm{token}||i||\mathrm{PID}||\mathrm{KU}_\mathrm{C}$

Fig. 1. Oniz *et al.*'s protocol

3. Cryptographic Backgrounds

In 1991, Girault [20] first introduced the notion of self-certified public keys which can be authenticated by other users without a separate certificate. That is, the verification for the valid public keys can be executed by corresponding applications later. In our protocol, Tsaur's ECC-based self-certified public key cryptosystem [13] is applied, including key generation and key exchange, to enhance protocol efficiency and guarantee security and fair exchange.

Key generation: In the Tsaur's system, each user needs to first generate a long-term private key and corresponding public key with the system authority. Let p be a large prime ($|p|$=160 bits), and $a,b \in \mathrm{F}_p$ be two elements specifying $4a^3+27b^2 (\bmod p) \neq 0$ and an elliptic curve $E(a,b)$ over the finite field F_p, defined by the equation $E: y^2 \equiv x^3+ax+b (\bmod p)$, the base point G of order q on E, and q is a large prime ($|q|$=160 bits). System Authority (*SA* for short) securely stores its private key x_{SA} ($x_{SA} \in [2,q-2]$) and publishes system parameters $\{E,G,p,q,Y_{SA},h(x)\}$, where $Y_{SA}=x_{SA}G$ and $h(x)$ is a one-way hash function which accepts a variable length input value and produces a fixed length output value ($|h(x)|$=160 bits). Then a user U has to perform the following steps to register with the *SA*:

1. U chooses an identity id and an random number $r \in [2,q-2]$ to compute a point $V=h(r\,||\,id)G$.
2. U sends $\{id,V\}$ to the *SA*.
3. *SA* chooses a random number $l \in [2,q-2]$ to compute U's public key Y and a witness w for U such that $Y=V+(l-h(id))G$ and $w=l+x_{SA}(X(Y)+h(id))(\bmod q)$, where $X(Y)$ denotes the x-coordinate of the point Y.
4. *SA* sends $\{Y,w\}$ to U.
5. U computes a private key $x=w+h(r\,||\,id)(\bmod q)$ and verifies the public key Y with the equation: $xG=Y+h(id)G+[(X(Y)+h(id))(\bmod q)]Y_{SA}$.

Session key exchange: We also use Tsaur's session key exchange [13] to generate a session key for a secure communication. Suppose that Alice and Bob require generating a session key for a communication session. The detail of key exchange is described below:

1. Alice chooses a random number $r_a \in [2, q-2]$ to compute a point $R_a = r_a G (\bmod p)$. Then, Alice sends $\{id_a, Y_a, R_a\}$ to Bob, where id_a is Alice's identity and Y_a is Alice's public key.
2. Upon receiving $\{id_a, Y_a, R_a\}$, Bob chooses a random number $r_b \in [2, q-2]$ to compute a point $R_b = r_b G (\bmod p)$. Then, Bob sends $\{id_b, Y_b, R_b\}$ to Alice, where id_b is Bob's identity and Y_b is Bob's public key. Finally, Bob computes a point $V_a = Y_a + h(id_a)G + [(X(Y_a) + h(id_a)) \bmod q]Y_{SA}$ and the session key $sk_{ba} = r_b V_a + x_b R_a$.
3. Upon receiving $\{id_b, Y_b, R_b\}$, Alice computes a point $V_b = Y_b + h(id_b)G + [(X(Y_b) + h(id_b)) \bmod q]Y_{SA}$ and the session key $sk_{ab} = r_a V_b + x_a R_b$.

Security of the Tsaur's session key exchange is shown in [13]. However, the Tsaur's session key exchange requires that each user exchanges his identity in advance so it cannot achieve customer unlinkability. Hence, we adopt the ECC-based Diffie-Hellman session key exchange [21] in our protocol to establish a secure communication between merchant and customer. As each participant uses an ephemeral key pair to generate a session key in each session, customer unlinkability can be achieved.

ECC-based Diffie-Hellman session key exchange: For a user *i*, let $x_i \in [2, q-2]$ be his private key, Y_i be his public key, $r_i \in [2, q-2]$ be a generated random integer, and a point $R_i = r_i G$. Let Y_i and R_i be public, and x_i and r_i be the secrets owned by a user *i*. Suppose that Alice generates a session key $sk_{AB} = x_A R_B + r_A Y_B$ and Bob generates a session key $sk_{BA} = x_B R_A + r_B Y_A$. If sk_{AB} and sk_{BA} are equivalent, it proves that Alice and Bob share the same session key. Correctness and security proof of ECC-based Diffie-Hellman session key exchange are discussed in [21].

In general, the public (asymmetric) key system is more computationally intensive than the symmetric key system. In practice, it is common to establish a symmetric session key through a public key cryptosystem with a key exchange scheme. This mechanism reduces data encryption/decryption time in a communication session, especially while massive amount of data is transmitted through network. In our protocol, Tsaur's session key exchange is adopted for communication between bank and merchant/customer. Also, ECC-based Diffie-Hellman session key exchange is adopted for communication between merchant and customer to achieve customer unlinkability.

4. The Proposed Protocol

The notations depicted in **Table 2** are used in our protocol. In our e-payment protocol, four distinct roles are involved and we define symbols C as online customer, M as the merchant who sells digital products, Z as the arbiter which generates and distributes product tokens to merchants and is responsible to resolve a transaction dispute, and B as the bank which provides e-payment transaction service between customers and merchants. We assume that every involving member, no matter what role it is, has gotten its long-term private key x_i and corresponding public key Y_i, which are generated in collaboration with the system authority SA. In addition, Z's identity id_Z, Z's public key Y_Z and SA's public key Y_{SA} are public information for every role member.

Table 2. Notations of the proposed protocol

Symbol	Description
SA	System authority
C	Customer
M	Merchant
B	Bank
Z	Arbiter
id_i	Identity of Entity i
ac_i	Bank account number of Entity i
x_i	Private key of Entity i, where $x_i \in [2, q-2]$, the base point G of order q on an elliptic curve E, and q is a large prime ($\|q\|$=160 bits)
x_i^1	Ephemeral private key of Entity i, where $x_i^1 \in [2, q-2]$
Y_i	Public key of Entity i
Y_i^1	Ephemeral public key of Entity i
t_i	Timestamp generated by Entity i
co_i	Content of digital product i
id_i^P	Identity of digital product i
pr_i	Price of digital product i
ds_i	Description of digital product i
k_i	Product decryption key of digital product i
sk_{ij}	Session key shared between Entity i and Entity j
s_i	Signature generated by Entity i using ECC-based self-certified public key system
s_i^1	Signature generated by Entity i using elliptic curve cryptosystem
r_i	Secret integer randomly chosen by Entity i, where $r_i \in [2, q-2]$
R_i	Point in ECC-based cryptosystem, where $R_i = r_i G$
$h(.)$	One-way hash function which accepts a variable length input value and produces a fixed length output value ($\|h(x)\|$=160 bits)
$H(sk_{ij}, msg)$	Keyed hash function, where inputs are session key and message, and output is Message Authenticated Code ($\|H(sk_{ij}, msg)\|$=160 bits)
$X(Y)$	Function to compute x-coordinate of the point Y in ECC-based cryptosystem
$Se(sk_{ij}, msg)$	Symmetric encryption function, where inputs are session key sk_{ij} and plaintext string msg, and output is a ciphertext string
$Sd(sk_{ij}, msg)$	Symmetric decryption function, where inputs are session key sk_{ij} and ciphertext string msg, and output is a plaintext string
$Ae(Y_i, msg)$	Asymmetric encryption function, where inputs are public key Y_i and plaintext string msg; output is a ciphertext string
$Ad(x_i, msg)$	Asymmetric decryption function, where inputs are private key x_i and ciphertext string msg, and output is a plaintext string
$Sg(x_i, msg)$	Signature function, where inputs are private key x_i and message msg, and output is a signature s_i
$Vr(Y_i, msg, s_i)$	Verification function, where inputs are public key Y_i, message msg, and

	signature s_i, and output is 1 (=Accept) or 0 (=Reject)
eac_C	Ephemeral bank account information for digital product *i*, such that $eac_C = Ae(Y_B, (ac_C \parallel id_C) \oplus X(Y_C^1))$
pay	Payment message, such that $pay = \{eac_C \parallel id_B \parallel id_M \parallel id_i^P \parallel pr_i \parallel t_C \parallel Y_C^1\}$
Tk_i	Product token of digital product *i*, such that $Tk_i = \{id_M \parallel id_i^P \parallel pr_i \parallel ds_i \parallel h(Se(k_i, co_i)) \parallel Ae(Y_Z, k_i) \parallel s_Z\}$

In a normal online transaction scenario, a customer *C* first needs to register his bank account and public key through the payment service registration of the bank *B*. Therefore, *B* knows *C*'s real identity and corresponding public key in advance. Then, *C* can surf online digital product shops and purchase digital products from any merchant *M* by sending purchase request. Upon receiving purchase request, *M* will transfer the encrypted digital product to *C*. Accordingly, *C* validates the downloaded product and then sends payment message to *M*. *M* first verifies the received payment message and then signs it before submitting the signed information along with payment request to *B*. *B* validates received payment message and transfers money payment from *C*'s account to *M*'s account. After the payment transaction is complete, *B* sends payment commitment messages to both *C* and *M*, respectively. Upon receiving payment commitment message, *M* sends product decryption key to *C*. The overview of product purchase and transaction phase is depicted in Fig. 2. If a dispute occurs, *C* will send a dispute request with payment commitment message to *Z* to obtain the product decryption key from *Z* directly without *M*'s consent. The overview of dispute resolution phase is depicted in Fig. 3.

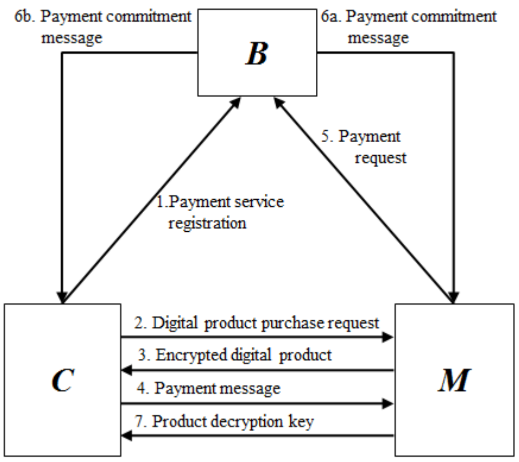

6b. Payment commitment message
B
6a. Payment commitment message
5. Payment request
1.Payment service registration
C
2. Digital product purchase request
3. Encrypted digital product
4. Payment message
7. Product decryption key
M

Fig. 2. Overview of product purchase and transaction phase

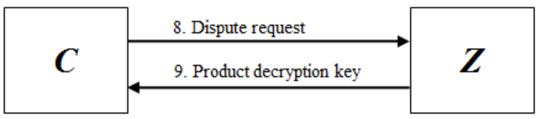


Fig. 3. Overview of dispute resolution phase

Our protocol is composed of three phases: product token generation and distribution phase, product purchase and transaction phase, and dispute resolution phase. In product token generation and distribution phase, *Z* generates product tokens and then distributes them to corresponding merchants before products made available online. Product purchase and transaction phase describes a normal purchase process among a customer, a merchant and the bank. When a dispute occurs, dispute resolution phase is activated to resolve the issue and preserve fair-exchange. The detailed description for each phase is depicted as follows.

4.1 Product Token Generation and Distribution Phase

Before placing digital products available online, a merchant *M* has to register his/her digital products to an arbiter *Z* over a secure channel. We describe the detailed steps in the following.

1. *M* sends the registration message $\{id_M \| co_i \| id_i^P \| ds_i \| pr_i\}$ for digital product *i* to *Z* to apply a corresponding product token Tk_i.
2. *Z* verifies the registration message of digital product *i* from *M*.
3. Suppose that the number of digital products registered from *M* is *j*. *Z* computes the current timestamp t_Z and recursively generates *j* product decryption keys $k_i = H((t_Z x_Z), k_{i-1})$, where $(t_Z x_Z)$ is computed as a secret key, $i \in \{1,2,...,j\}$, and k_0 is a randomly generated bit string by *Z*.
4. For each product, *Z* applies the product-related information $\{id_M \| id_i^P \| pr_i \| ds_i \| h(Se(k_i, co_i)) \| Ae(Y_Z, k_i)\}$ to compute a signature $s_Z = Sg(x_Z, id_M \| id_i^P \| pr_i \| ds_i \| h(Se(k_i, co_i)) \| Ae(Y_Z, k_i))$ with his private key x_Z, where $Se(k_i, co_i)$ is the encrypted digital product, $Ae(Y_Z, k_i)$ is the encrypted product decryption key for dispute resolution usage, and $h(Se(k_i, co_i))$ is used for digital product validation.
5. *Z* concatenates both product-related information and generated signature s_Z to derive a product token Tk_i for the corresponding digital product *i*, where $Tk_i = \{id_M \| id_i^P \| pr_i \| ds_i \| h(Se(k_i, co_i)) \| Ae(Y_Z, k_i) \| s_Z\}$. Note that each digital product has its own product token. The product token (Tk_i) is unique for each copy of a digital product because it is composed of an identifier of digital product (id_i^P) and a distinct product decryption key (k_i).
6. *Z* sends $\{k_i, Tk_i\}$ to *M* over a secure channel.
7. Upon receiving $\{k_i, Tk_i\}$, *M* stores them to the database. Finally, *M* publishes encrypted digital products and corresponding product tokens on his online shop.

4.2 Product Purchase and Transaction Phase

Every customer *C* must register himself to use e-payment service of the bank *B* before shopping online as *B* is the only role in our protocol who knows *C*'s identity. When *C* goes shopping online, he will download encrypted digital products and corresponding product tokens. Then, *C* will validate the correctness of downloaded product and pay *M* first to obtain theses product decryption keys from *M*. This phase is constructed by three sub-phases: payment service registration, digital product validation, and payment transaction.

Payment service registration sub-phase: A customer *C* should register his identity information $\{id_C, ac_C, Y_C\}$ for using e-payment service from the bank *B* through a secure offline channel. For instance, a customer registers his identity information for using e-payment service at the bank's counter in person. If the registration process is successfully complete, *B* gives bank identity information $\{id_B, Y_B\}$ to *C*.

Digital product validation sub-phase: Once customer *C* downloads an encrypted digital product and the corresponding product token successfully, *C* will verify the integrity of product token and the correctness of downloaded product. We depict the details in the following.

1. *C* downloads the encrypted product $\{Se(k_i, co_i)\}$ and the corresponding product token Tk_i from *M*'s online shop.
2. *C* derives both s_Z and msg_i from Tk_i, where $msg_i = \{id_M \| id_i^P \| pr_i \| ds_i \| h(Se(k_i, co_i)) \| Ae(Y_Z, k_i)\}$. Then, C verifies s_Z by executing verification function $Vr(Y_Z, msg_i, s_Z)$. If the value of $Vr(Y_Z, msg_i, s_Z)$ is equal to 1, Tk_i is valid.
3. *C* derives $h(Se(k_i, co_i))$ from Tk_i and computes the hash value of downloaded product $\{Se(k_i, co_i)\}$ by using hash function $h(.)$. If the above two hash values are equivalent, the integrity of downloaded product is preserved.
4. If the product token is valid and the integrity of downloaded product is preserved, *C* will initiate the payment transaction sub-phase.

Payment transaction sub-phase: After the customer *C* and the corresponding merchant *M* authenticate each other, *C* sends the payment message to *M*. The payment message includes *C*'s signature, *C*'s identity, and *C*'s bank account number.

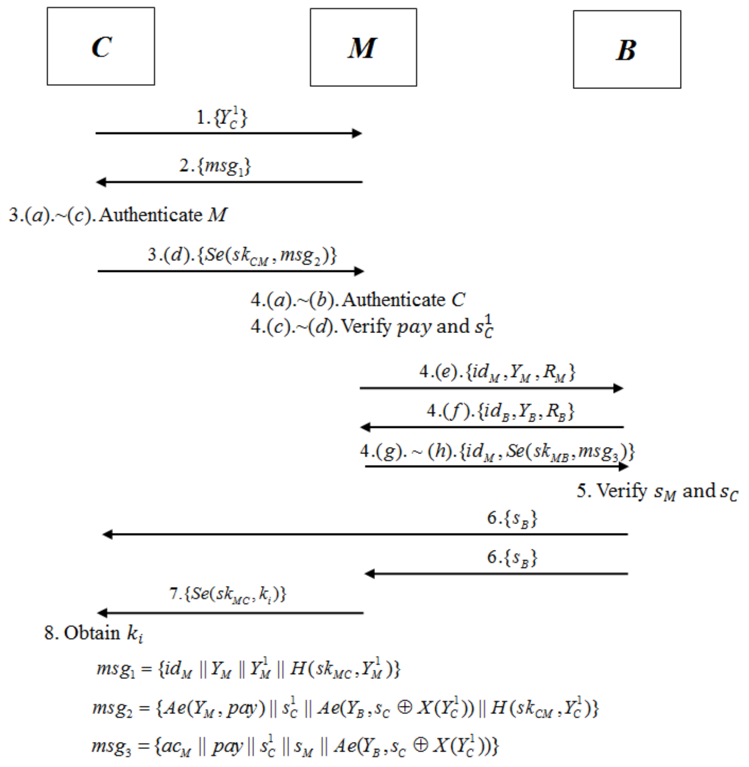

C
M
B
1.$\{Y_C^1\}$
2.$\{msg_1\}$
3.(a).~(c). Authenticate M
3.(d).$\{Se(sk_{CM}, msg_2)\}$
4.(a).~(b). Authenticate C
4.(c).~(d). Verify pay and s_C^1
4.(e).$\{id_M, Y_M, R_M\}$
4.(f).$\{id_B, Y_B, R_B\}$
4.(g). ~ (h).$\{id_M, Se(sk_{MB}, msg_3)\}$
5. Verify s_M and s_C
6.$\{s_B\}$
6.$\{s_B\}$
7.$\{Se(sk_{MC}, k_i)\}$
8. Obtain k_i
$msg_1 = \{id_M \| Y_M \| Y_M^1 \| H(sk_{MC}, Y_M^1)\}$
$msg_2 = \{Ae(Y_M, pay) \| s_C^1 \| Ae(Y_B, s_C \oplus X(Y_C^1)) \| H(sk_{CM}, Y_C^1)\}$
$msg_3 = \{ac_M \| pay \| s_C^1 \| s_M \| Ae(Y_B, s_C \oplus X(Y_C^1))\}$

Fig. 4. Details of payment transaction sub-phase

For the purpose of customer anonymity and customer unlinkability, the above three values are encrypted with bank *B*'s public key. In other words, only bank *B* can decrypt them and then knows *C*'s identity. Upon receiving the payment message, *M* checks the product identity and its price. Then, *M* signs the payment message with his private key as a confirmation to *C*'s payment message and sends the payment message associated with his signature to *B*. After *B* verifies *C*'s and *M*'s signatures successfully, *B* will transfer money from *C*'s account to *M*'s account. Then, *B* sends the payment commitment message to both *C* and *M*. Upon receiving the payment commitment message, *M* sends the product decryption key to *C*. **Fig. 4** shows the process diagram of payment transaction sub-phase. The detailed steps in payment transaction sub-phase are stated as follows.

1. *C* generates an ephemeral key pair (x_C^1, Y_C^1), where x_C^1 is a secret integer selected randomly and $Y_C^1 = x_C^1 G$. Then, *C* sends the message $\{Y_C^1\}$ to *M*.
2. *M* generates an ephemeral key pair (x_M^1, Y_M^1), where x_M^1 is a secret integer selected randomly and $Y_M^1 = x_M^1 G$. Then, *M* computes a session key $sk_{MC} = x_M^1 Y_C^1$ and sends the message $\{id_M \| Y_M \| Y_M^1 \| H(sk_{MC}, Y_M^1)\}$ to *C*.
3. After *C* authenticates *M*, *C* computes the payment message and sends it to *M*. We depict the details in the following.
   (a) Upon receiving the message $\{id_M \| Y_M \| Y_M^1 \| H(sk_{MC}, Y_M^1)\}$, *C* computes the session key $sk_{CM} = x_C^1 Y_M^1$.
   (b) *C* uses sk_{CM} and Y_M^1 to compute the value of $H(sk_{CM}, Y_M^1)$. If the derived value of $H(sk_{CM}, Y_M^1)$ is equal to the received value of $H(sk_{MC}, Y_M^1)$, C authenticates *M*.
   (c) After *M* is authenticated, *C* computes the current timestamp t_C, ephemeral bank account information eac_C, payment message pay, signature s_C^1 and signature s_C, such that $eac_C = Ae(Y_B, (ac_C \| id_C) \oplus X(Y_C^1))$, $pay = \{eac_C \| id_B \| id_M \| id_i^P \| pr_i \| t_C \| Y_C^1\}$, $s_C^1 = Sg(x_C^1, pay)$, and $s_C = Sg(x_C, pay)$.
   (d) *C* encrypts the payment message pay with *M*'s public key Y_M by $Ae(Y_M, pay)$ and sends the message $\{Se(sk_{CM}, Ae(Y_M, pay) \| s_C^1 \| Ae(Y_B, s_C \oplus X(Y_C^1)) \| H(sk_{CM}, Y_C^1))\}$ to *M*. Note that the payment can only be revealed by genuine *M*.
4. *M* decrypts the received message from *C* to obtain payment message and verifies *C*'s signature. If *C*'s signature is valid, *M* signs *C*'s payment message with his private key and establishes a session key with *B*. Finally, *M* sends payment message along with his signature to *B*. We depict the details in the following.
   (a) *M* decrypts the message $\{Se(sk_{CM}, Ae(Y_M, pay) \| s_C^1 \| Ae(Y_B, s_C \oplus X(Y_C^1)) \| H(sk_{CM}, Y_C^1))\}$ with session key sk_{MC} to obtain $\{Ae(Y_M, pay) \| s_C^1 \| Ae(Y_B, s_C \oplus X(Y_C^1)) \| H(sk_{CM}, Y_C^1)\}$.
   (b) *M* uses sk_{MC} and Y_C^1 to compute the value of $H(sk_{MC}, Y_C^1)$. If the derived value of $H(sk_{MC}, Y_C^1)$ is equal to the received value of $H(sk_{CM}, Y_C^1)$, *M* authenticates *C*.
   (c) If *C* is authenticated, *M* decrypts the encrypted payment message with his private key x_M to obtain the payment message pay, derives the product identity id_i^P and the product price pr_i from pay, and checks whether the price pr_i is correct. If pr_i is correct, *M* verifies signature s_C^1 by executing verification function $Vr(Y_C^1, pay, s_C^1)$. If the value of $Vr(Y_C^1, pay, s_C^1)$ is equal to 1, the integrity of received payment message

is preserved.

(d) If the product price pr_i and signature s_C^1 are verified, *M* generates a signature $s_M = Sg(x_M, pay)$ with his private key x_M as a confirmation to *C*'s payment message.

(e) *M* randomly chooses a secret integer r_M and computes $R_M = r_M G$. Then, *M* sends the message $\{id_M, Y_M, R_M\}$ to *B*.

(f) Upon receiving $\{id_M, Y_M, R_M\}$, *B* randomly chooses a secret integer r_B and computes $R_B = r_B G$. Then, *B* sends the message $\{id_B, Y_B, R_B\}$ to *M*. Finally, *B* computes $V_M = Y_M + h(id_M)G + [(X(Y_M) + h(id_M)) \bmod q]Y_{SA}$ and the session key $sk_{BM} = r_B V_M + x_B R_M$.

(g) Upon receiving the message $\{id_B, Y_B, R_B\}$, *M* computes $V_B = Y_B + h(id_B)G + [(X(Y_B) + h(id_B)) \bmod q]Y_{SA}$ and the session key $sk_{MB} = r_M V_B + x_M R_B$.

(h) *M* sends the message $\{id_M, Se(sk_{MB}, ac_M \| pay \| s_C^1 \| s_M \| Ae(Y_B, s_C \oplus X(Y_C^1)))\}$ to *B*.

5. Bank *B* decrypts the received message from *M* to verify *M*'s signature and *C*'s signature. If the two signatures are valid, *B* will check whether *C*'s deposit is enough for this payment. If the deposit is enough, *B* will transfer money from *C*'s account to *M*'s account. We depict the details in the following.

(a) Upon receiving the message $\{id_M, Se(sk_{MB}, ac_M \| pay \| s_C^1 \| s_M \| Ae(Y_B, s_C \oplus X(Y_C^1)))\}$, B obtains *M*'s identity id_M. Based on id_M, *B* decrypts $\{Se(sk_{MB}, ac_M \| pay \| s_C^1 \| s_M \| Ae(Y_B, s_C \oplus X(Y_C^1)))\}$ with session key sk_{BM} to obtain $\{ac_M \| pay \| s_C^1 \| s_M \| Ae(Y_B, s_C \oplus X(Y_C^1))\}$.

(b) If the value of id_M is equal to *M*'s identity derived from payment message pay, and timestamp t_C derived from pay is within the pre-defined valid period, then *B* verifies *M*'s signature s_M by executing verification function $Vr(Y_M, pay, s_M)$. If the value of $Vr(Y_M, pay, s_M)$ is equal to 1, *B* confirms that *M*'s signature s_M is valid.

(c) From the payment message pay, *B* uses his private key x_B and Y_C^1 to decrypt ephemeral bank account information eac_C and the message $\{Ae(Y_B, s_C \oplus X(Y_C^1))\}$. Then, *B* can obtain *C*'s bank account number ac_C, *C*'s identity id_C, and *C*'s signature s_C.

(d) By using the tuple (ac_C, id_c) as search keys in the database, *B* can search and find *C*'s public key Y_C. Then, *B* will verify *C*'s signature s_C by executing verification function $Vr(Y_C, pay, s_C)$. If the value of $Vr(Y_C, pay, s_C)$ is equal to 1, *B* confirms that *C*'s signature s_C is valid.

(e) After s_M and s_C are verified successfully, *B* checks whether *C*'s deposit is enough for the payment transaction. If it is enough, *B* transfers money from *C*'s account ac_C to *M*'s account ac_M.

6. After money transaction is successfully executed, *B* uses his private key x_B to generate payment commitment message $s_B = Sg(x_B, pay)$ and sends it to both *C* and *M*, respectively.

7. Upon receiving the payment commitment message s_B, both *C* and *M* use *B*'s public key Y_B to verify s_B by executing verification function $Vr(Y_B, pay, s_B)$, respectively. If the value of $Vr(Y_B, pay, s_B)$ is equal to 1, the validation of s_B is confirmed. Then, *C* stores s_B for dispute resolution usage, and *M* encrypts the product decryption key k_i with the session key sk_{MC} and sends the message $\{Se(sk_{MC}, k_i)\}$ to *C*.
8. *C* decrypts the message $\{Se(sk_{MC}, k_i)\}$ with the session key sk_{CM} to obtain the product decryption key k_i. Then, *C* decrypts encrypted digital product with k_i to obtain the content of digital product *i*. If the content is valid, the product purchase and transaction phase is complete. Otherwise, *C* will launch the dispute resolution phase to get the correct product decryption key from *Z*.

4.3 Dispute Resolution Phase

If a customer *C* cannot obtain a valid decryption key from *M* after the money transaction, C will invoke a dispute resolution request to the arbiter *Z*. The detailed steps of the dispute resolution phase shown in **Fig. 5** are stated as follows.

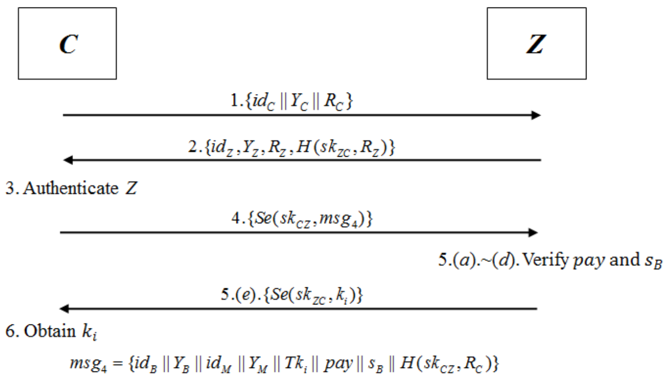

C
Z
1.{$id_C \| Y_C \| R_C$}
2.{$id_Z, Y_Z, R_Z, H(sk_{ZC}, R_Z)$}
3. Authenticate Z
4.{$Se(sk_{CZ}, msg_4)$}
5.(a).~(d). Verify pay and s_B
5.(e).{$Se(sk_{ZC}, k_i)$}
6. Obtain k_i
$msg_4 = \{id_B \| Y_B \| id_M \| Y_M \| Tk_i \| pay \| s_B \| H(sk_{CZ}, R_C)\}$

Fig. 5. Details of dispute resolution phase

1. *C* randomly chooses a secret integer r_C and computes $R_C = r_C G$. Then, *C* sends a dispute resolution request with the message $\{id_C, Y_C, R_C\}$ to *Z*.
2. Upon receiving $\{id_C, Y_C, R_C\}$, *Z* randomly chooses a secret integer r_Z and computes $R_Z = r_Z G$, $V_C = Y_C + h(id_C)G + [(X(Y_C) + h(id_C)) \bmod q]Y_{SA}$, and session key $sk_{ZC} = r_Z V_C + x_Z R_C$. Then, *Z* sends the message $\{id_Z, Y_Z, R_Z, H(sk_{ZC}, R_Z)\}$ to *C*.
3. Upon receiving $\{id_Z, Y_Z, R_Z, H(sk_{ZC}, R_Z)\}$, *C* computes $V_Z = Y_Z + h(id_Z)G + [(X(Y_Z) + h(id_Z)) \bmod q]Y_{SA}$, and session key $sk_{CZ} = r_C V_Z + x_C R_Z$. Then, *C* uses sk_{CZ} and R_Z to compute the value of $H(sk_{CZ}, R_Z)$. If $H(sk_{CZ}, R_Z)$ and $H(sk_{ZC}, R_Z)$ are equivalent, *C* authenticates *Z*.
4. After *C* authenticates *Z*, *C* uses the session key sk_{CZ} to encrypt the message $\{id_B \| Y_B \| id_M \| Y_M \| Tk_i \| pay \| s_B \| H(sk_{CZ}, R_C)\}$. Then, *C* sends the message $\{Se(sk_{CZ}, id_B \| Y_B \| id_M \| Y_M \| Tk_i \| pay \| s_B \| H(sk_{CZ}, R_C))\}$ to *Z*.
5. *Z* decrypts the message from *C* and verifies payment commitment message, payment

message, and product token. Then, Z decrypts product decryption key derived from product token and sends it to C. We depict the details in the following.

(a) Z decrypts $\{Se(sk_{CZ}, id_B \| Y_B \| id_M \| Y_M \| Tk_i \| pay \| s_B \| H(sk_{CZ}, R_C))\}$ with session key sk_{ZC} to obtain $\{id_B \| Y_B \| id_M \| Y_M \| Tk_i \| pay \| s_B \| H(sk_{CZ}, R_C)\}$.

(b) Z uses sk_{ZC} and R_C to compute the value of $H(sk_{ZC}, R_C)$. If the derived value of $H(sk_{ZC}, R_C)$ is equal to the received value of $H(sk_{CZ}, R_C)$, Z authenticates C.

(c) After C is authenticated, Z compares the value of $\{id_i^P\}$ derived from pay with the value of $\{id_i^P\}$ derived from Tk_i. If they are equivalent, Z verifies payment commitment message s_B by executing verification function $Vr(Y_B, pay, s_B)$. If the value of $Vr(Y_B, pay, s_B)$ is equal to 1, then Z ensures that C had completed payment transaction.

(d) Z decrypts the message $\{Ae(Y_Z, k_i)\}$ with its private key x_Z to obtain product decryption key k_i.

(e) Z encrypts k_i with session key sk_{ZC} and sends $\{Se(sk_{ZC}, k_i)\}$ to C.

6. C decrypts the message $\{Se(sk_{ZC}, k_i)\}$ with session key sk_{CZ} to obtain the product decryption key k_i. Finally, C can decrypt the encrypted product $\{Se(k_i, co_i)\}$ with k_i to obtain its content co_i.

5. Discussion

In this section, we first define security notions for a fair-exchange e-payment protocol for digital products about security requirements for message confidentiality and unforgeability. The security of our proposed concrete protocol is based on the difficulty of solving Elliptic Curve Discrete Logarithm Problem (ECDLP for short) [22] and One-way Hash Function (OHF for short) [23]. Then, we discuss whether our protocol supports true fair exchange, customer anonymity and unlinkability. The robust security of our protocol is based on the following assumptions.

- All participants, i.e., the roles of bank, merchant and customer, are honest.
- Arbiter Z is secure and available when dispute resolution mechanism is invoked.
- Payment commitment messages generated from bank B will be successfully received by corresponding customer C and merchant M.

Finally, performance comparisons among our protocol and others are also discussed in this section.

5.1 Security Analysis

Message Confidentiality: We define a security model for indistinguishability of a fair-exchange e-payment protocol for digital products under a chosen ciphertext attack below.
Definition 1: A fair-exchange e-payment protocol for digital products is semantically secure against a chosen ciphertext attack if there is no attacker who possesses a non-negligible probability to break the chosen ciphertext within polynomial time in the following game.
Initialization: A challenger α chooses system parameters and generates identities public keys for all entities. After the system initialization, an attacker β chooses a target user with a public key.
Queries: The attacker β issues the following queries adaptively

- Symmetric-encryption queries: The attacker *β* chooses a message *M* and two identities of Entities *i* and *j*, and submits a symmetric-encryption query for *M* to the challenger *α*. The attacker will be given the ciphertext by *α*.
- Symmetric-decryption queries: The attacker *β* chooses a ciphertext and two identities of Entities *i* and *j*, and submits a symmetric-decryption query to the challenger *α*. The attacker will be given the plaintext by *α*.
- Asymmetric-encryption queries: The attacker *β* can obtain the ciphertext with an entity's public key.
- Asymmetric-decryption queries: The attacker *β* chooses a ciphertext and an identity of Entity *i*, and submits an asymmetric-decryption query to the challenger *α*. The attacker will be given the plaintext by *α*.
- Signature queries: The attacker *β* chooses a message *M* and an identity of Entity *i* and submits a signature query for *M* to the challenger *α*. The attacker will be given the result by *α*.
- Private-key queries: The attacker *β* chooses an identities of Entity *i* and submits a private-key query to the challenger *α*. The attacker will be given the private key by *α*.
- Session-key queries: The attacker *β* chooses two identities of Entities *i* and *j*, and submits a symmetric-decryption query to the challenger *α*. The attacker will be given the plaintext by *α*.

Challenge: The attacker *β* produces two equal length messages, M_0 and M_1. The challenger *α* flips a coin *b*←{0, 1} and computes a ciphertext that will be sent to *β* as a challenge.

Guess: At the end of the game, *β* outputs a bit b_0. The attacker *β* wins if $b_0 = b$. The advantage for the attacker *β* is Adv(*β*)= $Pr[b_0 = b]$-1/2=0, where $Pr[b_0 = b]$ means the probability for $b_0 = b$. The ciphertext is encrypted by using symmetric encryption or asymmetric encryption. In the game, it is not allowed to issue a private-key/session-key query for the target user or an encryption/decryption query for the target message.

Theorem 1. *Our protocol is semantically secure against a chosen ciphertext attack if there is no polynomial-time algorithm that solves the ECDLP with non-negligible probability.*

Proof. Let β_1 be a polynomial-time algorithm that breaks the fair-exchange e-payment protocol for digital products in the chosen ciphertext attack. Let $(G, Y = xG)$ be a random instance of the ECDLP. We will show how to use β_1 to construct a polynomial-time algorithm that solves the ECDLP with non-negligible probability. In the simulation, β_1 adaptively issues queries as in the game of Definition 1.

For the target transmitted message among *C*, *M*, and *B*, β_1 can obtain each user's identities id_M, id_B, id_C, id_Z and other information Y_M, Y_B, Y_C, Y_Z, R_M, R_B, R_C, R_Z, and $V_M = Y_M + h(id_M)G + [(X(Y_M) + h(id_M)) \bmod q]Y_{SA}$, $V_B = Y_B + h(id_B)G + [(X(Y_B) + h(id_B)) \bmod q]Y_{SA}$, $V_C = Y_C + h(id_C)G + [(X(Y_C) + h(id_C)) \bmod q]Y_{SA}$, and $V_Z = Y_Z + h(id_Z)G + [(X(Y_Z) + h(id_Z)) \bmod q]Y_{SA}$ in the simulation. Note that β_1 cannot submit private-key/session-key queries for the target users in simulations. However, without valid private keys x_M^1, x_C^1, x_B, x_M, x_Z, or x_C, the attacker cannot obtain valid session keys $sk_{MC} = x_M^1 Y_C^1$, $sk_{CM} = x_C^1 Y_M^1$, $sk_{BM} = r_B V_M + x_B R_M$, $sk_{MB} = r_M V_B + x_M R_B$, $sk_{ZC} = r_Z V_C + x_Z R_C$, and $sk_{CZ} = r_C V_Z + x_C R_Z$. The simulated experiment is distributed identically as real experiment. Hence, the advantage for β_1 is Adv(*β*)= $Pr[b_0 = b]$-1/2=0. That is, the is no polynomial-time algorithm that solves the ECDLP with non-negligible probability. We conclude that our proposed protocol achieves message confidentiality.

Unforgeability: We define a model for unforgeability of a fair-exchange e-payment protocol for digital products as follows.

Definition 2: A fair-exchange e-payment protocol for digital products is semantically secure against a chosen-message attack if there is no attacker who possesses a non-negligible probability to break the chosen message within polynomial time in the following game.

Initialization: A challenger α chooses system parameters and generates identities public keys for all entities. After the system initialization, an attacker β chooses a target message M^*.

Queries: The attacker β submits the same queries described in Definition 1 to the challenger α.

Forgery: The attacker β produces a signature for the target message M^*. The attacker β wins if the signature for M^* is valid. In the game, it is not allowed to issue a signature query for M^* or a private-key query for the target user.

Theorem 2. Let β_2 be a polynomial-time algorithm that breaks the fair-exchange e-payment protocol for digital products in the chosen message attack. Let $(G, Y = xG)$ be a random instance of the ECDLP. We will show how to use β_2 to construct a polynomial-time algorithm that solves the ECDLP with non-negligible probability. In the simulation, β_2 adaptively issues queries as in the game of Definition 2.

In the simulation, the payment message pay^* is separately signed by *C* and *M* during the transaction process, and the signature verification function is applied to validate the target payment signatures $s_C{}^*$ and $s_M{}^*$. Note that it is not allowed to issue a signature query for the target payment message pay^* or a private-key query for the target customer C^* and M^*. In the simulation, β_2 cannot obtain C^*'s private key $x_C{}^*$ and M^*'s private key $x_M{}^*$. The probability of successfully forging the payment message is ε, where ε is negligible. Hence, there is no polynomial-time algorithm that solves the ECDLP with non-negligible probability. We conclude that our proposed protocol achieves payment unforgeability. That is, no attacker can forge valid signatures from both customer and the corresponding merchant.

5.2 Achievement of True Fair Exchange

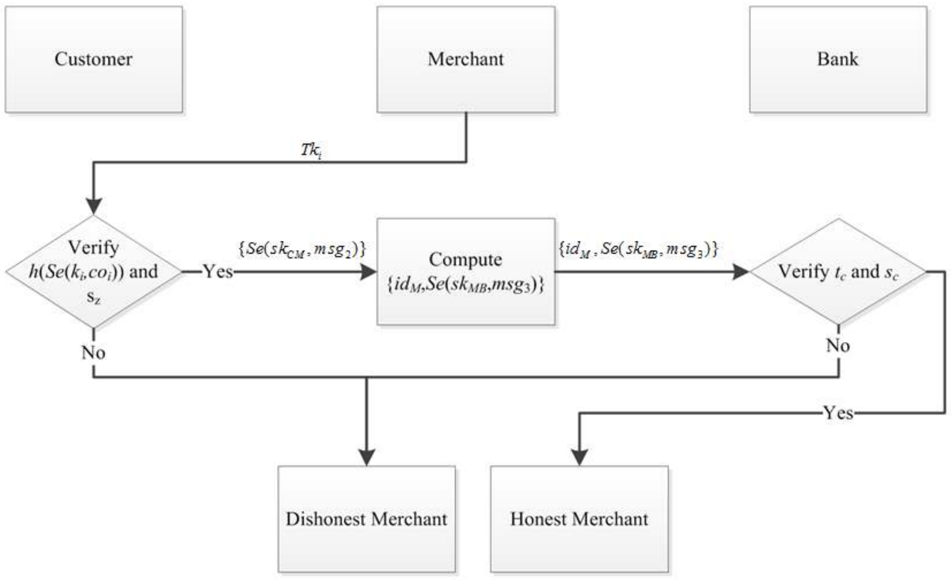
In this section, we analyze achievement of true fair exchange in our protocol. In the scenario of both customer *C* and merchant *M* being honest, after *C* validates the received digital product and pays for it, *M* should send *C* the product decryption key during product purchase and transaction phase. If a dispute occurred, *C* can obtain the product decryption key from the arbiter *Z*. Furthermore, true fair exchange can be still guaranteed in the following dispute scenarios in which either the merchant or the customer is dishonest. The flow charts of dishonest merchant and dishonest customer are shown as **Fig. 6** and **Fig. 7**, respectively.

Dishonest Merchant: A dishonest merchant *M*' may try to gain advantage with the following malicious actions.

- *M*' may send an incorrect product token or an incorrect product to *C*. This is not applicable because *C* will verify product token $Tk_i = \{id_M \| id_i^P \| pr_i \| ds_i \| h(Se(k_i, co_i)) \| Ae(Y_Z, k_i) \| s_Z\}$ and the integrity of encrypted digital product before invoking the payment transaction. Without Z's private key x_Z, *M*' cannot forge a valid Z's signature s_Z.
- *M*' may reuse an authorized payment message to transfer extra money from *C*'s account to his own account. However, payment message $pay = \{eac_C \| id_B \| id_M \| id_i^P \| pr_i \| t_C \| Y_C^1\}$

contains a timestamp t_C. Hence, *M*' cannot reuse a payment message to bank *B* for the money transaction. In addition, *M*' cannot modify the timestamp t_C in the payment message because *B* will verify *C*'s payment signature $s_C = Sg(x_C, pay)$. Without *C*'s private key x_C, *M*' cannot forge a valid *C*'s payment signature s_C.

- *M*' may forge a higher product price in a payment message to get extra money from *C*. This is not applicable because *C* has generated a payment signature $s_C = Sg(x_C, pay)$ and then bank *B* will validate *C*'s payment signature s_C by checking the value of $Vr(Y_C, pay, s_C)$ is equal to 1. Hence, if *M*' modifies a product price in the payment message pay, *M*' cannot forge a valid *C*'s payment signature s_C.
- *M*' may send incorrect product decryption key to *C*. This malicious action cannot succeed because *C* will launch a dispute resolution request to obtain the correct product decryption key k_i if the digital product *i* cannot be decrypted correctly.



$$Tk_i = \{id_M \| id_i^P \| pr_i \| ds_i \| h(Se(k_i, co_i)) \| Ae(Y_Z, k_i) \| s_Z\}$$
$$msg_2 = \{Ae(Y_M, pay) \| s_C^1 \| Ae(Y_B, s_C \oplus X(Y_C^1)) \| H(sk_{CM}, Y_C^1)\}$$
$$pay = \{eac_C \| id_B \| id_M \| id_i^P \| pr_i \| t_C \| Y_C^1\}$$
$$msg_3 = \{ac_M \| pay \| s_C^1 \| s_M \| Ae(Y_B, s_C \oplus X(Y_C^1))\}$$

Fig. 6. Flow chart of dishonest merchant

Dishonest Customer: A dishonest customer *C'* may try to gain advantage with the following malicious actions.

- *C'* may forge a payment message, such as decreasing the original product price to pr_i', to cheat *M*. This is not applicable because *M* checks pr_i' by matching product identity id_i^P with the corresponding pr_i in the database. Once the product price pr_i' is not equal to pr_i, *M* will terminate the transaction session.
- *C'* may reuse an authorized payment message to obtain different products from *M*. This is

not applicable because M utilizes bank B's public key Y_B to verify payment commitment message s_B by checking the value of $Vr(Y_B, pay, s_B)$ is equal to 1 and then sends product decryption key k_i to C' according to the product identity id_i^P derived from pay. Therefore, C' cannot reuse pay to obtain different digital products.

- C' may receive a valid product decryption key k_i but claim that the product decryption key is invalid. This is not applicable because in our proposed protocol if any dispute occurs, C' should launch a dispute resolution request to obtain the correct product decryption key from the arbiter Z and cannot request bank B to cancel payment transaction.
- C' may forge a payment commitment message $s_B = Sg(x_B, pay)$ and then launches a dispute resolution request to obtain the product decryption key k_i. This is not applicable because C' cannot forge a valid bank B's signature s_B.
-

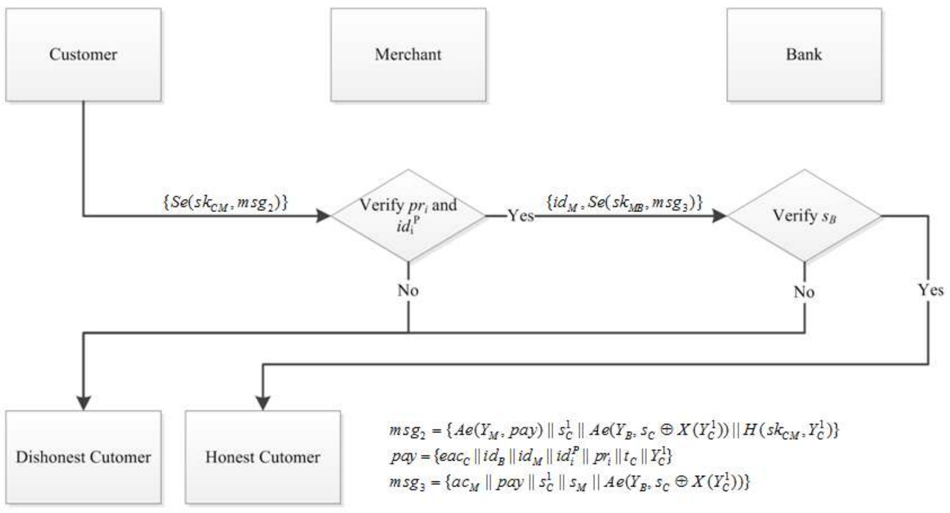

Customer
Merchant
Bank
$\{Se(sk_{CM}, msg_2)\}$
Verify pr_i and id_i^P
Yes
$\{id_M, Se(sk_{MB}, msg_3)\}$
Verify s_B
No
No
Yes
Dishonest Cutomer
Honest Cutomer
$msg_2 = \{Ae(Y_M, pay) \| s_C^1 \| Ae(Y_B, s_C \oplus X(Y_C^1)) \| H(sk_{CM}, Y_C^1)\}$
$pay = \{eac_C \| id_B \| id_M \| id_i^P \| pr_i \| t_C \| Y_C^1\}$
$msg_3 = \{ac_M \| pay \| s_C^1 \| s_M \| Ae(Y_B, s_C \oplus X(Y_C^1))\}$

Fig. 7. Flow chart of dishonest customer

5.3 Achievement of Customer anonymity and unlinkability

In our protocol, a customer C provides his identity information $\{id_C, ac_C, Y_C\}$ to the bank B to register a payment service over a secure channel during payment service registration sub-phase. Therefore, an adversary cannot get C's identity id_C. During payment transaction sub-phase, the customer C uses ephemeral bank account information eac_C and ephemeral key pair (x_C^1, Y_C^1) to achieve customer anonymity and customer unlinkability. In $eac_C = Ae(Y_B, (ac_C \| id_C) \oplus X(Y_C^1))$, customer's identity id_C is encrypted with bank's public key Y_B. Only the bank B can decrypt eac_C to get id_C. An adversary or a merchant cannot know customer's identity unless he has bank's private key. Hence, customer anonymity is achieved. Furthermore, eac_C is ephemeral in each payment transaction because id_C is Exclusive-ORed with new generated public key Y_C^1. The adversary/merchant cannot store and use eac_C or Y_C^1 to trace the customer. Hence, customer unlinkability is achieved.

Notice that the customer cannot ensure the received ephemeral public key is generated from the genuine merchant. In such case, one may wonder if the received ephemeral public key is generated by an adversary, does customer unlinkability still be achieved? Assume the received ephemeral public key is generated from an adversary M', the customer sends the message $\{S_e(sk_{CM'}, Ae(Y_{M'}, pay)) \| S_C^1 \| Ae(Y_B, s_C \oplus X(Y_C^1)) \| H(sk_{CM'}, Y_C^1)\}$ to M' in the third step of payment transaction sub-phase. M' will decrypt this message to obtain the payment message pay. Without the bank's private key x_B, M' cannot know customer's signature s_C to trace the customer C. Furthermore, according to payment message $pay = \{eac_C \| id_B \| id_{M'} \| id_i^P \| pr_i \| t_C \| Y_C^1\}$ and ephemeral bank account information $eac_C = Ae(Y_B, (ac_C \| id_C) \oplus X(Y_C^1))$, even if M' can obtain pay, without the bank's private key x_B, M' cannot know customer's bank account ac_C and identity id_C to trace the customer C. In addition, an adversary is unable to observe the value of $Ae(Y_B, s_C \oplus X(Y_C^1))$ to identify C because Y_C^1 is freshly generated in each payment transaction. Hence, customer unlinkability is still accomplished.

5.4 Protocol Comparisons on Important Features and Performance

In this section, we compare four recently published protocols, which were proposed by Ray *et al.*'s [11], Oniz *et al.*'s [9], Lin-Liu's [10], and Alaraj-Munro's [17], on eight important features for e-payment protocol and protocol performance in terms of computational complexity. **Table 3** shows protocols comparison in terms of eight important features, which include true fairness, automated dispute resolution, product validation, customer anonymity, customer unlinkability, implementation requirement for Secure Electronic Transaction (SET), implementation requirement e-cash system, storage requirement for decryption keys on the arbiter. The feature of true fairness allows the arbiter to provide a product decryption key to a customer after customer's payment is validated. The feature of automated dispute resolution assures the enforcement of fair exchange without manual intervention if a dispute occurs. The feature of product validation ensures the integrity of digital product. The feature of customer anonymity is to conceal customer's identity. The feature of customer unlinkability provides privacy protection to avoid a customer being tracked by a fixed individual-identifiable value used in online transactions. The features of implementation requirements for SET and e-cash system provide system support for secure electronic payment transaction. The feature of storage requirement for decryption keys on the arbiter indicates the demand of storage space for the arbiter to store the huge amount of decryption keys. Notice that extra cost is required for SET and e-cash system to function since arbiter, bank, customer, and merchant all need to install system components in general to guarantee payment service operations. In addition, the large amount of storage space requirement for arbiter may raise storage scalability concern and performance bottleneck for key search operation at the server environment.

As shown in **Table 3**, Ray *et al.*'s and Lin-Liu's protocols adopt e-cash to achieve customer anonymity. Therefore, an e-cash system is required to support e-cash withdrawal and validation functions. In addition, Oniz *et al.*'s protocol requires SET implementation to preserve customer anonymity. Instead of relying on extra SET or e-cash system to achieve fair exchange and customer anonymity, our protocol adopts self-certified public key mechanism and invents product token to achieve the same features, i.e., no extra e-payment system is required. Moreover, only our protocol can achieve customer unlinkability by preventing a merchant from tracing his customer's identity intentionally. Regarding to the storage requirement for decryption keys on the arbiter, Oniz *et al.*'s protocol requires less space for key storage than other previously proposed protocols by utilizing key chain mechanism to reduce the total number of stored decryption keys. In contrast, our protocol does not need to store any decryption keys on the arbiter.

Table 3. Comparison of important features among fair-exchange e-payment protocols

Protocol / Property	Ray *et al.*[11]	Oniz *et al.*[9]	Lin-Liu[10]	Alaraj-Munro [17]	Ours
a	Yes	Yes	Yes	Yes	Yes
b	Yes	Yes	Yes	Yes	Yes
c	Yes	Yes	Yes	Yes	Yes
d	Yes	Yes	Yes	No	Yes
e	No	No	No	No	**Yes**
f	No	Yes	No	No	No
g	Yes	No	Yes	No	No
h	Yes	Yes*	Yes	Yes	**No**

a: True fairness
b: Automated dispute resolution
c: Product validation
d: Customer Anonymity
e: Customer Unlinkability
f: Implementation requirement for Secure Electronic Transaction (SET)
g: Implementation requirement for e-cash system
h: Storage requirement for decryption keys on the arbiter
Yes *: Oniz *et al.*'s protocol requires few keys for dispute resolution on the arbiter by using key chain mechanism.

To evaluate computational complexity for our protocol, we define notations for execution time as follows. T_{mm} is the execution time of one modular multiplication. T_h is the execution time of one hash function operation. T_{mi} is the execution time of one modular inverse operation. T_{me} is the execution time of one modular exponentiation operation. T_{pa} is the execution time of one point addition operation. T_{pm} is the execution time of one point multiplication operation. T_{sym} is the execution time of one symmetric encryption/decryption operation. T_{sig} is the execution time of one signature generation operation. T_{ver} is the execution time of one signature verification operation. T_{enc} is the execution time of one asymmetric encryption operation. T_{dec} is the execution time of one asymmetric decryption operation. According to [13][24], T_h, T_{mi}, T_{me}, T_{pa}, and T_{pm} can be presented in T_{mm} unit where $T_h \approx 4T_{mm}$, $T_{mi} \approx 3T_{mm}$, $T_{me} \approx 240T_{mm}$, $T_{pa} \approx 0.12T_{mm}$ and $T_{pm} \approx 29T_{mm}$. In our protocol, mutual authentication and session key exchange are achieved at the same time during protocol execution. A secure channel between two parties can be established through Diffie-Hellman key exchange [25], which requires at least four modular exponentiations. To achieve mutual authentication property, at least four hash function operations are required in these compared protocols. These existing protocols, except our protocol, need to perform these computations in advance to get a secure channel and mutual authenticated between two parties. To simplify our comparison on some RSA-based schemes, we assume that one modular exponentiation operation and one hash function operation are generally required to perform each of the following tasks: computing a signature, verifying a signature, encrypting a data string and decrypting a message in RSA mechanism [26]. Therefore, T_{sig}, T_{ver}, T_{enc}, and T_{dec} can be assessed to $244T_{mm}$ unit for RSA-based. In our protocol, T_{sig}, T_{ver}, T_{enc}, and T_{dec} can be also assessed in terms of T_{mm} unit such that $T_{sig} \approx 34T_{mm}$, $T_{ver} \approx 124T_{mm}$, $T_{enc} \approx 120T_{mm}$, and $T_{dec} \approx 29T_{mm}$ [13]. According to [27][28], the execution time of RSA encryption/decryption operation, measured with software implementation, is at least 100 times slower than the execution time of Data Encryption Standard [29] or the execution time of Advanced Encryption Standard [30]. Therefore, here we ignore the execution time of symmetric encryption/decryption operations qT_{sym} in all protocols when we calculate the total assessment time, where q is an integer and $1 \le q \le 24$. The detailed comparisons on computational complexity among fair-exchange e-payment protocols in product purchase and transaction phase and in dispute resolution phase are depicted in **Table 4** and **Table 5**, respectively, where i stands for the chain key index [9].

Table 4 shows that our protocol reduces 56% to 87% of the total execution times in comparison with others in product purchase and transaction phase. In **Table 5**, our protocol is about one order less than the other protocols in terms of computational complexity on total execution time in dispute resolution phase. Hence, our protocol is more efficient than previous ones in terms of computational complexity.

In addition, our arbiter only requires protecting his own private key and does not store any decryption key for encrypted digital products. This greatly reduces the management cost on the arbiter. Our proposed protocol only requires a customer to download the encrypted product once in normal situation and download only the decryption key from the arbiter during dispute resolution phase. Hence, from a customer's point of view, our protocol is also efficient in terms of session flow.

Table 4. Comparison on computational complexity among fair-exchange e-payment protocols in product purchase and transaction phase

Protocols	Computational complexity	Total assessment time
Ray *et al.*[11]	$5,276T_{mm} + 2T_{sym} + 31T_{enc}$	$13,290T_{mm}$
Oniz *et al.*[9]	$(8i + 2,972)T_{mm} + T_{sym} + 4T_{sig} + 5T_{ver} + T_{enc} + T_{dec}$	$(8i + 5,656)T_{mm}$
Lin-Liu[10]	$4,729T_{mm} + 24T_{sym} + 22T_{enc}$	$10,097T_{mm}$
Alaraj-Munro [17]	$1,008T_{mm} + 3T_{sym} + 2T_{sig} + 9T_{ver} + 3T_{enc} + 2T_{dec}$	$4,912T_{mm}$
Ours	$431T_{mm} + 7T_{sym} + 4T_{sig} + 5T_{ver} + 2T_{enc} + 2T_{dec}$	$1,485T_{mm}$

Table 5. Comparison on computational complexity among fair-exchange e-payment protocols in dispute resolution phase

Protocols	Computational complexity	Total assessment time
Ray *et al.*[11]	$3,216T_{mm} + 7T_{enc}$	$4,924T_{mm}$
Oniz *et al.*[9]	$(8i + 3,960)T_{mm} + T_{sym} + 2T_{sig} + 3T_{ver} + T_{enc} + T_{dec}$	$(8i + 5,668)T_{mm}$
Lin-Liu[10]	$2,480T_{mm} + 5T_{sym} + 10T_{enc}$	$4,920T_{mm}$
Alaraj-Munro [17]	$1,984T_{mm} + 2T_{sym} + 6T_{ver} + 2T_{enc} + 2T_{dec}$	$4,424T_{mm}$
Ours	$315T_{mm} + 5T_{sym} + T_{ver} + T_{dec}$	$468T_{mm}$

6. Conclusion

In this paper, we have presented a fair-exchange e-payment protocol using ECC-based self-certified public key. Our protocol provides true fairness, customer anonymity, and customer unlinkability. To reduce communication and computational overheads, no offline arbiter is required in product purchase and transaction phase of our protocol. If a dispute occurs, our automated dispute resolution mechanism preserves the transaction fairness. Furthermore, in our protocol the arbiter does not require any decryption key management. For product validation, a customer only needs to download encrypted digital products once from a merchant. In addition, privacy protection is achieved by customer unlinkability of our protocol as a customer's identity cannot be traced by any merchant and malicious eavesdropper.

References

[1] Asokan N., Schunter M. and Waidner M. "Optimistic protocols for fair exchange". In *Proc. 4th ACM Conf. Computer and Communications Security*, Zurich, Switzerland, pp. 7-17, 1997. Article (CrossRef Link)

[2] Ray I., Ray I. and Narasimhamurthi N. "A fair-exchange e-commerce protocol with automated dispute resolution." In *Proc. the IFIP TC11/ WG11.3 Fourteenth Annual Working Conf. Database Security: Data and Application Security, Development and Directions*, Amsterdam, The Netherlands, pp.27-38, 2002. Article (CrossRef Link)

[3] Ray I., Ray I. "An optimistic fair exchange e-commerce protocol with automated dispute resolution." In *Proc. First International Conf. Electronic Commerce and Web Technologies*, Greenwich, UK, 84-93, 2000;. Article (CrossRef Link)

[4] Nenadic A. and Zhang N., Cheetham B., Goble C. "RSA-based certified delivery of e-goods using verifiable and recoverable signature encryption." *Journal of Universal Computer Science*, vol. 11, no. 1, pp.175-192, 2005. Article (CrossRef Link)

[5] Alaraj A. and Munro M. "An e-commerce fair exchange protocol for exchanging digital products and payments." In *Proc. 2nd International Conf. Digital Information Management*, Lyon, France, pp. 248-253, 2007. Article (CrossRef Link)

[6] Fan W., Shu H., Fife E. and Yan Q. "An enhanced-security fair e-payment protocol." *2009 World Congress on Computer Science and Information* Engineering, Los Angeles, USA, pp. 516-519, 2009. Article (CrossRef Link)

[7] Franklin M.K. and Reiter M.K. "Fair exchange with a semi-trusted third party." In *Proc. 4th ACM Conference Computer and Communications Security*, Zurich, Switzerland, pp.1-6, 1997. Article (CrossRef Link)

[8] Ray I., Ray I." An anonymous fair-exchange e-commerce protocol." In *Proc. 15th International in Parallel and Distributed Processing Symposium*, San Francisco, CA, pp. 1790-1797, 2001. Article (CrossRef Link)

[9] Oniz C.C., Savas E. and Levi A. "An optimistic fair e-commerce protocol for large e-goods." In *Proc. of 7th IEEE International Symposium on Computer Networks*, Istanbul, Turkey, pp. 214-219, 2006. Article (CrossRef Link)

[10] Lin S.J. and Liu D.C. "A fair-exchange and customer-anonymity electronic commerce protocol for digital content transactions." In *Proc. 4th international* conf. *Distributed Computing and Internet Technology*, Bangalore, India, pp. 321-326, 2007; 321-326.

[11] Ray I., Ray I., Narasimhamurthi N. "An anonymous and failure resilient fair-exchange e-commerce protocol." *Decision Support Systems*, pp. 267-202, 2005. Article (CrossRef Link)

[12] Zhang Q., Markantonakis K. and Mayes K. "A mutual authentication enabled fair-exchange and anonymous e-payment protocol." In *Proc. 8th IEEE* International *Conf. E-Commerce Technology and 3rd IEEE international Conf. Enterprise Computing, E-Commerce, and E-Services*, San Francisco, CA, pp. 20-27, 2006. Article (CrossRef Link)

[13] Tsaur W.J. "Several security schemes constructed using ECC-based self-certified public key

cryptosystems. *Applied Mathematics and Computation*," vol.168, pp.447-464, 2005. Article (CrossRef Link)

[14] Wang J.H., Liu J.W., Li X.H., Kou W.D. Fair e-payment protocol based on blind signature. *The Journal of China Universities of Posts and Telecommunications*, vol.16, no.5, pp.114-118,2009;. Article (CrossRef Link)

[15] The Internet Engineering Task Force (IETF) - The PKIX Working Group *RFC 2459 Internet X.509 Public Key Infrastructure-Certificate and CRL Profile*.

[16] Alaraj A. and Munro M. "An efficient fair exchange protocol that enforces the merchant to be honest." In *Proc. 2007 International Conference Collaborative Computing: Networking, Applications and Worksharing table of contents*, New York, pp. 196-202, 2007; 196-202. Article (CrossRef Link)

[17] Alaraj A. and Munro M. "An efficient e-commerce fair exchange protocol that encourages customer and merchant to be honest." In *Proc. 27th international conference on Computer Safety, Reliability, and Security*, Newcastle upon Tyne, UK, pp. 193-206, 2008. Article (CrossRef Link)

[18] Kupcu A. and Lysyanskaya A. "Usable optimistic fair exchange." *Computer Networks*, vol.56, no. 1, pp. 50-63, 2012. Article (CrossRef Link)

[19] Chaum D., Fiat A and Naor M. "Untraceable electronic cash." *Advances in Cryptology – In Proc. CRYPTO '88*, Santa Barbara, CA, pp.200-212, 1990.

[20] Girault M. Self-certified public keys. *Advances in* Cryptology - *EURO-CRYPT'91*, Brighton, UK,pp. 491-497, 1991.

[21] Washington L.C. *Elliptic Curves- Number Theory and* Cryptography. 2nd ed., Taylor and Francis Group, LLC, Boca Raton, pp.170-171, 2008. Article (CrossRef Link)

[22] IEEE 1363 Working Group *IEEE P1363 standard* specifications *for public key cryptography*.

[23] National Institute of Standards and Technology NIST FIPS PUB 180 *Secure hash standard*. U. S. Department of Commerce, 1993.

[24] Wu T.S., Hsu C.L., Lin H.Y. Self-certified multi-proxy signature schemes with message recovery. *Journal of Zhejiang University - Science A*, vol. 10, no.2, pp.290-300, 2009. Article (CrossRef Link)

[25] Diffie W., Hellman M.E. New directions in cryptography. *IEEE Trans. Information Theory*, IT-22, pp. 644-654, 1976. Article (CrossRef Link)

[26] Rivest R.L., Shamir A. and Adleman L. "A method for obtaining digital signatures and public-key cryptosystems." *Communications of the ACM*, vol. 21, no. 2, pp. 120-126, 1978. Article (CrossRef Link)

[27] Schneier B. *Applied Cryptography*. John Wiley & Sons, Inc., Canada, pp.469, 1996.

[28] Lo N.W., Yang C.H. and Yeh K.H. "Performance evaluation of two secure communication schemes on vehicular ad hoc networks." In *Proc. 2008 International Computer Symposium*, Taipei, 2008.

[29] National Bureau of Standards, NBS FIPS PUB 46 *Data encryption standard*. National Bureau of Standards, U.S. Department of Commerce, 1977.

[30] National Institute of Standards and Technology, FIPS PUB 197 *Advanced encryption standard*. U.S. Department of Commerce, 2001.

[31] Chen C.L. and Liao J.J. "a fair online payment system for digital content via subliminal channel," *Electronic Commerce Research and Applications*, vol. 10, no. 3, pp.279-287, 2011. Article (CrossRef Link)

[32] Wang G. "An abuse-free fair contract-signing protocol based on the RSA signature", *IEEE Transactions on Information Forensics and Security*,vol. 5, no. 1, pp. 158-168, 2010. Article (CrossRef Link)

Yi-Chung Yen received the M.Sc. degree in Information Security from Royal Holloway, University of London, in 2002. He is a Ph.D. Student at the Department of Information Management, National Taiwan University of Science and Technology, Taiwan. His major field of study is information security.

Tzong-ChenWu is a Professor at the Department of Information Management, National Taiwan University of Science and Technology, Taiwan. He completed his PhD in the Department of Computer Science and Information Engineering from National Chiao Tung University, Taiwan, in 1992. Now, he serves as Honor Consultant of the Chinese Cryptology and Information Security Association (CCISA for short). He is also the members of IEEE and ACM. His current research interests include cryptography, data security, network security, and data engineering.

Nai-Wei Lo is an Associate Professor at the Department of Information Management, National Taiwan University of Science and Technology, Taiwan. He completed his PhD in the Department of Computer Science and Information Engineering from State University of New York at Stony Brook. His current research interests include Network Security, RFID Security, and cloud computing Security.

Kuo-Yu Tsai is an Assistant Professor at the Department of Management Information Systems, Hwa Hsia Institute of Technology, Taiwan. He received his Ph.D. Degree in the Department of Information Management from National Taiwan University of Science and Technology, Taiwan, in 2009. His recent research interests include information security, cryptography, network security, and cloud computing.