LITESET/A++: A New Agent-assisted Secure Payment Protocol

Yan Wang
Department of Computing
Macquarie University
Sydney, NSW 2109
Australia
yanwang@ics.mq.edu.au

Tieyan Li
Infocomm Security Department
Institute for Infocomm Research
21 Heng Mui Keng Terrace
Singapore 119613
litieyan@i2r.a-star.edu.sg

Abstract

In this paper, we proposed a new agent-assisted secure payment protocol LITESET/A++, which aims at enabling the dispatched consumer-agent to choose the "best" one after negotiating with merchants to close a deal, autonomously sign contracts and make the payment on behalf of the cardholder without the possibility of disclosing any secret to any participant. This is realized by adopting the Signature-Share and Signcryption-Share schemes, and employing a Trusted Third Party (TTP). In LITESET/A++, the principle that each participant knows what is strictly necessary for his/her role is followed as in SET payment protocol while the non-repudiation property is improved.

1. Introduction

Autonomous agents, stationary or mobile, offer new paradigms with autonomy, intelligence and flexibility. Autonomous agent based e-commerce has increasingly drawn attentions from both research community [1,2,3,4,5] and applications (e.g., Amazon [6] and eBay [7]). In our real life, people can turn to a few agents or agencies for buying air tickets and notebooks, renting or buying a house or even shopping for groceries. They can choose a satisfactory one from multiple provided plans. Similarly, 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: looking for and filtering online shops selling the specified products, asking offers, negotiating with shops and even completing transactions [3,8].

On the other hand, the introduction of mobile agents increases the risk in terms of security [9,10,11], particularly when they carry critical/confidential information (e.g. credit card information), sign contracts or make payment on behalf of the consumers since the agents and their carried sensitive data will be exposed to potentially hostile environments.

SET (Secure Electronic Transaction) protocol developed by VISA and MasterCard is regarded as a good protocol [13] aiming at protecting users' credit card information with important properties, such as authentication of the participants, data integrity and confidentiality.

By using a new cryptography technique Signcryption [14], LITESET (Light-Weight Secure Electronic Transaction) protocol [15] reduces the heavy computation and message overhead in the employment of SET, the implementation of which is based on traditional RSA signature and encryption scheme [16]. Several agent-based extensions of the SET protocol have been proposed, such as the SET/A [12], SET/A+ [18] and LITESET/A+[19], aiming at utilizing the autonomy of a mobile payment agent while ensuring the security of payments. But in SET/A+, a pre-generated signature is carried by the agent that can be abused by any merchant. In LITESET/A+, some critical arguments are exposed to the merchant who can re-generate the cardholder's secret signature key.

In this paper, we proposed a new agent-assisted secure payment protocol LITESET/A++. The goal of LITESET/A++, which is based on SET, is to enable a mobile agent to automatically and autonomously make final transactions and payment with the "best" merchant without interacting with the-consumer after performing all kinds of tasks including looking for offers, and negotiating with merchants. This requires the capability of the agent in the protocol to dynamically sign with the "best" merchant, who cannot be determined in advance, and pass the payment information to the payment gateway (*PG*), which can be determined only after the interaction with the "best" merchant according to the brand of the credit card. Hence encrypting everything in advance is not possible while asking the agent to carry any key for encryption is certainly a risk. In LITESET/A++, by adopting the Signature-Share and Signcryption-Share schemes, the agent can sign contracts and pass the payment information to the *PG* in corporation with the Trusted Third Party (TTP) without the possibility of disclosing any secret to any participants.

The rest of this paper is organized as follows. Section 2 briefly reviews SET, the Signcryption-Share scheme and Signature-Share scheme. LITESET/A++ is presented in Section 3 and its features and security properties are analyzed in Section 4. Section 5 finally concludes our work.

2. Background

In this section, we will first review SET, Signature-Share scheme and Signcryption-Share scheme. The notations and symbols used in this paper are listed as follows.

(Ky_{PG}, Kx_{PG})	a pair of temporally generated session keys (*Public Key, Secret Key*) for payment gateway *PG*
C	card holder
CA	consumer agent
$C_K(A)$	key-exchange certificate of participant *A*
$c_{->A}$	the ciphertext that should be *passed to* participant *A*
$C_S(A)$	signature certificate of participant *A*
$E_k\{m\}$	the ciphertext of message *m* encrypted by key *k*
$E_{PG}\{K, PI\}$	the digital envelop generated by *PG* (= $\{E_{yKPG}\{K\},E_K\{PI\}\}$), *K* is a symmetric key
g	a (random) integer in [*1, ..., p-1*] with order *q mod p* (public to all)
$H(m)$	a one-way hash function applied to message *m*
I_A	the unique transaction number issued by participant *A*
KH	a keyed one-way hash function
M	merchant
p	a large prime (public to all)
OI	order information
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	a random number chosen from [*1, ..., q-1*]
$r_{->A}$	the hash value *r* that should be *passed to* participant *A*
SIG_A	the signature *generated by* participant *A*
$s_{->A}\text{-}i$	the *i*th shared signature that should be *passed to* participant *A*
T_e	the timestamp when the purchase request expires
T_A	the timestamp at participant *A*
(y_{K_A}, x_{K_A})	(public key, secret key) of participant *A* for encryption /decryption
(y_{S_A}, x_{S_A})	signature (public key, secret key) of participant *A*
z	a random number chosen from [*1, ..., q*]
$X\|\|Y$	concatenation of two messages *X* and *Y*
$A\text{->}B{:}m$	participant *A* sends a message *m* to participant *B*

2.1 SET

We will first review SET in this section. The SET protocol [13] is composed of several kinds of transactions, ranging from certification of participants, to purchase requests and to payment processing. It uses two distinct asymmetric key pairs, 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 public signature key of participant *A* is included in the signature certificate $C_S(A)$.

In this paper we are particularly interested in the *purchase request* phase, which can be outlined as follows (see Figure 1):

Step 1. When a consumer, called a *cardholder* (*C*), decides to purchase something from a merchant (*M*), he/she sends a request to the merchant's server. The request includes the description of the services or the quantities of the goods, the terms of the order, and the brand of the credit card that will be used for payment.

Step 2. Upon receiving the request, the merchant sends back its own signature certificate C_S *(M),* and the key-exchange certificate C_K *(PG)* of a *payment gateway* (*PG*). The payment gateway is a device operated by a financial institution with which the merchant established an account for processing payments with the brand used by the cardholder. The merchant also sends a unique identifier, assigned to this transaction.

Step 3. The cardholder verifies the certificates by contacting a *certificate authority* (*CA*) He/she receives back a confirmation that assures the authenticity and integrity of the data (the merchant had digitally signed it), and creates two pieces of information:

- The *Order Information* (*OI*), containing control information verified by the merchant to validate the order, card brand and bank identification.

The *OI* also includes *a digest of the order descriptio*n, which includes the amount of the transaction and other elements such as quantity, size and price of the items ordered, shipping and billing addresses, etc.

- The *Payment Instructions* (*PI*), containing the amount of the transaction, the card account number and expiration date, instructions for instalment payments (if that's the case) and a couple of secret values to prevent guessing and dictionary attacks on the data, among other elements. The *PI* is encrypted with a randomly generated symmetric key, *K*.
  Both elements will contain the transaction identifier and are dually signed, so they can later be linked together by the payment gateway. Then, the encrypted *PI* (i.e. $E_K(PI)$), and the key (*K*) used to encrypt it are encrypted into a *digital envelope* (E_{PG}), using the payment gateway's public key. Finally, the *OI* and the digital envelope are sent to the merchant, along with the cardholder's signature certificate $C_S\,(C)$.

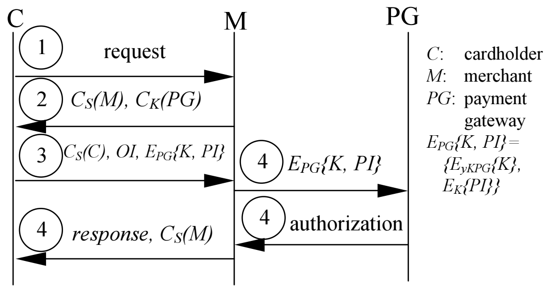

Figure 1. **SET purchase request transaction**

Step 4. The merchant verifies the cardholder certificate and the dual signature on the *OI*. The request is then processed, which includes forwarding the digital envelope to the payment gateway, for authorization (the details of this operation are outside the scope of this description). After processing the order, the merchant generates and signs a purchase response, and sends it to the cardholder along with its signature certificate. If the payment was authorized, the merchant will fulfill the order, by delivering the products bought by the cardholder.

Step 5. The cardholder verifies the merchant signature certificate, checks the digital signature of the response, and takes any appropriate actions based on its contents.

SET provides important properties, such as authentication of participants, data integrity and confidentiality. Each participant knows what is strictly necessary for his/her role.

SET/A, SET/A+ and LITESET/A++ are existing agent-based secure payment protocols. We will compare them with LITESET/A++ in section 4.

2.2 Signature-Share Scheme and Signcryption-Share Scheme

In this section, we will briefly preview the Signature-Share scheme and Singcryption-Share Scheme proposed in [19].

2.2.1 Signcryption Scheme. Signcryption public-key scheme [14,15] is as follows:

p is a large prime (public to all)
q is a large prime factor of *p-1* (public to all)
g is a (random) integer in [*1, ..., p-1*] with order *q mod 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, which is for secure message authentication
(*E, D*)- the encryption and decryption algorithms

Assume that the sender *A* has chosen a secret key x_A from [*l, ... , 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 [*l, ..., q*], and let
   $k = H(y_B^z \bmod p)$
   Split *k* into k_1 and k_2 of appropriate length.
2. $c = E_{k1}(m)$
3. $r = KH_{k2}(m)$
4. $s = z/(r + x_A) \bmod q$
5. A->B: the signcrypted text *(c, r, s)*

The unsigncryption algorithm works by taking advantage of the property that $g^z \bmod p$ 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
   $k = H((y_A \bullet g^r)^{s \bullet x_B} \bmod p)$
2. Split *k* into k_1 and k_2.
3. $m = D_{k1}(c)$
4. Accept *m* as a valid message originated from Alice only if $KH_{k2}(m)$ is identical to *r*.

2.2.2 Signature-Share Scheme. The Signature-Share scheme [19] is based on signcryption. In this scheme, the sender A wants to send a message m to recipient B through t sharing parties, say A_i ($i=1...t$). The signature key of A is shared by t parties, namely, $x_A = x_{A_1} + x_{A_2} + ... + x_{A_t}$. 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.

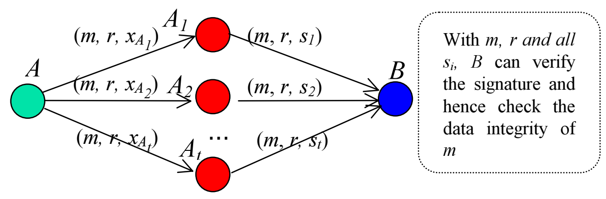
A
A1
A2
At
B
(m, r, xA1)
(m, r, xA2)
(m, r, xAt)
(m, r, s1)
(m, r, s2)
(m, r, st)
...
With m, r and all si, B can verify the signature and hence check the data integrity of m

Figure 2. Signature-Share scheme

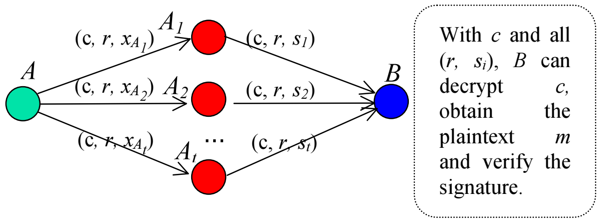
A
A1
A2
At
B
(c, r, xA1)
(c, r, xA2)
(c, r, xAt)
(c, r, s1)
(c, r, s2)
(c, r, st)
...
With c and all (r, si), B can decrypt c, obtain the plaintext m and verify the signature.

Figure 3. Signcryption-Share scheme

2.2.3 Signcryption-Share Scheme. Signcryption-Share scheme is based on signcryption too. In this scheme, the sender A wants to send a message m to recipient B through t sharing parties, say A_i ($i=1...t$). The secret signature key of A is shared by t parties, $x_A = x_{A_1} + x_{A_2} + ... + x_{A_t}$. Each party generates the shared signature s_i on hash value r obtained from A, and all the shared signatures are sent to B with the ciphertext c. With c and all (r, s_i), B can decrypt c, obtain the plaintext m and verify the signature.

3. LITESET/A++ Protocol

Based on the Signature-Share scheme and Signcryption-Share scheme, we will present the new secure payment protocol LITESET/A++. Once the agent is authorized to buy certain kind of product, all further activities such as appropriate decision of buying, negotiation and signing for certain action, will be performed without the cardholder's assistance. In this paper, we will focus on the purchase request phase only.

In LITESET/A++, Signature-Share scheme is adopted for passing securely the information on order information to the merchant while Signcryption-Share scheme is adopted for passing the payment information to the payment gateway and a temporarily session public key pair is used to encrypt the payment information *PI*. The dispatched agent does not carry its shared private signature key. Instead it only carries two half shared signatures signed on the order information (*OI*) and payment information (*PI*) respectively by the cardholder that should be sent to the merchant and payment gateway accordingly. The shared signature key is kept by the cardholder. The other 2 half signatures are generated with the assistance of TTP. The merchant can verify the order information (*OI*) and check the data integrity. Meanwhile the payment gateway (*PG*) can not only decrypt the payment information *PI* but also check the data integrity after obtaining the shared signatures from the agent and TTP respectively. This is better than using a symmetric key only as in SET, SET/A, SET/A+ and LITESET/A+. Additionally, non-repudiation property of LITESET/A++ is significantly improved.

3.1 Secret-Sharing of Cardholder's Signature Private Key x_{S_C}

The consumer agent *CA* and TTP share the cardholder's signature private key x_{S_C} based on shamir-threshold scheme [30].

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

Namely, according to the two share schemes presented in Section 2.2 and 2.3, $A_1=CA$, $A_2=$TTP.

3.2 Description of LITESET/A++: Secure Agent-assisted Payment Protocol

Step 1: Cardholder C generates a pair of temporary session keys -(Ky_{PG} , Kx_{PG}), where $Ky_{PG}=g^{Kx_{PG}} \bmod p$- for the payment gateway. It is different from PG's public encryption key pair ($y_{K_{PG}}$, $x_{K_{PG}}$).

1) Then C uses signcryption algorithm to encrypt the payment information (*PI*):

$(k_1, k_2)=H(K_{y_{PG}}{}^z \bmod p)$

$c_{->PG}=E_{k1}(PI)$

$r_{->PG}=KH_{k2}(PI)$

$s_{->PG}\text{-}1=z/(r_{->PG}+x_{S_{CA}}) \bmod q$

and ciphertext $E_{yK_{TTP}}(x_{TTP}||z||(Kx_{PG}+R+I_C+T_C+T_e))$;

- R is a random number chosen from $[1, ..., q\text{-}1]$;
- I_C is the transaction identifier assigned by cardholder;
- and T_C is timestamp at C;
- and T_e ($T_e > T_C$) is the time when the purchase request expires. It is unique to each purchase order.

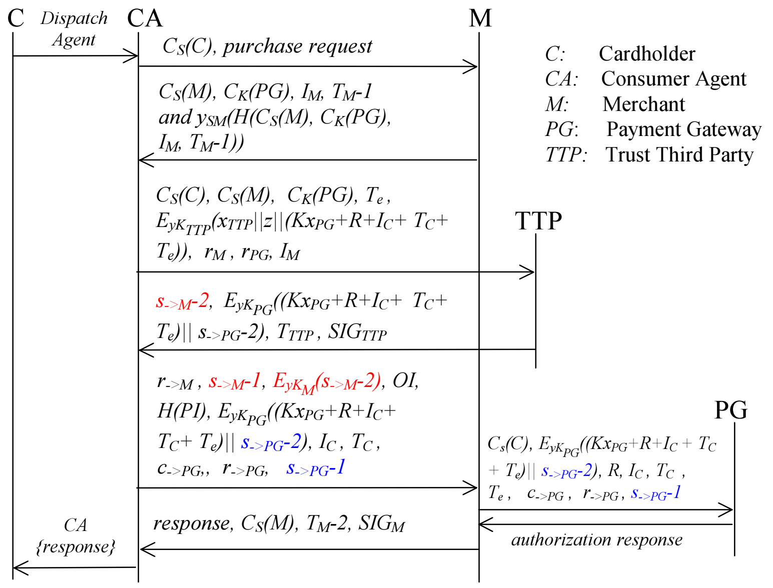
C
Dispatch Agent
CA
M
TTP
PG
$C_S(C)$, purchase request
$C_S(M)$, $C_K(PG)$, I_M, T_M-1 and $y_{SM}(H(C_S(M), C_K(PG), I_M, T_M\text{-}1))$
$C_S(C)$, $C_S(M)$, $C_K(PG)$, T_e, $E_{yK_{TTP}}(x_{TTP}||z||(Kx_{PG}+R+I_C+T_C+T_e))$, r_M, r_{PG}, I_M
$s_{\text{->}M}$-2, $E_{yK_{PG}}((Kx_{PG}+R+I_C+T_C+T_e)||s_{\text{->}PG}\text{-}2)$, T_{TTP}, SIG_{TTP}
$r_{\text{->}M}$, $s_{\text{->}M}$-1, $E_{yK_M}(s_{\text{->}M}\text{-}2)$, OI, H(PI), $E_{yK_{PG}}((Kx_{PG}+R+I_C+T_C+T_e)||s_{\text{->}PG}\text{-}2)$, I_C, T_C, $c_{\text{->}PG}$, $r_{\text{->}PG}$, $s_{\text{->}PG}$-1
$C_s(C)$, $E_{yK_{PG}}((Kx_{PG}+R+I_C+T_C+T_e)||s_{\text{->}PG}\text{-}2)$, R, I_C, T_C, T_e, $c_{\text{->}PG}$, $r_{\text{->}PG}$, $s_{\text{->}PG}$-1
authorization response
response, $C_S(M)$, T_M-2, SIG_M
CA {response}
C: Cardholder
CA: Consumer Agent
M: Merchant
PG: Payment Gateway
TTP: Trust Third Party

Figure 4. LITESET/A++ purchase request transaction

(Note: $s_{\text{->}PG}\text{-}1$ is the half shared signature generated by C that should be passed to payment gateway PG and the consumer agent carries it instead of the shared secret key- $x_{S_{CA}}$. $x_{S_{CA}}$ is kept by C.)

2) Meanwhile, C generates a half shared signature $s_{\text{->}M}\text{-}1$ on the dual hash value

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

$s_{\text{->}M}\text{-}1=z/(r_{\text{->}M}+x_{S_{CA}}) \bmod q$

3) Then C dispatches the consumer agent CA encapsulating the following arguments

$C_S(C)$, $E_{yK_{TTP}}(x_{TTP}||z||(\ Kx_{PG}+R+I_C+T_C+\ T_e))$, $r_{\text{->}M}$, $s_{\text{->}M}\text{-}1$, OI, $H(PI)$, I_C, T_C, T_e, R, $c_{\text{->}PG}$, $r_{\text{->}PG}$, $s_{\text{->}PG}\text{-}1$

Step 2: After completing the negotiation with merchants, the agent will choose the best one M to make the deal.

CA->M: $C_S(C)$, *purchase request*, T_e

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

M->CA: $C_S(M)$, $C_K(M)$, $C_K(PG)$, I_M, $T_M\text{-}1$ *and* $y_{S_M}(H(C_S(M), C_K(PG), I_M, T_M\text{-}1))$

where I_M is a unique transaction number issued by M and $T_M\text{-}1$ is the current timestamp *at M*

Step 4: CA will send TTP a message so that $s_{\text{->}PG}\text{-}2$ and $s_{\text{->}M}\text{-}2$ can be generated by TTP

CA->TTP: $C_S(C)$, $C_S(M)$, $C_K(M)$, $C_K(PG)$, T_e, $E_{yK_{TTP}}(x_{TTP}||z||(Kx_{PG}+R+I_C+\ T_C+\ T_e))$, $r_{\text{->}M}$, $r_{\text{->}PG}$, I_M

Step 5: When receiving the message, TTP verifies the validation of $C_S(C)$, $C_S(M)$, $C_K(M)$, *and* $C_K(PG)$, checks if current time $T<T_e$, decrypts the ciphertext and generates 2 half shared signatures on hash values of $r_{\text{->}PG}$ and $r_{\text{->}M}$ respectively.

$s_{\text{->}PG}\text{-}2=z/(r_{\text{->}PG}+x_{TTP}) \bmod q$

$s_{\text{->}M}\text{-}2=z/(r_{\text{->}M}+x_{TTP}) \bmod q$, and computes $E_{yK_{PG}}((Kx_{PG}+R+I_C+T_C+T_e)||s_{\text{->}PG}\text{-}2)$ and $E_{yK_M}(s_{\text{->}M}\text{-}2)$.

(note: TTP knows $(Kx_{PG}+R+I_C+\ T_C+\ T_e)$ but doesn't know Kx_{PG})

TTP keeps $RCT_1=(C_S(C), C_S(M), I_C, T_C, T_e, I_M, T_M\text{-}1, T_{TTP}, y_{S_M}(H(C_S(M), C_k(PG), I_M, T_M\text{-}1)))$ as a non-repudiation receipt and sends a message to CA

TTP->CA: $E_{yK_M}(s_{\text{->}M}\text{-}2)$, $E_{yK_{PG}}((Kx_{PG}+R+I_C+T_C+\ T_e)||\ s_{\text{->}PG}\text{-}2)$, T_{TTP}, SIG_{TTP}

where T_{TTP} is the timestamp at TTP, $SIG_{TTP}= x_{S_{TTP}}(H(C_S(C), C_S(M), C_K(PG), E_{yK_M}(s_{\text{->}M}\text{-}2), E_{yK_{PG}}((Kx_{PG}+R+I_C+\ T_C+\ T_e)||s_{\text{->}PG}\text{-}2), r_{\text{->}M}, r_{\text{->}PG}, I_M, T_{TTP}))$ is the signature generated by TTP at time T_{TTP}.

Step 6: Once receiving the message from TTP, *CA* will send a message to the merchant.
CA->M: $r_{->M}$, $s_{->M}-1$, $E_{yK_M}(s_{->M}-2)$, OI, $H(PI)$, $E_{yK_{PG}}((Kx_{PG}+R+I_C+T_C+T_e)||s_{->PG}-2)$, I_C, T_C, T_e, $c_{->PG}$, $r_{->PG}$, $s_{->PG}-1$

Step 7: When receiving the message, *M* will verify the signature

$$v = H((y_C \cdot g^{2r})^{(\sum_{i=1}^{2} s_M - i^{-1})^{-1}} \bmod p)$$

$H(v, H(PI)||H(OI)||H(C_S(C)||I_C||T_C||Te)) = r_{->M}$
If it holds and current time $T<T_e$, *M* keeps $RCT_2=(r_{->M}, s_{->M}-1, s_{->M}-2, OI, H(PI), I_C, T_C, T_e)$ as a receipt. Then *M* sends a message to *PG*.
M->PG: $C_S(C)$, $E_{yK_{PG}}((Kx_{PG}+R+I_C+T_C+T_e)||s_{->PG}-2)$, R, I_C, T_C, T_e, $c_{->PG}$, $r_{->PG}$, $s_{->PG}-1$

Step 8: From the message, *PG* obtains $s_{->PG}-1$. After decrypting $E_{yK_{PG}}((Kx_{PG}+R+I_C+T_C+T_e)||s_{->PG}-2)$, it obtains Kx_{PG} and $s_{->PG}-2$. Hereafter *PG* can decrypt $c_{->PG}$ and thus obtains *PI*

$$(k_1, k_2) =$$

$$H((y_C \cdot g^{2r})^{(\sum_{i=1}^{2} s_{->PG} - i^{-1})^{-1} \cdot Kx_{PG}} \bmod p)$$

$$PI = D_{k_1}(c_{->PG})$$

and check data integrity:

$$KH_{k_2}(PI) \overset{?}{=} r_{->PG}$$

If all are correct and current time $T<T_e$, *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 along with its signature certificate.
M->CA: $C_S(M)$, T_M-2, SIG_M
where T_M-2 is the timestamp ($T_M-2 > T_M-1$) at *M* when the SIG_M is issued; $SIG_M = x_{S_M}(H(C_S(M), r_{->M}, s_{->M}-1, s_{->M}-2, OI, H(PI), I_C, T_e, I_M, T_M-2))$ is the signature generated by *M* at time T_M-2.
If the payment is authorized, the merchant will fulfill the order by delivering the products 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 $C_S(M)$, $C_S(TTP)$, $C_K(PG)$, T_{TTP}, T_M-1, T_M-2, SIG_{TTP}, SIG_M. The owner takes any appropriate actions based on its contents.

4. Security Analysis

In this section, we will analyse the security properties of LITESET/A++ focusing on the following possible issues.

- if it is possible for any participant to re-generate the secret signature key of the cardholder;
- if it is possible for any participant to re-perform the payment (replay attack);
- if it is possible for any participant except *PG* to obtain the payment information;
- if the non-repudiation property is improved.

(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 keys.

In LITESET/A++, the agent in the transaction period is more of a messenger. Most of the encryption and signing work are done by the TTP. What the agent should do is to communicate with different participants sending relevant messages to them.

(2) *CA* carries shared signatures - $s_{->M}-1$ and $s_{->PG}-1$. But they are generated by cardholder *C* and the shared secret key $x_{S_{CA}}$ is kept by *C*. Any party could not obtain both 2 shared signatures (i.e. $s_{->M}-1$ and $s_{->M}-2$, or $s_{->PG}-1$ and $s_{->PG}-2$) together with some argument (i.e. r and z), so it is not possible for any party to get 2 shared secret keys so as to generate the secret signature key of the cardholder (i.e. x_C.).

For instance, for the merchant, it can obtain the $r_{->M}$, $s_{->M}-1$, $s_{->M}-2$, $c_{->PG}$,, $r_{->PG}$, $s_{->PG}-1$ *and* $H(PI)$, but cannot obtain *PI* and $s_{->PG}-2$. Argument z is also protected against the merchant. So it is not possible for *M* to obtain x_C.

Likewise, TTP knows $(Kx_{PG}+R+I_C+T_C+T_e)$ but doesn't know Kx_{PG}. Meanwhile $E_{k1}(PI)$ is not passed to TTP. As $s_{->M}-1$ and $s_{->PG}-1$ are not passed to TTP, TTP cannot re-generate x_C.

In LITESET/A++, the cardholder's secret signature key can be re-generated only if *M* and TTP collude. In this case, the merchant can re-perform the payment. But it is impossible regarding the nature of TTP. Even if they collude, both sides cannot obtain *PI* since only *PG* can obtain the key to decrypt $E_{k1}(PI)$. As I_C , T_C *and* T_e are included in $E_{yK_{PG}}((Kx_{PG}+R+I_C+T_C+T_e)$ and *M* cannot modify, *PG* can easily find the replay attack.

(3) After obtaining 2 shared signatures -$s_{->PG}-1$ *and* $s_{->PG}-2$- *PG* can not only decrypt the payment information *PI* but also check the data integrity. Its session secret key Kx_{PG} is encrypted as $E_{yK_{PG}}((Kx_{PG}+R+I_C+T_C)||s_{->PG}-2)$. As *M* cannot know $s_{->PG}-2$ and the random number *R* is added in

the ciphertext, this helps to 'mask' the ciphertext so that it is not possible for M to guess Kx_{PG} by trials of encrypting messages to obtain the same ciphertext.

(4) The property of non-repudiation is improved. In terms of non-repudiation, timestamps are important in many electronic transactions indicating the time that a particular event or action took place [23]. Beside T_C, more timestamps are added in different stages, such as T_e, T_{TTP}, T_M-$1$and T_M-2. 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 between CA, TTP and M. 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 (e.g. 128 bytes by MD5).

As we analyzed in section 1 and above, LITESET/A++ corrects the security flaw in LITESET/A+ so that it is not possible for the merchant to re-generate the private signature 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 LITESET/A++ does not need to do any encryption and decryption. In contrast, in SET/A+ [18] and LITESET/A+ [19], the agent executes at the merchant's server and completes encryption operations.

In LITESET/A++, both the Signature-Share scheme and Signcryption-Share scheme are used. For the first one, it is similar to LITESET/A+ [19]. The Signcryption-Share scheme is used to pass the session secret key to the payment gateway while no participant but PG can decrypt PI and check the data integrity. In the process, TTP not only generates a shred signature $s_{->PG}$-2, but also helps encrypt $E_{yK_{PG}}((Kx_{PG}+R+I_C+T_C+T_e)||s_{->PG}\text{-}2)$ without the possibility of knowing Kx_{PG}. Moreover, as the cardholder's signature is dynamically generated with the assistance of TTP, LITSSET/A++ avoids the flaw in SET/A+ [18] using a pre-generated signature that can be abused by any merchant causing the loss of the cardholder.

In addition, the non-repudiation properties in SET/A [12], SET/A+ [18] and LITESET/A+ [19] are all week.

5. Conclusions

In this paper, we proposed an agent-assisted secure payment protocol LITESET/A++ adopting Signature-Share and Signcryption-Share schemes and employing a Trusted Third Party. The dispatched agent can dynamically and flexibility chose the merchant and sign on behalf of the cardholder in corporation with the TTP without the possibility of disclosing any secret to the merchant and TTP. In LITESET/A++, the principle that each participant knows what is strictly necessary for his/her role is followed as in SET while the non-repudiation property is improved. In comparison with other agent-based secure payment protocols, it can prevent the replay attack and has improved the non-repudiation property.

For future work, we would like to integrate the LITESET/A++ protocol into our existing framework, PumaMart - A Parallel and Autonomous Agents based Internet Marketplace [25, 26, 31] implemented on top of JDK [27] and ASDK [28, 9] where agents are employed for shop searching/filtering, offer/searching/filtering and negotiating on behalf of the consumer.

6. Acknowledgement

This work is supported by the start-up grant of ICS division of Macquarie University (9600/0004).

7. References

[1] D. Chess, C. Harrison, and A. Kershenbaum, "Mobile Agents: Are They a Good Idea," Technical Report, IBM T.J. Watson Research Center, NY, March 1995.

[2] D. Lange and M. Oshima, "Seven Good Reasons for Mobile Agents", *CACM,* Vol 42, No.3, 1999, pp 88-89

[3] R.H. Guttman and P. Maes, "Agent-mediated integrative negotiation for retail electronic commerce", *Proceedings of Workshop on Agent Mediated Electronic Trading*, Minneapolis, Minnesota, USA, May 1998.

[4] P. Maes, R. Guttman and A. Moukas, "Agents That Buy and Sell", *CACM*, 42 (3), 1999, pp 81-91

[5] T. D. Rodrigo, and A. Stanski, "The Evolving Future of Agent-based Electronic Commerce", in *Electronic Commerce: Opportunity and Challenges* (Edited by S. M. Rahman and M. S. Raisinghani), Idea Group Publishing, Hershey, USA, 2000, pp. 337-351

[6] URL: http://www.amason.com

[7] URL: http://www.ebay.com

[8] R. Corradi, R. Montanari and C. Stefanell, "Mobile Agent Integrity in E-commerce Application", *Electronic Commerce and Web-based Applications/Middleware, 19th IEEE International Conference on Distributed Computing Systems*, 1999, pp 519-533

[9] S. Berkovits, J. Guttman, V. Swarup, "Authentication for Mobile Agents", in *Proceedings of Mobile Agents and Security*, Springer Verlag, LNCS 1419, 1998, pp.114-136.

[10] D. Chess, "Security Issues in Mobile Code Systems", in Proceedings of Mobile Agents and Security", Springer Verlag, LNCS 1419, 1998, pp.1-14.

[11] P. Kotzanikolaou, M. Burmester and V. Chrissikopoulos, "Secure Transactions with Mobile Agents in Hostile

Environments", *ACISP 2000*, LNCS 1841, pp.289-297, 2000

[12] A. Romao and M. M. da Silva, "An Agent-based Secure Internet Payment System for Mobile Computing", *TrEC'98*, Hamburg, Germany, 3–5 June 1998, LNCS, vol. 1402, Springer.

[13] Visa International and MasterCard International, *Secure Electronic Transaction (SET) specification*, Version 1.0, May 1997.

[14] Y. Zheng, "Digital Signcryption or How to Achieve Cost (Signature&Encryption)<<Cost (Signature)+Cost (Encryption)", in *Advances in Cryptology-CRYPO'97*, vol. 1294, pp.165-179, Springer-Verlag, 1997

[15] Y. Zheng, "Signcryption and Its Applications in Efficient Public Key Solutions", *Information Security Workshop (ISW '97)*, Springer-Verlag, LNCS 1397, pp.291--312. 1998

[16] R. L. Rivest, A. Shamir, L. Adleman, "A Method for Obtaining Digital Signatures and Public-key Cryptosystems", *CACM*, (21), pp.120-126, 1978

[17] G. Hanaoka, Y. Zheng and H. Imai, "LITESET: A Light-weight Secure Electronic Transaction Protocol", *Proc. ACISP'98*, Lecture Notes in Computer Science, Springer-Verlag, vol.1438, pp.215--226, 1998

[18] X. Yi, C. K. Siew, X. F. Wang and E. Okamoto, "A secure Agent-based Framework for the Internet Trading in Mobile Computing Environments", in *Distributed and Parallel Databases*, 8, pp.85-117, 2000

[19] X. Pang, K.-L. Tan, Y. Wang, J. Ren, "A Secure Agent-Mediated Payment Protocol"**,** *Fourth International Conference on Information and Communications Security (ICICS2002), Springer-Verlag, LNCS Vol. 2512,* pp 422-433, 9-12 December 2002, Singapore

[20] U. Whilem, and X. Defago, Objets "Protégés Cryptographiquement", In *Proceedings of RenPar'97*, Lausanne, Switzerland, May 1997.

[21] B. Yee, "A Sanctuary for Mobile Agents", In *Proceedings of the DARPA Workshop on Foundations for Secure Mobile Code*, Monterey CA, USA, March 1997.

[22] T. Sander, and C. Tschudin, "Towards Mobile Cryptography", Technical Report TR-97- 049, International Computer Science Institute, November 1997.

[23] J. Zhou and K.Y. Lam, "Securing Digital Signatures for Non-repudiation", *Computer Communications*, Vol. 22, pp710-716, 1999

[24] J. Zhou, "Achieving Fair Nonrepudiation in Electronic Transactions", *Journal of Organizational Computing and Electronic Commerce*, Vol. 11, No. 4, pp 253-267, 2001

[25] Y. Wang, K.-L. Tan and J. Ren, "A Study of Building Internet Marketplaces on the Basis of Mobile Agents for Parallel Processing", *World Wide Web Journal: Internet and Web Information Systems (WWWJ)*, Kluwer Academics Publisher, Vol. 5, Issue 1, 2002, pp 41-66

[26] Y. Wang, K.-L. Tan and J. Ren, "Towards Autonomous and Automatic Evaluation and Negotiation in Agent-mediated Internet Marketplaces", accepted for publication in *Electronic Commerce Research*, 2003

[27] URL: http://java.sun.com/products/

[28] URL: http://www.trl.ibm.co.jp/aglets/

[29] D. Lange and M. Oshima, *Programming and Deploying Java Mobile Agents with Aglets*, Addison-Wesley Press, Massachusetts, USA, 1998

[30] A. Menezes, P. van Oorschot, and S. Vanstone, "Threshold Schemes", *Handbook of Applied Cryptography*, CRC Press, 1996.

[31] Y. Wang, K.-L. Tan and J. Ren, "PumaMart: A Parallel and Autonomous Agents based Internet Marketplace", *Electronic Commerce Research and Applications*, Elsevier, Vol. 3, No.3, 2004