Mobile Information Systems 4 (2008) 51–68
IOS Press

Employ a mobile agent for making a payment

Yan Wang[a,*], Duncan S. Wong[b] and Huaxiong Wang[a,c]
[a]*Department of Computing, Macquarie University, Sydney, Australia*
[b]*Department of Computer Science, City University of Hong Kong, Hong Kong*
E-mail: duncan@cs.cityu.edu.hk
[c]*Division of Mathematical Sciences, School of Physical & Mathematical Sciences, Nanyang Technological University, Singapore*
E-mail: hxwang@ntu.edu.sg

Abstract. The mobile agent paradigm offers flexibility and autonomy to e-commerce applications. But it is challenging to employ a mobile agent to make a payment due to the security consideration. In this paper, we propose a new agent-assisted secure payment protocol, which is based on SET payment protocol and aims at enabling the dispatched consumer-agent to autonomously sign contracts and make the payment on behalf of the cardholder after having found the best merchant, without the possibility of disclosing any secret to any participant. This is realized by adopting the Signature-Share scheme, and employing a Trusted Third Party (TTP). In the proposed protocol, the principle that each participant knows what is strictly necessary for his/her role is followed as in SET. In addition, mechanisms have been devised for preventing and detecting double payment, overspending and overpayment attacks. Finally the security properties of the proposed protocol are studied analytically. In comparison with other existing models, the proposed protocol is more efficient and can detect more attacks.

Keywords: Mobile agent, secure payment protocol, SET and Signature-Share scheme

1. Introduction

Autonomous agents, stationary or mobile, offer new paradigms with autonomy, intelligence and flexibility. Autonomous agent based e-commerce technologies have drawn attentions from both the research community [7,11,15] and applications (e.g., Amazon [1] and eBay [2]). The introduction of autonomous agents acting on behalf of end-consumers could reduce the effort required from users to conduct e-commerce transactions by automating a variety of activities, such as, looking for and filtering out online shops selling the specified products, requesting offers, negotiating with shops and even completing payments [4,7,22]. Due to the feature of mobile agent paradigm, an agent can be taken as a special service integrated as part of web services, namely mobile service, that is more suitable for handheld devices (e.g. PDA) with the interface of wireless communication [8,9]. In the mobile agent paradigm, the network connection is required only when the agent is dispatched to remote servers and the result collected by the dispatched agent is sent back to the agent server/owner. For handheld devices, they are powered by batteries. Each battery can supply power for a few hours. In addition, the capacity of the CPU and RAM is limited too. Due to these features, it is a good choice to leave the computation task to other servers, during which the network connection is not necessary. The mobile agent paradigm

*Corresponding author. Tel.: +61 2 98509539; Fax: +61 2 98509551; E-mail: yanwang@ics.mq.edu.au.

is the right choice to bridge the handheld devices and remote servers. The latter can be the platforms of merchants in an e-commerce environment.

However, with respect to security, the introduction of mobile agents increases the risk as each agent is exposed to the visited servers [6,17,19]. Some studies have been done for protecting the offers carried by a roaming buyer agent [5,19]. But employing a mobile agent for making payment is always a challenging issue as it is not feasible to ask the agent to carry critical/confidential information (e.g. credit card information) even secret keys when visiting a set of remote hosts as this will expose sensitive data to potentially hostile environments [10].

In applications, most online payment protocols are based on SSL or S-HTTP. But as credit card information is stored on the merchant server, they are not considered secure enough. SET (Secure Electronic Transaction) protocol developed by VISA and MasterCard is regarded as a better protocol [3] aiming at protecting users' credit card information with important properties, such as authentication of the participants, data integrity and confidentiality. In SET, the credit card information is encrypted by the public key of the payment gateway. Therefore it is protected against the merchant and other parties.

In the literature, some protocols have been proposed aiming at employing one mobile agent to fulfill the payment task, such as SET/A [16], SET/A+[25], LITESET/A+ [14] and LITESET/A++ [20]. In the best case, one mobile agent is expected to be employed for searching shops/offers, negotiate with shops and complete the transaction including payment with the best seller with the best offer.

However, as we analyzed in Section 5 in this paper, the above agent-based payment protocols have various problems. For example, all protocols lack the mechanism preventing overspending and over payment problems. Most protocols except LITESET/A++ have no mechanism preventing double payment.

In this paper, we proposed a new protocol based on SET aiming at enabling a mobile agent to automatically and autonomously make final transactions and payment with the "best" merchant with the best offer without interacting with the end-consumer after having performed all kinds of tasks including asking for offers, and negotiating with merchants. This requires the capability of the agent in the protocol to dynamically sign with the "best" merchant, which is not determined in advance. Then the payment instruction is dynamically passed to the payment gateway (PG), which can be determined only after the interaction with the merchant according to the brand of the credit card. Hence encrypting everything in advance is impossible while asking the agent to carry any key for encryption is certainly a risk. The proposed protocol is based on Signcryption algorithm [26], which alleviates the burden for encryption and signature generation consuming less resource and hence is more suitable to mobile agent environments. Meanwhile, by adopting the Signature-Share scheme, the agent can sign contracts and pass the payment instruction to the PG in cooperation with independent TTP without the possibility of disclosing any secret to other participants. In the proposed protocol, a symmetric key is used to encrypt the payment instruction (PI), which is more efficient than the public key scheme used in LITESET/A++. In addition, in the proposed protocol, we have designed mechanisms preventing overspending and over payment problems.

The rest of this paper is organized as follows. Section 2 briefly reviews SET and Signature-Share scheme. The proposed protocol is presented in Section 3 and its features and security properties are analyzed in Section 4. In Section 5, we analyze existing agent-based secure payment protocols and compare them with the proposed protocol. Section 6 finally concludes our work in this paper.

Table 1
Notions

$K_{\rightarrow PG}$	the session key used for encrypting PI that should be passed to PG.
$c_{\rightarrow A}$	the ciphertext *passed to* participant A
C	card holder
CA	cardholder agent
$C_K(A)$	key-exchange certificate of participant A
$C_S(A)$	signature certificate of participant A
$E_k\{m\}$	message m encrypted by key k
$E_{PG}\{K, PI\}$	the digital envelope generated by PG (= $\{E_{yK_{PG}}\{K\}, E_K\{PI\}\}$) K is a symmetric key
g	a (random) integer in $[1, \ldots, p-1]$ with order $q \ mod \ p$ (public to all)
$H(m)$	a one-way *collision-resistant* hash function applied to message m
I_A	unique transaction number issued by A
KH	a keyed one-way hash function
M	merchant
OI	order information
p	a large prime (public to all)
PA	payment agent
PG	payment gateway
PI	payment instruction including card number, expiry date etc
q	a large prime factor of $p-1$ (public to all)
$r_{\rightarrow A}$	the hash value that should be *passed to* participant A
R	a random number chosen from $[1, \ldots, q]$
$s_{i_{\rightarrow A}}$	the ith shared signature that should be *passed to* participant A
SIG_A	the signature generated by participant A
T_e	a timestamp when the purchase request expires
T_{i_A}	the ith timestamp at participant A
TR_A	transaction record kept by participant A
(y_{K_A}, x_{K_A})	(*public key, secret key*) of participant A for encryption and decryption
(y_{S_A}, x_{S_A})	signature ($public\ key,\ secret\ key$) of participant A
z	a random number chosen from $[1, \ldots, q]$
$X\|\|Y$	concatenation of two messages X and Y
$A \rightarrow B : m$	A sends a message m to B

2. Background

In this section, we will briefly review SET [3] and Signature-Share scheme [14]. The notations and symbols used in this paper are listed in Table 1.

2.1. SET

The SET protocol [3] is composed of several kinds of transactions, ranging from registration of participants, to purchase request and payment processing. There are different roles in SET. They are cardholder (C), credit card issuer, merchant, acquirer and payment gateway (PG) [3]. PG is a device of acquirer where the merchant has an account. As requested by the PG, successful payment should be finally authorized by the card issuer whereafter the issuer will pay on behalf of the cardholder and the money will be deposited to the merchant's account at the acquirer.

SET uses two distinct asymmetric key pairs for each party, one for key-exchange. The corresponding public key y_{K_A} is contained in public key certificate $C_K(A)$ of participant A. The key pairs (y_{K_A}, x_{K_A}) are used for encrypting and decrypting messages. Another key pair is used for the creation and verification of signatures. The signature public key of participant A is included in the signature certificate $C_S(A)$. Figure 1 depicts the purchase request phase of SET.

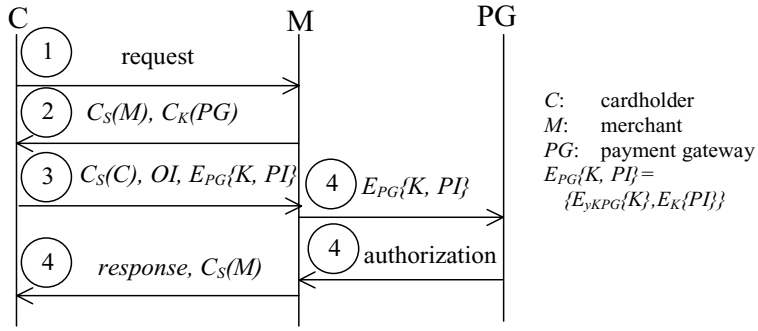

Fig. 1. SET purchase request transaction.

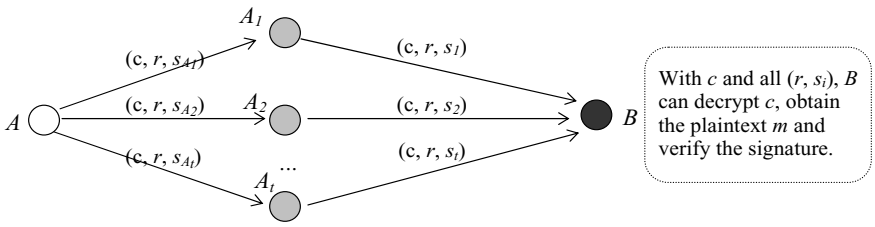

Fig. 2. Signature-Share scheme.

In SET, the key issue is to pass the payment instruction (PI) including card number, cardholder's name and expiry date to the payment gateway (PG) determined according to the brand of the cardholder's credit card that is included in purchase request (in step 1 in Fig. 1). PI is encrypted by a symmetric session key K that is contained in a digital envelope $E_{PG}\{K, PI\}$ passed to PG via merchant M. Finally the payment can be completed by PG without the possibility of disclosing PI to M. Due to the limited space, readers can refer to [3] for more details.

2.2. Signature-Share scheme

The Signature-Share scheme [14] (see Fig. 2) is based on Signcryption public key algorithm (see Appendix 1). In this scheme, the sender A wants to send a message m to recipient B through t sharing parties, say A_i $(i = 1, \ldots, t)$. The signature key of A is shared by t parties, namely, $x_A = x_{A_1} + x_{A_2} + \ldots + x_{A_t}$. Each party generates the shared signature s_i on the hash value r of message m, and all shared signatures are sent to B. With all (r, s_i), B can verify the signature and hence check the data integrity of m.

Details of the Signature-Share scheme are described in Appendix 2.

3. The proposed protocol

In the proposed protocol, Signature-Share scheme is adopted for passing securely the order information to the merchant. The cardholder's signature secret key is divided into two parts. The first part is kept by the cardholder. The second part is encrypted using the public key of the TTP and will be passed to the TTP for generating shared signatures. The dispatched agent does not carry any shared signature secret

key. Instead it only carries one half shared signature signed on the order information (OI) and payment instruction (PI) respectively by the cardholder that should be sent to the merchant M. The other half shared signature is generated with the assistance of the TTP. On obtaining the two shared signatures (i.e. $s_{1\rightarrow M}$ and $s_{2\rightarrow M}$), the merchant M can verify the order information (OI) and check the data integrity. Meanwhile the payment instruction (PI) is encrypted by a symmetric session key, It can be passed to PG via the merchant and can be decrypted by PG only. Additionally, in the proposed protocol, mechanisms are also provided for preventing and detecting double payment, overspending and overpayment attacks.

3.1. Secret-sharing of cardholder's signature secret key x_{S_C}

In the proposed protocol, the cardholder and TTP share the cardholder's signature secret key x_{S_C} based on Shamir-threshold scheme [12].

$$x_{S_C} = x_{S_{C_1}} + x_{S_{TTP}}$$

According to the two share schemes presented in Section 2.2, $A_1 = C$ and $A_2 = TTP$. $x_{S_{C_1}}$ is kept by C as a secret key always while $x_{S_{TTP}}$ can be carried by the agent after being encrypted using the TTP's public key and will be passed to the TTP for generating the second shared signature that will be passed to M.

3.2. Description of the protocol

Step 1: Cardholder C (i.e. C's software) generates a temporary session key $K_{\rightarrow PG}$ for the payment gateway.

1) Then C uses $K_{\rightarrow PG}$ to encrypt the payment instruction (PI):

$$c_{\rightarrow PG} = E_{K_{\rightarrow PG}}\{PI\}$$

and generate ciphertext

$$E_{y_{K_{TTP}}}\{x_{S_{TTP}}||z||(K_{\rightarrow PG} + R + I_C + T_C + T_e)\}.$$

where

– R is a random number chosen from $[1, \ldots, q]$;
– I_C is the transaction identifier assigned by cardholder C;
– T_C is the timestamp at C when to complete the encryption and shared signature generation;
– T_e ($T_e > T_C$) is the timestamp when the purchase request expires. It is unique to each purchase order.

2) Meanwhile, C generates the first half shared signature $s_{1\rightarrow M}$ on the hash value that will be passed to merchant M:

$$r_{\rightarrow M} = H(g^z \bmod p, H(PI)||H(OI)||H(C_S(C)||I_C||T_C||T_e))$$

$$s_{1\rightarrow M} = z/(r_{\rightarrow M} + x_{S_{C_1}}) \bmod q$$

where

– OI is the description and constraint for the order of *Product*, namely,

$$OI = \textit{OrderDescription, PriceLimit,}\ x_{S_C}(H(\textit{OrderDescription, PriceLimit,}\ T_C))$$

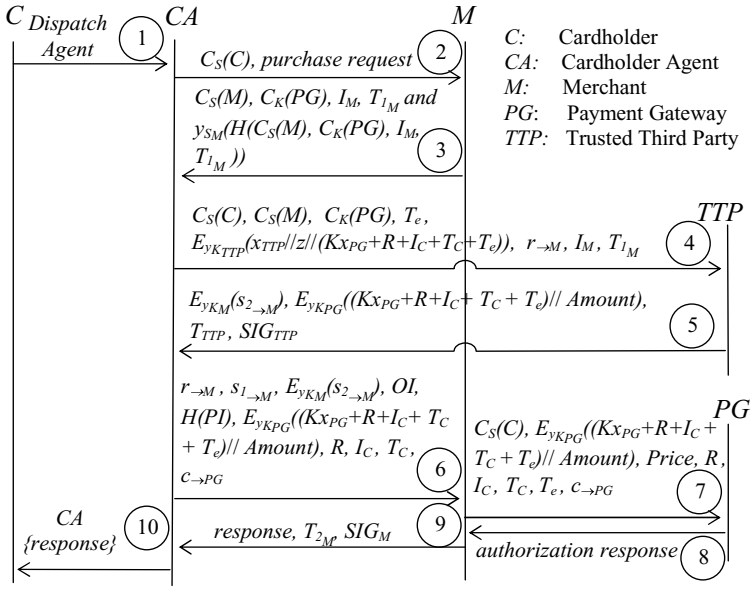

Fig. 3. Purchase request in the proposed protocol.

3) Then C dispatches the consumer agent CA encapsulating the following arguments $C_S(C)$,

$$E_{y_{K_{TTP}}}\{x_{S_{TTP}}||z||(K_{\rightarrow PG}+R+I_C+T_C+T_e)\},$$

$$OI,\ H(PI),\ R,\ I_C,\ T_C, T_e,\ r_{\rightarrow M},\ s_{1_{\rightarrow M}},\ c_{\rightarrow PG}$$

The dispatched agent will visit a set of merchants asking offers and negotiating with them. An offer evaluation model and negotiation model can be found in [22].

Step 2: After completing the negotiation with some merchants, the agent chooses the best one -M- with the best offer to make the deal and send M the purchase request (see Fig. 3). The request contains the brand of the credit card that will be used for payment.

$$CA \rightarrow M:\ C_S(C),\ \text{purchase request},\ T_e$$

Step 3: After receiving the request, M verifies $C_S(C)$ and reply CA.

$$M \rightarrow CA: C_S(M),\ C_K(PG),\ C_K(M),\ I_M,\ T_{1_M} \text{ and}$$
$$x_{S_M}(H(C_S(M),\ C_K(PG),\ C_K(M),\ I_M,\ T_{1_M}))$$

where

– I_M is a unique transaction number issued by M and T_{1_M} is the current timestamp at M;
– $x_{S_M}(H(C_S(M),\ C_K(PG),\ C_K(M),\ I_M,\ T_{1_M}))$ is the signature generated by M;
– PG is the payment gateway that is determined according to the brand of C's credit card, which is contained in the purchase request.

Step 4: From M's reply, CA obtains the public key certificate of the payment gateway. Then CA sends TTP a message so that $s_{2_{\rightarrow M}}$ can be generated by TTP.

$$CA \rightarrow TTP:\ C_S(C),\ C_S(M),\ C_K(PG),\ C_K(M),\ T_e,\ OI,\ \textit{Amount},$$
$$E_{y_{K_{TTP}}}\{x_{S_{TTP}}||z||(K_{\rightarrow PG}+R+I_C+T_C+T_e)\},\ r_{\rightarrow M},\ I_M,\ T_{1_M}$$

where

– *Amount* = *Price*. *Price* is the price of the Product, which is determined by CA and M after the negotiation between them. *Amount* is the variable sent by CA. *Price* is the variable sent by M in Step 7. Here we distinguish *Amount* and *Price* as both of them will be passed to the PG where a consistency verification will be performed (in Step 8).

Step 5: On receiving the message, TTP verifies the validation of $C_S(C)$, $C_S(M)$, $C_K(M)$ and $C_K(PG)$, checks whether the current time $T < T_e$ and *Amount* $\leqslant$ *PriceLimit*. If all are correct, TTP decrypts the ciphertext from CA obtaining z and $x_{S_{TTP}}$, generates the 2nd half shared signature on hash value $r_{\to M}$,

$$s_{2_{\to M}} = z/(r_{\to M} + x_{S_{TTP}}) \; mod \; q$$

and generates

$$E_{y_{K_{PG}}}\{(K_{\to PG} + R + I_C + T_C + T_e)||\textit{Amount}\} \text{ and } E_{y_{K_M}}\{s_{2_{\to M}}\}$$

Note: TTP knows $(K_{\to PG} + R + I_C + T_C + T_e)$ but doesn't know $K_{\to PG}$.
Hereafter TTP keeps

$$TR_{TTP} = \{C_S(C),\ C_S(M),\ C_K(M),\ C_K(PG),\ I_C,\ T_C,\ T_e,\ I_M,\ T_{1_M},\ T_{TTP},\\ x_{S_M}(H(C_S(C),\ C_S(M),\ C_K(M),\ C_K(PG),\ I_M,\ T_{1_M}))\}$$

as a transaction record and sends a message to CA.

$$TTP \to CA:\ E_{y_{K_M}}\{s_{2_{\to M}}\},\ E_{y_{K_{PG}}}\{(K_{\to PG} + R + I_C + T_C + T_e)||\textit{Amount}\},\\ T_{TTP},\ SIG_{TTP}$$

where

– T_{TTP} is the timestamp at TTP when to generate the shared signature (i.e. $s_{2_{\to M}}$);
– $SIG_{TTP} = x_{S_{TTP}}(H(C_S(C),\ C_S(M),\ C_K(PG),\ E_{y_{K_M}}\{s_{2_{\to M}}\},\ E_{y_{K_{PG}}}\{(K_{\to PG} + R + I_C + T_C + T_e)||\textit{Amount}\},\ r_{\to M},\ I_M,\ T_{TTP}))$ is the signature generated by TTP that can be kept by the cardholder as a non-repudiation receipt.

Step 6: Once receiving the message from TTP, CA sends a message to the merchant.

$$CA \to M:\ C_K(C),\ r_{\to M},\ s_{1_{\to M}},\ E_{y_{K_M}}\{s_{2_{\to M}}\},\ OI,\\ H(PI),\ E_{y_{K_{PG}}}\{(K_{\to PG} + R + I_C + T_C + T_e)||\textit{Amount}\},\ R,\ I_C,\ T_C,\ c_{\to PG}$$

Step 7: After having received the message, M computes v by applying the Signature-Share scheme

$$v = H((y_{K_C} \cdot g^{2 \cdot r_{\to M}})^{(\sum_{i=1}^{2} {s_{i_{\to M}}}^{-1})^{-1}} \; mod \; p)$$

and verify signature

$$H(v, H(PI)||H(OI)||H(C_S(C)||I_C||T_C||T_e)) \stackrel{?}{=} r_{\to M}$$

If it holds and the current time $T < T_e$, M keeps

$$TR_M = \{C_S(C),\ r_{\to M},\ s_{1_{\to M}},\ s_{2_{\to M}},\ OI,\ H(PI),\ I_C,\ T_C,\ T_e\}$$

as a transaction record.
Then M sends a message to PG.

$$M \to PG:\ E_{y_{K_{PG}}}\{(K_{\to PG} + R + I_C + T_C + T_e)||\textit{Amount}\},\ \textit{Price},\ R,\ I_C,\ T_C,\ T_e,\ c_{\to PG},\ s_{1_{\to PG}}$$

Step 8: From the message, PG obtains $E_{y_{K_{PG}}}\{(K_{\to PG} + R + I_C + T_C + T_e)||\textit{Amount}\}$. After decryption, it obtains $K_{\to PG}$ and *Amount*. Hereafter PG can decrypt $c_{\to PG}$ and thus obtain PI: If current time $T < T_e$ and *Amount* = *Price*, PG will send M an authorization response.

Step 9: After processing the order, the merchant generates and signs a purchase response, and sends it to the agent.

$$M \to CA : \textit{PurchaseResponse},\ T_{2_M}, SIG_M$$

where

– T_{2_M} is the timestamp ($T_{2_M} > T_{1_M}$) at M when SIG_M is issued;
– $SIG_M = x_{S_M}(H(\textit{PurchaseResponse},\ C_S(M),\ r_{\to M},\ s_{1_{\to M}},\ s_{2_{\to M}},\ OI,\ H(PI),\ \textit{Price},\ I_C,\ T_C,\ T_e,\ I_M,\ T_{2_M}))$ is the signature generated by M at time T_{2_M}. It will be finally passed to the cardholder as a non-repudiation receipt by the agent.

If the payment is authorized, the merchant will fulfill the order by delivering the product bought by the cardholder.

Step 10: The agent verifies the merchant signature certificate, checks the digital signature of the response, and then returns back to its owner carrying

$$\textit{PurchaseResponse},\ C_S(M),\ C_S(TTP),\ C_K(PG),\ C_K(M),\ \textit{Amount},\ T_{TTP},\ T_{1_M},\ T_{2_M},\ I_M,\ E_{y_{K_{PG}}}\{(K_{\to PG} + R + I_C + T_C + T_e)||\textit{Amount}\},\ SIG_{TTP},\ SIG_M.$$

The owner takes appropriate actions based on the obtained contents.

4. Security analysis

From the above description, we could observe that in cooperation with TTP, the agent can sign contracts with the dynamically chosen merchant M and make payment. In this section, we will analyze the security properties of the proposed protocol focusing on the following possible issues.

– whether it is possible for any participant to re-generate the secret signature key of the cardholder (ATK1);
– whether it is possible for any participant except PG to obtain the payment instruction (ATK2);
– whether it is possible for any participant to re-perform the payment (double payment, ATK3);
– whether it is possible for the agent to pay more than required by the cardholder (overspend, ATK4), and

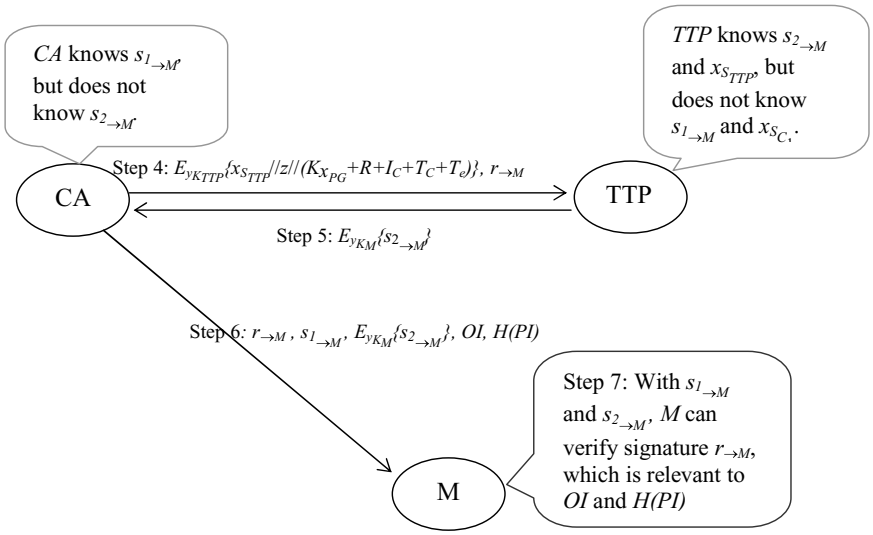

Fig. 4. Shares passed to the merchant.

– whether it is possible for the merchant to pass a wrong price to the PG (overpayment, ATK5).

1. In this protocol, the dispatched agent CA does not have any task for encryption, decryption or signing. So it is not necessary for it to carry any key. Therefore, the agent in the transaction is more of a messenger. Most of the encryption and signing works are done by TTP. What the agent should do is to communicate with different participants sending relevant messages to them.
2. CA carries one shared half signature -$s_{1\to M}$. But it is generated by cardholder C and the shared secret key $x_{S_{C_1}}$ is kept by C. No irrelevant party could obtain both of the two shared signatures (i.e. $s_{1\to M}$ and $s_{2\to M}$) together with some arguments (i.e. r and z); so it is not possible for any party to obtain two shared secret keys so as to generate the secret signature key of the cardholder (i.e. x_{S_C}).
   For instance, for the merchant, it can obtain the $r_{\to M}$, $s_{1\to M}$, $s_{2\to M}$, $c_{\to PG}$ and $H(PI)$, but cannot obtain PI. Argument z is also protected against the merchant. So it is not possible for M to obtain x_{S_C} (ATK1) (see Fig. 4).
   Likewise, TTP knows $(K_{\to PG} + R + I_C + T_C + T_e)$ but doesn't know $K_{\to PG}$. Meanwhile $c_{\to PG} = E_{K_{\to PG}}(PI)$ is not passed to TTP. As $s_{1\to M}$ is not passed to TTP, TTP cannot generate $x_{S_{C_1}}$ so as to re-generate x_{S_C}.
   In the proposed protocol, the cardholder's secret signature key can be re-generated only if M and TTP collude. But it is impossible regarding the nature of TTP.
3. Once obtaining $E_{y_{K_{PG}}}\{(K_{\to PG} + R + I_C + T_C)||Amount\}$, PG can decrypt it and obtain $K_{\to PG}$. Thus PG can decrypt $E_{\to PG}(PI)$ and obtain the payment instruction PI. M knows $E_{y_{K_{PG}}}\{(K_{\to PG} + R + I_C + T_C)||Amount\}$, but it doesn't know $K_{\to PG}$ (ATK2).
4. The property of non-repudiation has been improved. In terms of non-repudiation, timestamps are important in many electronic transactions indicating the time when a particular event or action has taken place [27]. T_e makes each purchase request from C unique preventing the re-payment attack. In addition, more timestamps are added in different stages, such as T_C, T_{TTP}, T_{1_M} and T_{2_M}. In the message from TTP to CA (in Step 5) and the message from M to CA (in Step 9), signatures are added including timestamps. These signatures adopt nested structure that can show

message exchange processes among CA, TTP and M. These signatures prevent the replay attack (double payment) (ATK3). Meanwhile the generation of signatures will not significantly increase the burden of the agent to migrate back to the cardholder since the signatures are generated on the hash value, which has a fixed length.

5. In the proposed protocol, *Amount*, the amount of the transaction that will be charged to the cardholder's account is first passed to the TTP by CA, which checks it with the limit of current transaction (i.e. *PriceLimit*) (ATK4). Moreover, the amount is included in the ciphertext by the TTP that will be passed to the PG where the comparison will be conducted with the price (i.e. *Price*) from M. This can prevent the overspending and overpayment attacks (ATK5).

5. Comparisons with existing agent-based secure payment protocols

5.1. The analysis of existing protocols

5.1.1. SET/A

SET/A protocol [16] is proposed to make SET adaptable to the mobile computing environments. Based on the principles used in purchase phase of SET, SET/A improves its performance only by adding a mobile agent for the cardholder to fulfill payment transaction since the cardholder need not frequently connect to Internet during the whole transaction phase. SET/A performs the same function of transaction as that in SET except that the mobile agent of SET/A replaces the cardholder of SET in the purchase phase.

For the protocol to be secure, aspects, such as, where the agent should execute securely, how to decrypt the encrypted information on OI and PI (i.e. $E_{K_?}\{OI\_PI\_data\}$ in Step 1) and where to generate the symmetric key to encrypt the PI (Step 4), are critical to a secure protocol. SET/A suggests running the agent in a tamper-proof environment [23] or a secure coprocessor [24] to protect the agent against malicious merchants. However, from the point of view the cardholder, it is insecure to expose some confidential information, such as the credit card information in the PI, to any merchant environment.

An alternative approach based on software using hidden computations [18] is also suggested in [16] without the cost of additional investment in hardware from each merchant. Another solution is given in [13], where an encrypted signature function is used so that a signature can be generated by the agent without the risk carrying and disclosing the secret signature key. But in this scheme, the function cannot be prevented from being abused and hence it is not secure and the non-repudiation property can hardly be ensured. If the cardholder gets PG's public key by sending a request to the merchant, the protocol is essentially the same as SET losing the autonomy and flexibility of mobile agents, which are the motivations of SET/A.

5.1.2. SET/A+

SET/A+ [25] is a more complex protocol, which adds a Trust Verification Center (TVC) in the payment system. The TVC keeps the sensitive information and charges cardholders or merchants by providing verification service.

The focus of SET/A+ is on how to pass securely the symmetric key K generated by the cardholder to PG so that PG can obtain PI that is encrypted by K (i.e. $E_K\{PI\}$).

In SET/A+, TVC is not only used for verification but also used for encrypting information. However, the agents are limited in their functionalities. For example, an agent cannot sign dynamically and perform encryption for the owner during trading (since it requires the secret key of the owner). The

agent carries the cardholder's signature -$x_{S_C}(H(H(OI)||H(PI)))$- generated in advance and passes it to the merchant to make a final deal. This is not flexible. In a malicious merchant environment, the signature can be abused easily and no sufficient non-repudiation mechanism is provided preventing the replay attack. This may cause the disputes between participants and result in the loss of the cardholder.

5.1.3. LITESET/A+

LITESET/A+ is based on Signcryption public-key algorithm [26] and Signature-Share scheme (called Signature-Threshold scheme in [14]). It employs a mobile agent and a Trusted Third Party (TTP). The role of TTP is the same as the TVC in SET/A+. The TTP can do both verification and encryption when necessary.

In this scheme, the most significant difference with SET/A+ is that it uses a Signature-Share scheme based on Shamir-threshold scheme [12,14]. The signature secret key of cardholder-x_{S_C} – is divided into two shared parts, say x_{S_A} for agent and $x_{S_{TTP}}$ for TTP.

$x_{S_A} = x'$, $x_{S_{TTP}} = x_{S_C} - x'$ where x' is a random number chosen from $[1, \ldots, q]$.

By carrying x_{S_A}, the agent A can sign over the order information OI and the hash value of payment instruction PI, yielding $x_{S_A}(H(H(OI)||H(PI)))$ Another shared signature $x_{S_{TTP}}(H(H(OI)||H(PI)))$ can be generated by TTP after it obtains its shared signature secret key $x_{S_{TTP}}$. Once obtaining the two shared signatures, by applying Signature-Share scheme, the merchant can verify the dual hash value $H(H(OI)||H(PI))$ and hence check the validity of the order and the payment data. It is equivalent to obtaining the cardholder's signature $x_{S_C}(H(H(OI)||H(PI)))$ as in SET/A+. But without the involvement of TTP, the merchant cannot obtain both shared signatures.

The aim of LITESET/A+ is to make the agent sign flexibly on behalf of the cardholder in cooperation with the TTP. This can be done after the process of negotiation with a set of merchants. So one consumer agent works well instead of dispatching a payment agent after the negotiation agent completes its tasks. By using Signature-Share scheme, the dispatched agent does not need to carry the cardholder's secret signature key. Instead, it carries its shared signature key x_{S_A}.

Thereafter the problem comes as the agent carries the shared secret key x_{S_A} and executes on a server provided by the chosen merchant where to make the final deal. For generating a shared signature s_A on a hash value r, the agent should also carry r and argument z (q is public), so as to compute

$$s_A = z/(r + x_{S_A}) \; mod \; q$$

This means x_{S_A}, r and z will be exposed to the merchant. Once the merchant obtains the shared signatures s_{TTP} from TTP (Step 8 in LITESET/A+), it can easily compute $x_{S_{TTP}}$ because

$$s_{TTP} = z/(r + x_{S_{TTP}}) \; mod \; q$$

Hence the merchant can obtain the cardholder's signature secret key

$$x_{S_C} = x_{S_A} + x_{S_{TTP}}$$

Moreover, the non-repudiation mechanism is weak in LITESET/A+. In addition to the obtained messages, a participant keeps I_C or I_M only as non-repudiation receipts. Though these identifiers are unique, no receipt can show when a transaction is completed. So I_C and I_M are not strictly non-repudiation receipts. This problem exists in SET/A and SET/A+ too.

5.1.4. LITESET/A++

LITESET/A++ [20] is based on Signcryption public-key algorithm [26], Signature-Share scheme and Signcryption-Share scheme (called Signature-Threshold scheme and Signcryption-Share scheme proposed in [14]). It solves the problem in LITESET/A+ that a shared secret key is carried by the dispatched agent. Instead, it is kept by the cardholder. LITESET/A++ keeps using Signature-Share scheme so that the dispatched agent has the capability to sign with the merchant chosen after dispatch in corporation with TTP. In addition, LITESET/A++ uses Signcryption-Share scheme to encrypt the payment instruction PI. In Signcryption-Share scheme, the secret signature key of sender A is shared by t parties. Each party generates the shared signature s_i on hash value r obtained from A, and all the shared signatures are sent to recipient B with the ciphertext c. With c and all (r, s_i), B can decrypt c, obtain the plaintext m and verify the signature.

In LITESET/A++, PI is encrypted by a *session public key*. The session secret key is encrypted by the cardholder. In corporation with TTP, PG can obtain the two shared signatures, hash value and ciphertext and thus decrypt the ciphertext and obtain the session secret key, with which PG can obtain the PI.

In terms of security, LITESET/A++ is a good protocol. But it has a few problems.

1. As Signcryption-Share scheme is a public key algorithm, it is less efficient than a symmetric key algorithm;
2. No mechanism is devised for preventing double payment, overspending attack and over-payment attack. This problem exists in all the above protocols.

5.2. Comparison of protocols

In some sense, the agent in SET/A+ has the same flexibility since the agent carries the cardholder's signature $x_{S_C}(H(H(OI)||H(PI)))$ signed on the order information and payment instruction. The signature can be passed to the merchant where the agent wants to make the deal. But the signature may be reused by any malicious merchant where a valid deal has been made. The merchant can mount a replay attack successfully by transferring the copy of a used digital envelope to the payment gateway causing the loss of the cardholder. This is because that its non-repudiation mechanism is weak. In addition, a visited merchant may abuse the pre-generated signature.

In the proposed protocol, the Signature-Share scheme avoids using a pre-generated signature. Meanwhile timestamps from different participants appear in the signatures of message senders. Also a unique expiry timestamp T_e appears in the signature from the cardholder and it can be sent to each participant. Hence a replay attack can be detected easily.

As we analyzed in Section 4 and Section 5, both LITESET/A++ and the proposed protocol correct the security flaw in LITESET/A+ so that it is not possible for the merchant to re-generate the signature secret key of the cardholder. Meanwhile the flexibility for the agent to "sign" on behalf of the cardholder and make a deal with the merchant remains unchanged. Moreover, with the involvement of TTP, the agent in the proposed protocol need not to do any encryption and decryption. In contrast, in SET/A+ and LITESET/A+, the agent executes at the merchant's server and completes the encryption operations.

In the proposed protocol, similar to LITESET/A+, Signature-Share scheme is adopted to give the capability of the dispatched agent to sign contract *dynamically* with M in corporation with TTP.

The proposed protocol inherits good features of LITESET/A++ in terms of the properties listed in Tables 2 and 3 while the focus of LITESET/A++ is how to pass PI securely to PG which is determined afterwards by M according to the brand of the credit card. That means C doesn't know PG beforehand so as to encrypt PI using PG's public key.

Table 2
Comparison of agent-based secure payment protocols (part I)

	Do encryption /decryption by agent	Execute at merchant's server	Sign dynamically
SET/A	yes	not specified	not specified
SET/A+	yes	yes	no
LITESET/A+	yes	yes[1]	yes
LITESET/A++	no	no	yes
proposed protocol	no	no[2]	yes

[1]The cardholder's secret signature key can be re-generated.
[2]No authorized party can obtain and re-generate the cardholder's secret signature key.

Table 3
Comparison of agent-based secure payment protocols (part II)

	TTP involved	Signature-Share scheme	Signcryption-Share scheme	PI Encryption
SET/A	no	/	/	
SET/A+	yes, TVC	/	/	symmetric key
LITESET/A+	yes	adopted	not adopted	symmetric key
LITESET/A++	yes	adopted	adopted	public key
proposed protocol	yes	adopted	not adopted	symmetric key

Regarding the protocol overhead, as analyzed in Section 5.1.4, based on Signcryption – a public key algorithm, the Signcryption-Share scheme is not as efficient as a symmetric key algorithm. In contrast, the proposed protocol adopts a symmetric key algorithm to encrypt *PI*. This inherits the efficiency property of SET and reduces the overhead at the side of the cardholder. In addition, the Signature-Share scheme is as efficient as a process of signature verification, which is carried out by the *PG*, not the agent. The proposed protocol needs to employ a TTP and thus leads to some communication with the TTP. But this is essential to an agent based payment protocol and it is the same as SET/A+, LITESET/A+ and LITESET/A++. The proposed protocol does not occur extra overhead than other protocols.

Furthermore LITESET/A++ has the drawback without any mechanism for preventing overspending and overpayment attacks. The problem also exists in other protocols.

In the proposed protocol, *Amount*, which is the transaction amount that will be charged to the cardholder's account, is first passed to TTP, who checks it with *PriceLimit* preventing overspending. Meanwhile, *Amount* is included in the ciphertext generated by the TTP, which is passed to the PG where *Amount* and *Price* from M are compared. This prevents the overpayment attack.

Security properties of different protocols are compared in Table 4.

6. Conclusions

In a mobile computing environment the cardholder's role can be played by an agent, which is dispatched to the merchant's server with the relevant information to perform the necessary operations. Since the cardholder does not need to keep the network connection while the transaction is being completed, this solution contributes to lower cost, higher robustness and autonomy while the requirements for security become more critical.

Table 4
Security properties of agent-based secure payment protocols

	Non-repudiation property	Double payment detection	Overspending detection	Over payment detection
SET/A	weak	no	no	no
SET/A+	A replay attack can succeed	no	no	no
LITESET/A+	weak	no	no	no
LITESET/A++	good	yes	no	no
proposed protocol	good	yes	yes	yes

In this paper, we proposed an agent-assisted secure payment protocol adopting Signature-Share scheme and employing a Trusted Third Party (TTP). The dispatched agent can dynamically choose the merchant and sign on behalf of the cardholder in cooperation with the TTP without the possibility of disclosing any secret to the merchant and TTP. In the proposed protocol, the principle that each participant knows what is strictly necessary for his/her role is followed as in SET while the efficiency is improved. In addition, mechanisms have been devised for preventing and detecting double payment, overspending and overpayment attacks.

The proposed protocol can be applied to e-commerce environments, where an agent is employed to collect offers, negotiate with merchants, place an order, and make the payment. For instance, the cardholder/customer using a laptop or PDA with wireless network interface and Internet access can apply this protocol for transactions. This can help reduce the complexity of operations at the side of the cardholder. In addition, the long-term network connection constraint for interactions will not apply. With the development of wireless communication and agent technology, we envisage that the above scenario is becoming applicable in practice. Thus, we expect the proposed protocol can be integrated with an agent-based e-commerce system [21] to enhance the autonomy of transactions and bring more convenience to customers.

Appendix 1. Signcryption public-key algorithm

Signcryption public-key algorithm [26] is as follows:

p – a large prime (public to all)
q – a large prime factor of $p-1$ (public to all)
g – a random integer in $[1,\ldots,p-1]$ with order $q \bmod p$ (public to all)
H – a one-way hash function whose output has, say, at least 128 bits
KH – a keyed one-way hash function
(E, D) – the encryption and decryption algorithms

Assume that sender A has chosen a secret key x_A from $[1,\ldots,q]$, and made public her matching public key $y_A = g^{x_A} \bmod p$. Similarly, the recipient B's secret key is x_B and his matching public key is $y_B = g^{x_B} \bmod p$.

Signcryption by A (the Sender)

1. Pick z randomly from $[1,\ldots,q]$, and let $k = H({y_B}^z \bmod p)$. Split k into k_1 and k_2 of appropriate length.
2. $c = E_{k_1}\{m\}$

3. $r = KH_{k_2}\{m\}$
4. $s = z/(r + x_A)\ mod\ q$
5. $A \rightarrow B$: the signcrypted text (c, r, s)

The unsigncryption algorithm works by taking advantage of the property that $g^z\ mod\ p$ can be recovered from $r,\ s,\ g,\ p,\ y_A$ by B. On receiving (c, r, s) from A, B unsigncripts the ciphertext and verifies the signature as follows.

Unsigncryption by B (the Recipient)

1. Recover k from r, s, g, p, y_A and x_B,

$$k = H(y_A \cdot g^r)^{s \cdot x_B}\ mod\ p$$

2. Split k into k_1 and k_2.
3. $m = D_{k_1}\{c\}$
4. Accept m as a valid message originated from A only if $KH_{k_2}\{m\}$ is identical to r.

Appendix 2. The proof of Signature-Share scheme

The Signature-Share scheme [14] is based on Signcryption algorithm [26]. In this scheme, the sender A wants to send a message m to recipient B through t sharing parties, say $A_i\ (i = 1, \ldots, t)$. The signature key of A is shared by t parties. Each party generates the shared signature s_i on the hash value r of message m, and all shared signatures are sent to B. With all (r, s_i), B can verify the signature and hence check the data integrity of m.

1. Each participant owes a key pair. For example, participant A has a key pair $(y_A,\ x_A)$, where key $y_A = g^{x_A}\ (mod\ p)$.
2. To protect one participant's secret key, we assume several parties share it using the special case of shamir secret-sharing scheme $(t = w)$. For example, to protect A's secret key x_A, t parties have secret shares x_{A_1}, x_{A_2}, …, x_{A_t} respectively, where $x_A = x_{A_1} + x_{A_2} + \ldots + x_{A_t}$.

The description of Signature-Share scheme:

Sender A: secret key x_A, chosen uniformly at random from $[1, \ldots, q]$, public key $y_A\ (y_A = g^{x_A}\ mod\ p)$. To protect one participant's secret key, we assume t sharing-parties share it using the special case of shamir secret-sharing scheme $(t = w)$. Namely, t parties have shared secret keys $x_{A_1}, x_{A_2}, \ldots, x_{A_t}$ respectively, where A's secret key $x_A = x_{A_1} + x_{A_2} + \ldots + x_{A_t}$.
Recipient B: secret key x_B, chosen uniformly at random from $[1, \ldots, q]$, public key

$$y_B\ (y_B = g^{x_B}\ mod\ p)$$

(i) A generates r and sends $(m,\ r,\ x_{A_i})$ to each sharing-party A_i. Here for simplicity, we suppose x_{A_i} is sent to A_i through a secure channel.

$$r = H(g^z\ mod\ p, m) \tag{1}$$

where z is a random number chosen from $[1, \ldots, q]$, and it is the same for each sharing- party.

(ii) For each A_i, once receiving $(m,\ r,\ x_{A_i})$, it generates s_i

$$s_i = z/(r + x_{A_i})\ mod\ q \quad 1 \leqslant i \leqslant t \tag{2}$$

Then A_i sends $(m,\ r,\ s_i)$ to B.

(iii) When receiving all (m, r, s_i), B verifies the signature.

$$v = H((y_A \cdot g^{tr})^{(\sum_{i=1}^{t} {s_i}^{-1})^{-1}}\ mod\ p) \tag{3}$$

$$\textit{accept}\ \ m\ \ \textit{only if}\ H(v,\ m) = r \tag{4}$$

Theorem 1. If (r, s_i) is computed as Eqs (1) and (2), (3) and (4) hold.
Proof: The key point is to prove that

$$g^z\ mod\ p = (y_A \cdot g^{tr})^{\sum_{i=1}^{t} s_i}\ mod\ p$$

From Eqs (1) and (2), we can deduce from the right of above.

$$\begin{aligned}
\textit{right} &= (y_A \cdot g^{tr})^{\sum_{i=1}^{t} s_i}\ mod\ p = (g^{x_A} \cdot g^{tr})^{\sum_{i=1}^{t} s_i}\ mod\ p \\
&= g^{(x_A+tr)(\frac{x_{A_1}+r}{z}+\frac{x_{A_2}+r}{z}+\ldots+\frac{x_{A_t}+r}{z})^{-1}}\ mod\ p \\
&= g^{(x_A+tr)\frac{z}{(x_{A_1}+x_{A_2}+\ldots+x_{A_t})+tr}}\ mod\ p \\
&= g^{(x_A+tr)\frac{z}{(x_A+tr)}}\ mod\ p \\
&= g^z\ mod\ p = \textit{left}
\end{aligned}$$

After getting the above deduction, Eq. (4) holds. Then the message can be verified according to Eqs (1) and (2).

References

[1] Amason. http://www.amason.com.
[2] eBay. http://www.eBay.com/.
[3] Visa International and MasterCard International. Secure Electronic Transaction (SET) specification, Version 1.0, May 1997.
[4] R.M.A. Corradi and C. Stefanell, *Mobile agent integrity in e-commerce application.* In Proceedings of 19th IEEE International Conference on Distributed Computing Systems, 1999, 59–64.
[5] J. Cheng and V. Wei, *Defenses against the truncation of computation results of free-roaming agent.* In Proceedings of Fourth Oniternational Conference on Information and Communication Security, pages 1–12, 2002. LNCS 2513, Springer-Verlag.
[6] D. Chess, *Security issues in mobile code systems.* In Proceedings of Mobile Agents and Security, pages 1–14, 1998. LNCS 1419, Springer-Verlag.
[7] R. Guttman and P. Maes, *Agent-mediated integrative negotiation for retail electronic commerce.* In Proceedings of the Workshop on Agent Mediated Electronic Trading (AMET'98), pages 1–13, 1998.
[8] F. Ishikawa, N. Yoshioka, Y. Tahara and S. Honiden, *Behavior descriptions of mobile agents for web services integration.* In ICWS, pages 342–349, 2004.
[9] F. Ishikawa, N. Yoshioka, Y. Tahara and S. Honiden, *Mobile agent system for web services integration in pervasive networks.* In IWUC, pages 38–47, 2004.

[10] B.P.J. Claessens and J. Vandewalle, *Can mobile agents do secure electronic transactions on untrusted hosts? A survey of the security issues and the current solutions.* In ACM Transactions on Internet Technology, volume 3, pages 28–48, February 2003.
[11] P. Maes, R. Guttman and A. Moukas, Agents that buy and sell, *CACM* **42**(3) (1999), 81–91.
[12] A. Menezes, P. Oorschot and S. Vanstone, *Handbook of Applied Cryptography,* CRC Press, 1996.
[13] M.B.P. Kotzanikolaou and V. Chrissikopoulos, *Secure transactions with mobile agents in hostile environments.* In ACISP2000, pages 289–297, 2000. LNCS 1841, Springer-Verlag.
[14] X. Pang, K.-L. Tan and Y. Wang, *A secure agent-mediated payment protocol.* In Fourth International Conference on Information and Communications Security (ICICS2002), volume LNCS 2512, Springer-Verlag, pages 422–433, Singapore, December 2002. Springer-Verlag.
[15] T.D. Rodrigo and A. Stanski, The evolving future of agent-based electronic commerce, in: *Electronic Commerce: Opportunity and Challenges*, S.M. Rahman and M.S. Raisinghani, eds, 2000, pp. 337–351.
[16] A. Romao and M.M. da Silva, *An agent-based secure internet payment system for mobile computing.* In Proceedings of TrEC'98, Hamburg, Germany, Springer, 1998.
[17] J.G.S. Berkovits and V. Swarup, *Authentication for mobile agents.* In Proceedings of Mobile Agents and Security, pages 114–136, 1998. LNCS 1419, Springer-Verlag.
[18] T. Sander and C. Tschudin, Technical Report TR-97- 049, International Computer Science Institute, November 1997.
[19] V. Varadharajan, *Security enhanced mobile agents.* In Proceedings of the 7th ACM conference on Computer and Communications Security (CCS'00), pages 200–209, Athens, Greece, November 1–4, 2000.
[20] Y. Wang and T. Li, *LITESET/A++: A new agent-assisted secure payment protocol.* In Proceedings of 6th IEEE International Conference on E-Commerce Technology (IEEE CEC'04), San Diego, California, USA, July 2004. IEEE Computer Society.
[21] Y. Wang, K.-L. Tan and J. Ren, Pumamart: A parallel and autonomous agents based internet marketplace, *Electronic Commerce Research and Applications* **3**(3) (2004), 294–310.
[22] Y. Wang, K.-L. Tan and J. Ren, Towards autonomous and automatic evaluation and negotiation in agent-mediated internet marketplaces, *Electronic Commerce Research* **5** (2005), 343–365.
[23] U. Whilem and X. Defago, *Objets protgs cryptographiquement.* In Proceedings of RenPar'97, Lausanne, Switzerland.
[24] B. Yee, *A sanctuary for mobile agents.* In Proceedings of the DARPA Workshop on Foundations for Secure Mobile Code, Monterey CA, USA, March 1997.
[25] X. Yi, C.K. Siew, X.F. Wang and E. Okamoto, A secure agent-based framework for the internet trading in mobile computing environments, *Distributed and Parallel Databases* **8** (2000), 85–117.
[26] Y. Zheng, *Digital signcryption or how to achieve cost* (*signature and encryption*) $\ll$ *cost* (*signature*) + *cost* (*encryption*). In Proceedings of Advances in Cryptology-CRYPO'97, volume 1294, pages 165–179. Springer-Verlag, 1997.
[27] J. Zhou and K. Lam, Securing digital signatures for non-repudiation, *Computer Communications* **22** (1999), 710–716.

Dr. Yan Wang is currently a Senior Lecturer in the Department of Computing, Macquarie University, Sydney, Australia. He received his B. Eng, M. Eng and Doctorate Degree of Engineering in computer science and technology in 1988, 1991 and 1996 respectively from Harbin Institute of Technology, China. He was a Senior Research Assistant/Associate in the Department of Computer Science, City University of Hong Kong in 1997 and 1999 respectively. Prior to joining Macquarie University in 2003, he was a Postdoctoral Fellow/Research Fellow in the Department of Computer Science, School of Computing, National University of Singapore from Nov. 1999 to June 2003. His research interests cover trust computing, computer security, and electronic commerce.

Dr. Yan Wang is the editor of International Journal of Web Engineering and Technology (IJWET) and the guest co-editor of a special issue of IEEE Internet Computing.

Duncan S. Wong received the B.Eng. degree from the University of Hong Kong in 1994, the M.Phil. degree from the Chinese University of Hong Kong in 1998, and the Ph.D. degree from Northeastern University, Boston, MA, USA in 2002. He is currently an Assistant Professor in the Department of Computer Science at the City University of Hong Kong. His primary research interest is applied cryptography; in particular, cryptographic protocols and security for wireless communications. In addition, he has many years of experience in developing practical cryptographic systems.

He has published more than 50 papers in international journals and conferences. He has been the co-editor of proceedings of "Information Security Practice and Experience Conference (ISPEC)" (LNCS Springer-Verlag, 2007) and editors of International Journal of Application Cryptography (IJACT) and a special issue on cryptography in computer system security, Journal of Universal Computer Science (JUCS), 2007.

Huaxiong Wang is currently an Associate Professor in Division of Mathematical Sciences, School of Physical and Mathematical Sciences, Nanyang Technological University, Singapore. Prior to taking up the position at NTU in July 2006, he was a Senior

Lecturer in the Department of Computing, Macquarie University, Australia. Prior to that, he held faculty and research positions in several universities such as, Senior Research Fellow at City University of Hong Kong (August 2005 – January 2006), Lecturer at University of Wollongong (2001), Research Fellow at National University of Singapore (1999), Visiting Fellow at Kobe University, Japan (1991–1992), and Associate Lecturer/Lecturer at Fujian Normal University, China (1984–1991). He received a PhD in Mathematics from University of Haifa, Israel in 1996 and a PhD in Computing Science from University in Wollongong, Australia in 2001.

He has published more than 90 papers in the international journals and conferences, such journals as IEEE Trans. Information Theory, Journal of Combinatorial Theory, Information and Computation, Theoretical Computer Science, Journal of Complexity, Computer Networks, Designs, Codes and Cryptography, and such conferences as Crypto, Eurocrypt, Asiacrypt and ACM Conference on Computer and Communication Security, etc. He is the co-editor of three conference proceedings: "Cryptology and Network Security" (LNCS, Springer-Verlag, 2005), "Information Security and Privacy" (LNCS, Springer-Verlag 2004) and "Cryptography and Computational Number Theory" (Birkhauser, 2001). He is also the co-author of 2 book chapters. He won the Inaugural Award for Best Research Contribution, the Computer Science Association of Australasia in 2004.