The Journal of Systems and Software 116 (2016) 69–74

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

# A mobile payment mechanism with anonymity for cloud computing

Jen-Ho Yang [a], Pei-Yu Lin [b,*]

[a] Department of Multimedia and M-Commerce, Kainan University, Taiwan
[b] Department of Information Communication, and Innovation Center for Big Data and Digital Convergence, Yuan Ze University, 135 Yuan-Tung Rd., Chung-Li 32003, Taiwan

ARTICLE INFO

*Article history:*
Received 29 September 2014
Revised 30 April 2015
Accepted 9 July 2015
Available online 17 July 2015

*Keywords:*
Electronic transaction
Mobile payment
Cloud computing

ABSTRACT

In recent years, traditional transactions have been replaced by electronic transactions. To protect the security of the electronic transactions, various electronic payment (e-payment) mechanisms have been proposed. However, we find the previous e-payment mechanisms do not provide the non-repudiation requirement in the client side. Thus, a malicious client can easily deny the transaction and the merchant may not get the payment. In addition, these mechanisms have large computation and communication costs so they cannot be applied to the mobile payment for cloud computing. To solve the above problems, we propose a new mobile payment mechanism with anonymity for cloud computing in this paper. The proposed mechanism not only reduces the computation cost but also provides the non-repudiation requirement in the client side. Compared with the related works, the proposed mechanism is securer, fairer, and more efficient. Therefore, the proposed mobile payment mechanism is more suitable and practical for the cloud computing.



## 1. Introduction

With the rapid development in network technologies, traditional transactions have been replaced by electronic transactions (e-transaction) in recent years. In e-transaction applications, many people use mobile devices to deal the transactions, which is so-called electronic commerce (e-commerce). Because the e-commerce can be dealt anytime and anywhere, more and more people use it for selling, buying, and marketing. To protect the security of e-commerce, various electronic payment (e-payment) mechanisms have been proposed (Abad-Peiro et al., 1998; Chari et al., 2001; Kungpisdan et al., 2003; Wei et al., 2005; Martinez-Pelaez et al., 2010; Sun et al., 2010; Tsai et al., 2011). Generally speaking, a secure e-payment mechanism has to provide confidentiality, anonymity, integrity, fairness, and non-reputation. To accomplish the above requirements, how to design a secure and fair e-payment mechanism becomes an important issue for mobile commerce.

In 2012, Isaac and Zeadally (2012) proposed an anonymous secure payment mechanism in a payment gateway centric model. In their e-payment mechanism, clients and the merchant communicate each other through a payment gate. The clients do not communicate with the merchant directly so the client anonymity can be accomplished. In addition, they use the symmetric key cryptosystem (Menezes et al., 1996) to provide the confidentiality. Thus, the e-payment information can be well-protected. Besides, their mechanism also provides the integrity by using the message authentication code (MAC). Thus, the clients and the merchant can ensure that the information is not tampered during transferring via networks.

According to the above description, Isaac and Zeadally use the payment gateway to accomplish the confidentiality, anonymity, and integrity. We find that their mechanism satisfies the requirements of cloud computing because the transaction messages can be stored and protected in the payment gateway. That is, the payment gateway can be implemented as a cloud server for the e-payment in mobile commerce (Madlmayr et al., 2008; Pailles et al., 2010; Alpár et al., 2012; Pourghoumi and Ghinea, 2012a,b; Kounelis et al., 2012; Pourghomi et al., 2013, 2014; Moss, 2012; HCE, 2014; GSMA, 2014).

However, we also find that Isaac and Zeadally's mechanism does not provides the fairness and non-reputation requirements for the e-transaction. In their mechanism, the client generates the payment information anonymously to protect the payment privacy. This causes that the client can deny the transaction because the payment information cannot link directly to the client. Moreover, their mechanism uses the redundant symmetric key between the client and the merchant. This redundant key is unnecessary because all messages must be transmitted through the payment gateway. This causes the key management problem of the client. If the client wants to buy items from difference merchants, the client needs to keep many keys for different merchants. This also increases the computation and communication costs in this scenario. According to the above reasons, Isaac and Zeadally's e-payment mechanism has some problems if we want to apply it to mobile payment applications for cloud computing.

* Corresponding author. Tel.: +886 3 4638800x2130; fax: +886 3 4638277.
*E-mail addresses:* jenhoyang@mail.knu.edu.tw (J.-H. Yang), pylin@saturn.yzu.edu.tw, pagelin3@gmail.com (P.-Y. Lin).

http://dx.doi.org/10.1016/j.jss.2015.07.023
0164-1212/

To solve the above problems, we propose a new mobile payment mechanism with anonymity for cloud computing in this paper. The proposed mechanism has the following advantages. First, the proposed model uses a payment gateway between the client and the merchant, and thus the client does not need to communicate with the merchant directly. That is, the client can transact anonymously to protect the payment privacy. Second, the client's bank generates the digital signature as the payment proof. Thus, the merchant can get the payment from the bank even if the malicious client denies the transaction. Third, we eliminate the redundant symmetric keys between the client and the merchant. The client does not need to maintain many keys for different merchants so the key management problem can be solved. Finally, we reduce the computation and communication costs in the user side so the proposed mobile payment mechanism is very suitable for cloud computing environments. According to the above advantages, the proposed mechanism provides the security requirements of confidentiality, anonymity, integrity, fairness, and non-reputation.

The proposed mechanism is securer than the related works because the payment gateway is used to be the cloud server for saving the client's payment information. In addition, the proposed mechanism is fairer than the related works because the payment proof is generated by the client's bank. Thus, the client cannot deny the transaction. Besides, the proposed mechanism has less computation costs so it is more efficient for the mobile payment. According to the above descriptions, the proposed mechanism is securer, fairer, and more efficient than the related works. Therefore, the proposed mobile payment mechanism is more suitable and practical for cloud computing environments.

## 2. Review of the related work

In this section, we introduce Isaac and Zeadally's e-payment mechanism (Isaac and Zeadally, 2012). Their mechanism has five roles: the client, the merchant, the issuer (the client's bank), the acquirer (the merchant's bank), and the payment gateway. Note that all payment messages among the client, the merchant, the issuer, and the acquirer must be transmitted through the payment gateway. The notations used in their mechanism are shown in Table 1.

**Table 1**
The notations of Isaac and Zeadally's e-payment mechanism.

| Notations | Descriptions |
|---|---|
| $ID_p$ | The identity of the participant $p$ |
| $NID_C$ | The temporary identity of the client |
| $TID$ | The identity of a transaction includes transaction time and date |
| $TST_p$ | The timestamp generated by the participant $p$ |
| $Stt$ | The state of a transaction |
| $OD$ | The order description |
| $Price$ | The amount of the currency |
| $OI$ | The order information ($OI = \{TID, OD, h(OD, Price)\}$) |
| $TC$ | The type of card used in purchase process |
| $TIDReq$ | The request of $TID$ |
| $MIDReq$ | The request for the merchant's identity |
| $SEC_{A-B}$ | The secret shared between the participants $A$ and $B$ |
| $\{M\}_x$ | The symmetric encryption with the message $M$ using the symmetric key $x$ |
| $h(M)$ | The one-way hash function of $M$ |
| $MAC(M, K)$ | Message authentication code of $M$ with the key $K$ |
| $KS_{A-B_i}$ | The session key shared between $A$ and $B$, where $i$ is $i$-bit cyclic shifting of $KS_{A-B}$ for generating the next session key (Isaac and Zeadally, 2012) |
| $PRequest$ | The payment request |
| $PResponse$ | The payment response |
| $VSRequest$ | The value-subtraction request |
| $VSResponse$ | The value-subtraction response |
| $VCRequest$ | The value-claim request |
| $VCResponse$ | The value-claim response |

The steps of Isaac and Zeadally's e-payment mechanism are described as follows.

Step 1: The client and the merchant exchange the necessary messages to start the mechanism through the payment gateway by the following sub-steps.
- *Step 1-1:* The client sends $NID_C$, $i$, and $TIDReq$ to the payment gateway.
- *Step 1-2:* The payment gateway forwards the above messages to the merchant.
- *Step 1-3:* The merchant sends $ID_p$ and $TID$ encrypted by $KS_{C-M_i}$ to the payment gateway.
- *Step 1-4:* The payment gateway forwards the above message to the client.

*Step 2:* The client generates the payment requirement $PRequest$ by the following sub-steps.
- *Step 2-1:* The client generates the value-subtraction request $VSRequest = (MAC, [(Price, h(OI), TST_C, TC, ID_M), KS_{C-I_Z}], TC, TST_C)$.
- *Step 2-2:* The client generates $PRequest = \{NID_C, ID_I, Price, OI, z, VSReques\}_{KS_{C-M_i}}$, and $MAC[OI, Price, NID_C, ID_I, TST_C, z, h(KS_{C-I_Z}), KS_{C-M_{i+1}}]$.
- *Step 2-3:* The client sends the payment request to the payment gateway.

*Step 3:* The payment gateway forwards $PRequest$ to the merchant.

*Step 4:* The merchant generates value-claim request $VCRequest$ by the following sub-steps.
- *Step 4-1:* The merchant decrypts $PRequest$ to obtain $OI$, $TST_C$, and $VSRequest$.
- *Step 4-2:* The merchant verifies the validity of $TST_C$. If it is valid, then the merchant generates $VCRequest = (VSRequest, TST_M, h(OI), TID, Price, NID_C, ID_I)$, $\{VCRequest, ID_M, z, h(KS_{C-I_Z})\}_{KS_{M-PG_k}}$, $k$, and $MAC[(VCRequest, TST_M, z, h(KS_{C-I_Z})), KS_{M-PG_{k+1}}]$.

*Step 5:* The payment gateway verifies and approve the payment using the private network of the banks by the following sub-steps.
- *Step 5-1:* The payment gateway decrypts $VCRequest$ and verifies the validity of $TST_M$. If it is valid, then the gateway sends $NID_C$, $ID_M$, $VSRequest$, $TID$, $h(OI)$, $z$, $Price$, and $h(KS_{C-I_Z})$ to the issuer. Besides, the gateway sends $Price$ and $ID_M$ to the acquirer.
- *Step 5-2:* The issuer checks the validity of the client by the messages from the payment gateway. If the client is valid, then the issuer approves the transaction.
- *Step 5-3:* The acquirer checks $Price$ and $ID_M$ and asks the issuer transfers the money to the merchant's account.
- *Step 5-4:* The issuer generates $VSResponse = \{Stt, h(OI), h(KS_{M-PG_{k+1}})\}_{KS_{C-I_Z}}$ and sends $VSResponse, Stt, h(Stt, h(OI)), h(KS_{C-I_Z})$ to payment gateway.

*Step 6:* The payment gateway generates $VCResponse = \{Stt, h(Stt, h(OI), h(KS_{C-I_Z}))\}_{KS_{M-PG_{k+1}}}$ and sends it to the merchant.

*Step 7:* The payment gateway generates the payment response by the following sub-steps.
- *Step 7-1:* The payment gateway generates $PResponse = \{VSResponse\}_{KS_{C-PG_{j+1}}}$ and sends it to the client.
- *Step 7-2:* The client decrypts $PResponse$ to get $h(OI)$. Then, the client checks if the received $h(OI)$ is equal to his own $h(OI)$. If they are not equal, then the client sends the failure message to the payment gateway. Then, the payment gateway starts the recovery procedure or resends the message.

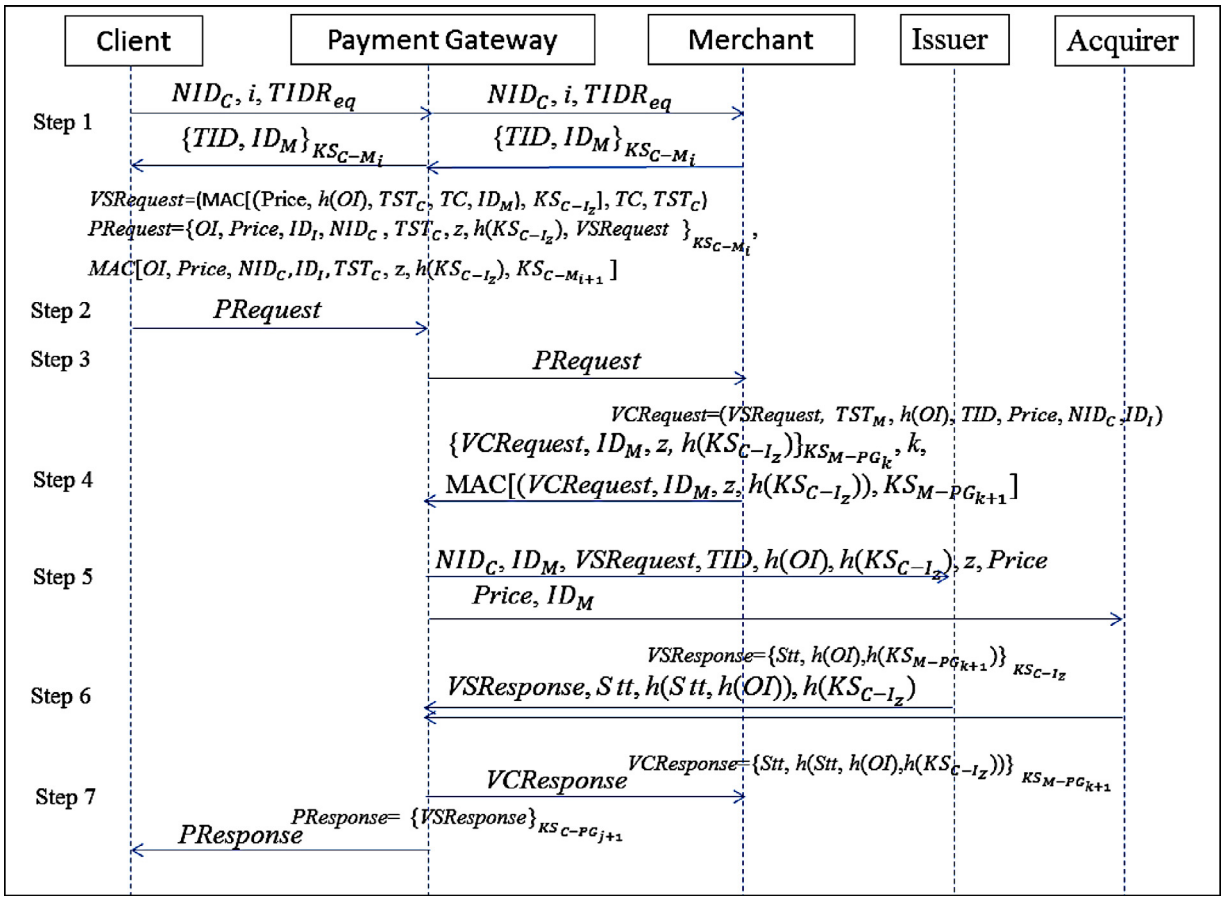


**Fig. 1.** The steps of Isaac and Zeadally's mechanism.

The steps of Isaac and Zeadally's mechanism are illustrated in Fig. 1. According to the above descriptions, we find that their mechanism has the following drawbacks. First, the client and the merchant own the same symmetric key $KS_{C-M_i}$, and thus the payment request *PRequest* can be also generated by the merchant. Thus, the client can claim that the *PRequest* is not generated by him/her and denies the transaction. Second, their mechanism uses four symmetric keys $KS_{C-M_i}$, $KS_{C-I_Z}$, $KS_{C-PG_j}$, and $KS_{M-PG_k}$. However, we find that $KS_{C-M_i}$ is unnecessary because all messages between the client and the merchant must be transmitted through the payment gateway. If the client wants to transact with several merchants, then the client has to maintain many symmetric keys for the different merchants. That is, using $KS_{C-M_i}$ causes the key management problem for the client. Besides, this also increases the computation and communication costs for the client. Thus, their mechanism is not suitable for the mobile payment in cloud computing applications.

## 3. The proposed mobile payment mechanism for cloud computing

To solve the drawbacks of Isaac and Zeadally's mechanism, we propose a new mobile payment mechanism with anonymity for cloud computing in this section. The proposed mechanism has five participants: the client, the merchant, the issuer, the acquirer, and the payment gateway in the cloud area. Note that the issuer is the client's bank which is responsible for issuing the credit card to the client. In addition, the acquirer is the merchant's bank which is responsible for receiving the money from the issuer. In addition, it is divided into two phases: the initialization phase and the transaction phase. The model of the proposed mechanism is illustrated in Fig. 2. The notations used in the proposed mechanism are shown in Table 2.

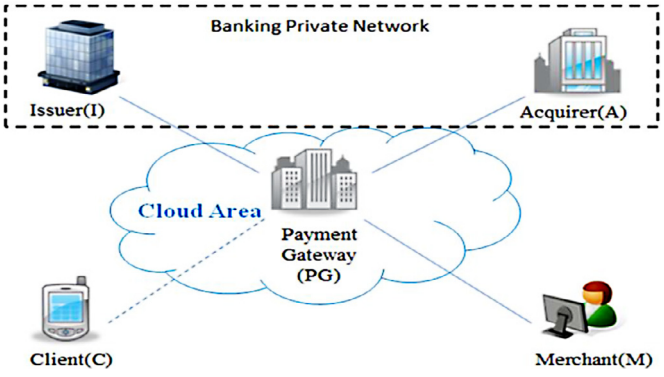


**Fig. 2.** The model of the proposed mobile payment mechanism.

### 3.1. The initialization phase

In this phase, the client, the merchant, the issuer, and the acquirer have to register with the payment gateway to obtain the session keys $KS_{C-PG_a}$, $KS_{M-PG_y}$, and $KS_{C-I_t}$. Then, the issuer generates the parameters $e$, $n$, and $d$ used in RSA cryptosystem (Rivest et al., 1978), where $(e, n)$ is the public key pair and $d$ is the private key. Note that the public key pair has been certified by a certificate authority.

**Table 2**
The notations used in the proposed mechanism.

| Notations | Descriptions |
|---|---|
| $NID_C$ | The temporary identity of the client |
| $A$ | The purchasing information includes the good information and the price |
| $ID_i$ | The identity of the participant $i$ |
| $TInfo$ | The transaction information includes transaction time, date, and the serial number |
| $Price$ | The amount of the payment |
| $m$ | The payment information computed by $m = \{NID_C, TInfo, Price\}$ |
| $KS_{A-B_j}$ | The session key shared between $A$ and $B$, where $j$ is $j$-bit cyclic shifting of $KS_{A-B}$ for generating the next session key (Isaac and Zeadally, 2012) |
| $SRequest$ | The signature request |
| $TS_i$ | The timestamp generated by the participant $i$ |
| $h(\cdot)$ | A secure one-way hash function |
| $Issuer\_ID$ | The identity of the issuer |
| $Acquirer\_ID$ | The identity of the acquirer |
| $Stt$ | The state of a transaction |
| $PResponse$ | The payment response |

### 3.2. The transaction phase

*Step 1:* The client generates the purchasing information $A$ and sends the purchasing request to the merchant to start the transaction by the following sub-steps.

*Step 1-1:* The client sends $NID_C$ and $A$ to the payment gateway. Then, the payment gateway forwards $NID_C$ and $A$ to the merchant.

*Step 1-2:* After receiving $NID_C$ and $A$, and generates $TInfo$ and sends it to the payment gateway. Then, the payment gateway forwards $TInfo$ to the client. Finally, the merchant records $NID_C$ and $A$ in its database.

*Step 2:* The client asks the issuer to sign the payment information as a payment poof by the following sub-steps.

*Step 2-1:* The client computes $SRequest = (h(TInfo), ID_C, NID_C, h(m), Price, TS_C)_{KS_{C-I_t}}$.

*Step 2-2:* The client sends $SRequest$ to the payment gateway. Then, the payment gateway forwards it to the issuer.

*Step 2-3:* The issuer decrypts $SRequest$ and checks if $TS_C$ is a valid timestamp or not. Then, the issuer uses its private key $d$ to compute $S = h(m)^d \mod n$ as a digital signature. Then, the issuer records $ID_C$, $NID_C$, $h(TInfo)$, and $S$ in its database. Then, it sends $(S)_{KS_{C-I_t}}$ to the client through the payment gateway.

*Step 2-4:* The client verifies the correctness of $S$ by checking if $h(m)$ is equal to $S^e \mod n$. If they are equal, then the client accepts this signature as a payment proof.

*Step 3:* The client sends $(S, m, h(TInfo), TS_C, Issuer\_ID)_{KS_{C-PG_a}}$ to the payment gateway. Then, the payment gateway decrypts the above message to get $S$, $m$, $h(TInfo)$, $TS_C$, and $Issuer\_ID$. Then, the payment gateway generates $(S, m, h(TInfo), TS_C, Issuer\_ID)_{KS_{M-PG_y}}$ and sends it to the merchant.

*Step 4:* The merchant decrypts $(S, m, h(TInfo), TS_C, Issuer\_ID)_{KS_{M-PG_y}}$ and checks the validity of $TS_C$ by comparing with the current time $T$. If $(T - TS_C)$ is in a valid time interval, then the merchant ensures that $(S, m, h(TInfo), TS_C, Issuer\_ID)_{KS_{M-PG_y}}$ is sent by the gateway not an attacker. Then, the merchant verifies $S$ using the issuer's public key and checks if $m$ is containing the correct $NID_C$ and $A$. If the above verification is correct, then the merchant generates $(S, m, h(TInfo), ID_M, TS_M, Price, Issuer\_ID, Acquirer\_ID)_{KS_{M-PG_y}}$ and sends it to the payment gateway.

*Step 5:* The payment gateway decrypts the above message and sends ($S$, $NID_C$, $ID_M$, $h(TInfo)$, $Price$, $Acquirer\_ID$) to the issuer using the private network of the banks. In addition, the payment gateway also sends ($h(TInfo)$, $Price$, $ID_M$, $Issuer\_ID$) to the acquirer.

*Step 6:* The issuer verifies and approves the transaction. Then, the issuer transfers the money to the merchant account in the acquirer by the following sub-steps.

*Step 6-1:* The issuer verifies the validity of $S$ and uses $NID_C$ to get the real identity $ID_C$ in its database. Then, the issuer checks if the client's credit is enough to pay the amount of the $Price$. If the above verifications are correct, then the issuer transfers the money to the merchant account in the acquirer. Finally, the issuer sends $PResponse$ and $h(TInfo)$ to the payment gateway.

*Step 6-2:* After receive the money, the acquirer checks if the amount of the money is correct. If it is correct, then the acquirer sends $Stt$ and $h(TInfo)$ to the payment gateway.

*Step 7:* The payment gateway sends ($PResponse$, $h(TInfo)$) and ($Stt$, $h(TInfo)$) to the client and the merchant, respectively. Finally, both the client and the merchant can check the result of the transaction.

Fig. 3 shows the above steps of the proposed mechanism. According to the above steps, the proposed mechanism has the following advantages. First, the client uses the anonymous identity $NID_C$ to generate the payment information. Thus, the merchant does not know who buys the goods. In addition, the issuer does not know what the client buys because $TInfo$ and $m$ are protected by the one-way hash function. That is, the client buying privacy can be well-protected. Second, the client's bank generates the signature $S$ as the payment proof. Thus, the merchant can get the payment from the issuer by using $S$ even if the malicious client denies the transaction. Third, the client does not need to keep a session key between the client and the merchant in the proposed mechanism. Thus, the client does not maintain many session keys if he/she wants to transact with different merchants. Compared with Isaac and Zeadally's mechanism, the proposed mechanism has less session keys so the computation and communication costs can be also reduced. Therefore, the proposed mobile payment mechanism is very suitable for cloud computing environments.

## 4. Analysis and discussions

In this section, we perform some possible attacks to analyze the security of the proposed mechanism. The detailed analyses are described as follows.

### 4.1. Outsider attack

Assume that an attacker wants to get the payment information from $SRequest = (h(TInfo), ID_C, NID_C, h(m), Price, TS_C)_{KS_{C-I_t}}$, then he/she tries to decrypt $SRequest$. However, this attack is impossible because the attacker does not know the session key $KS_{C-I_t}$. Without knowing the session key, the attacker cannot break the symmetric encryption (Menezes et al., 1996). Similarly, it is impossible that an attacker tries to get the secret information from $(S, m, h(TInfo), TS_C, Issuer\_ID)_{KS_{C-PG_a}}$ and $(S, m, h(TInfo), TS_C, Issuer\_ID)_{KS_{M-PG_y}}$. Therefore, the outsider attack is infeasible for the proposed mechanism.

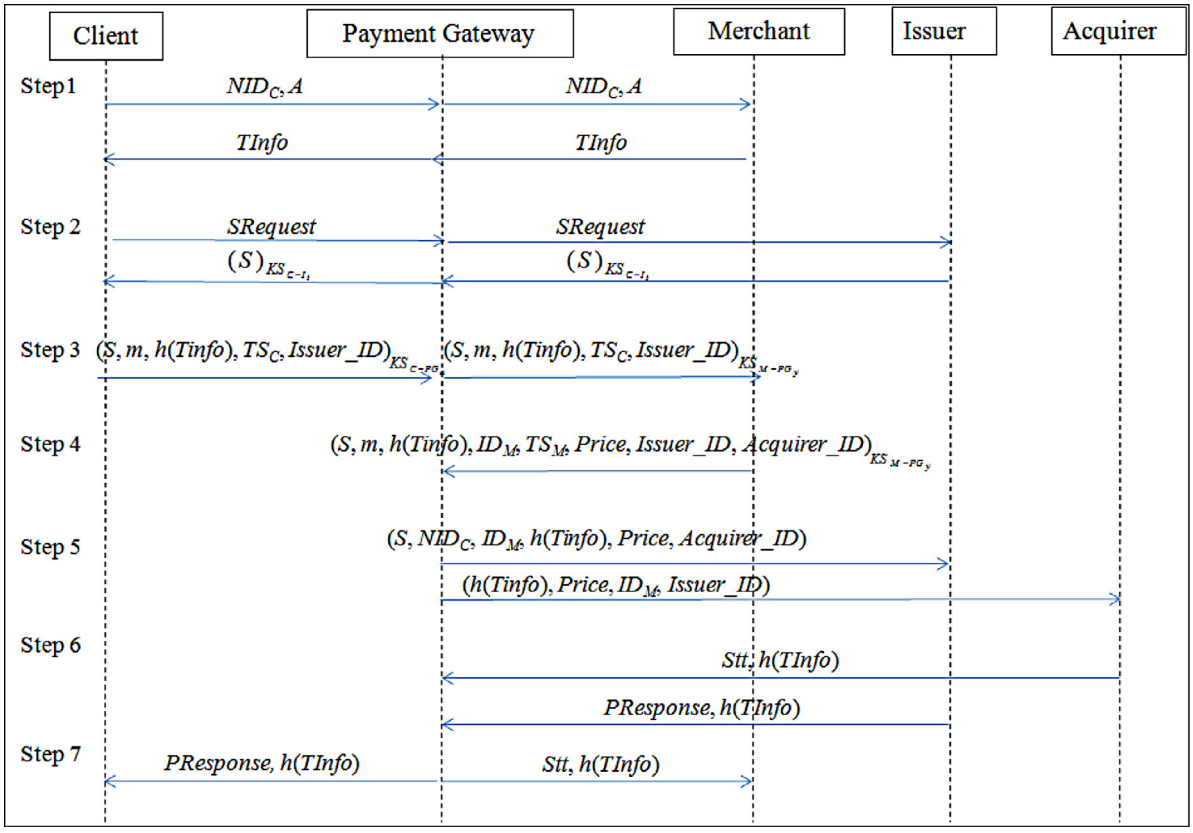


**Fig. 3.** The steps of the proposed e-payment mechanism.

### 4.2. Insider attack

Assume that the issuer wants to know what the client buys from $SRequest = (h(TInfo), ID_C, NID_C, h(m), Price, TS_C)_{KS_{C-I_t}}$. However, it is impossible because the payment information $m$ is protected by a one-way hash function. Thus, the issuer cannot know what the client buys so the buying privacy can be protected.

### 4.3. Replay attack

Assume that an attacker eavesdrops the communication between the client and the payment gateway to get $(S, m, h(TInfo), TS_C, Issuer\_ID)_{KS_{C-PG_a}}$. Then, the attacker resends the message to the payment gateway and tries to transact with the merchant. However, this attack is infeasible because $(S, m, h(TInfo), TS_C, Issuer\_ID)_{KS_{C-PG_a}}$ contains a timestamp $TS_C$, which indicates the real client buying time. The payment gateway can know the message is sent by an attacker by checking the timestamp and the message received time $T$. If $(T - TS_C)$ is larger than a reasonable time interval, then the merchant will discover that the message is sent by an attacker. Thus, the proposed mechanism can prevent from the replay attack.

### 4.4. Anonymity and double spending

The proposed mechanism provides anonymity because all the transaction steps are designed by using the client's anonymous identity $NID_C$. Thus, the client can transact with the merchant anonymously.

On the other hand, the proposed mechanism can prevent double spending because the payment information $m$ contains *TInfo*, which indicates transaction time, date, and the serial number. If a malicious client wants to use old $S_m$ and $m$ to transact again, then the merchant can discover the double-spending behavior by checking *TInfo* in its database.

### 4.5. Non-repudiation

Assume that a malicious client denies a legal transaction that he/she has made. In this case, the merchant can use $S$ as a payment proof because $S$ is signed by the issuer. After verifying $S$, the issuer will transfer the money to the merchant's account even if the client denies this transaction. Thus, the proposed mechanism provides non-repudiation security requirement.

We analyze the computation costs of Isaac and Zeadally (2012), Pourghomi et al. (2014) and our mechanism in Table 3. The

**Table 3**
Analyses for the computation costs of the related works.

| Roles | Method | | |
|---|---|---|---|
| | Isaac and Zeadally (2012) | Pourghomi et al. (2014) | The proposed mechanism |
| Client | 4Sym + 4Hash | 7Sym | 3Sym + 2Hash |
| Merchant | 5Sym + 2Hash | 3Sym | 2Sym + 1Hash |
| Gateway | 3Sym | 8Sym | 2Sym |

notations Sym and Hash are symmetric encryption/decryption and one-way hash function computations, respectively. According to Table 3, the computation cost of the proposed mechanism is less than those of Isaac and Zeadally (2012) and Pourghomi et al. (2014). Therefore, the proposed mechanism is more suitable and practical for the mobile payment in cloud computing.

## 5. Conclusions

In this paper, we proposed a new mobile payment mechanism with anonymity for cloud computing. The proposed mechanism provides the security requirements of confidentiality, anonymity, integrity, fairness, and non-reputation. Compared with the related works, the proposed e-payment mechanism has less computation and communication costs. Therefore, the proposed mobile payment mechanism is very efficient and suitable for cloud computing applications.

## Acknowledgement

This research was supported by the National Science Council, Taiwan, under contract No. MOST 103-2410-H-424-010 and NSC 102-2221-E-155-035-MY3.

## References

Abad-Peiro, J., Asokan, N., Steiner, M., Waidner, M., 1998. Designing a generic payment service. IBM Syst. J. 37 (1), 72–88.

Alpár, G., Batina, L., Verdul, R., 2012. Using NFC phones for proving credentials. Measurement, Modelling, and Evaluation of Computing Systems and Dependability and Fault Tolerance, Lecture Notes in Computer Science, 7201, pp. 317–330.

Chari, S., Kermani, P., Smith, S., Tassiulas, L., 2001. Security issues in m-commerce: a usage-based taxonomy. E-Commer. Agents 2033, 264–282.

GSMA, 2014. Case Study: ISIS Mobile Wallet. Available at: http://www.gsma.com/digitalcommerce/wp-content/uploads/2013/10/Mobile-Wallet-Case-Study-ISIS-digital-version1.pdf.

Host Card Emulation (HCE) 101, 2014. A Smart Card Alliance Mobile and NFC Council White Paper. Available at: http://www.smartcardalliance.org/wp-content/uploads/HCE-101-WP-FINAL-081114-clean.pdf.

Isaac, J., Zeadally, S., 2012. An anonymous secure payment protocol in a payment gateway centric model. Proc. Comput. Sci. 10 (2), 758–765.

Kounelis, I., Loschner, J., Shaw, D., Scheer, S., 2012. Security of service requests for cloud based m-commerce. In: Proceedings of IEEE International Convention, pp. 1479–1483.

Kungpisdan, S., Srinivasan, B., Le, P., 2003. Lightweight mobile credit-card payment protocol. In: Proceedings of 4th International Conference on Cryptology. India, pp. 295–308.

Madlmayr, G., Langer, J., Scharinger, J., 2008. Managing an NFC ecosystem. In: Proceedings of IEEE International Conference on Mobile Business, pp. 95–101.

Martinez-Pelaez, R., Rico-Novella, F., Satizabal, C., 2010. Study of mobile payment protocols and its performance evaluation on mobile devices. International Journal of Information Technology and Management 9 (3), 337–356.

Menezes, A., Orschot, P., Vanstone, S., 1996. Hand-Book of Applied Cryptography. CRC Press.

Moss, A., 2012. Google Wallet Implementation and Looking Beyond. Available at: http://global.verifone.com/company/resources/.

Pailles, J., Gaber, C., Alimi, V., Pasquet, M., 2010. Payment and privacy: a key for the development of NFC mobile. In: Proceedings of 2010 International Symposium in Collaborative Technologies and Systems (CTS), pp. 378–385.

Pourghomi, P., Saeed, M., Ghinea, G., 2013. Trusted integration of cloud-based NFC transaction players. In: Proceedings of IEEE International Conference for Information Assurance and Security (IAS), pp. 6–12.

Pourghomi, P., Saeed, M., Ghinea, G., 2014. A secure cloud-based NFC mobile payment protocol. Int. J. Adv. Comput. Sci. Appl. 5 (10), 24–31.

Pourghoumi, P., Ghinea, G., 2012. Challenges of managing secure elements within the NFC ecosystem. In: Proceedings of IEEE International Conference for Internet Technology and Secured Transactions (ICITST), pp. 720–725.

Pourghoumi, P., Ghinea, G., 2012. Managing NFC payment applications through cloud computing. In: Proceedings of IEEE International Conference for Internet Technology and Secured Transactions (ICITST), pp. 772–777.

Rivest, R., Shamir, A., Adleman, L., 1978. A method for obtaining digital signatures and public key cryptosystems. Commun. ACM 21 (2), 120–126.

Sun, P., Luo, J., Liu, Y., 2010. Perceived risk and trust in online group buying context. In: *Proceedings of International Conference on Information Management, Innovation Management and Industrial Engineering*, 3. Kunming, China,, pp. 660–663.

Tsai, M., Cheng, N., Chen, K., 2011. Understanding online group buying intention: the roles of sense of virtual community and technology acceptance factors. Total Qual. Manage. Bus. Excell. 22 (10), 1091–1104.

Wei, K., Chen, Y., Smith, A., Vo, B., May 2005. Whopay: a scalable and anonymous payment system for peer-to-peer environments. In: Proceedings of 26th International Conference on Distributed Computing Systems, pp. 1–14.

**Jen-Ho Yang** received the B.S. degree in computer science and information engineering from I-Shou University, Kaoshiung, Taiwan in 2002. He is currently pursuing his Ph.D. degree in computer science and information engineering from National Chung Cheng University, Chiayi, Taiwan. His current research interests include electronic commerce, information security, cryptography, mobile communications, and fast modular multiplication algorithm.

**Pei-Yu Lin** received the M.S. and Ph.D. degrees in computer science and information engineering from National Chung Cheng University, Chiayi, Taiwan, in 2004 and 2009, respectively. Since 2009, she has been an Assistant Professor in the Department of Information communication at Yuan Ze University, Chung-Li, Taiwan. Her current research interests include image protection, data mining and information security.