PAPER *Information Security*

Improving the secure electronic transaction protocol by using signcryption **

Goichiro Hanaoka†, *Nonmember*, **Yuliang Zheng**††, *and* **Hideki Imai**†, *Members*

SUMMARY In the past few years, we have seen the emergence of a large number of proposals for electronic payments over open networks. Among these proposals is the Secure Electronic Transaction (SET) protocol promoted by MasterCard and VISA which is currently being deployed world-widely. While SET has a number of advantages over other proposals in terms of simplicity and openness, there seems to be a consensus regarding the relative inefficiency of the protocol. This paper proposes a light-weight version of the SET protocol, called "LITESET". For the same level of security as recommended in the latest version of SET specifications, LITESET yields a 56.2/51.4% reduction in the computational time in message generation/verification and a 79.9% reduction in communication overhead. This has been achieved by the use of a new cryptographic primitive called *signcryption*. We hope that our proposal can contribute to the practical and engineering side of real-world electronic payments.
key words: signcryption, SET, computational overhead, message overhead

1. Introduction

There is a growing demand for global electronic payments. The Secure Electronic Transaction (SET) protocol is being regarded as one of the important candidates. However, straightforward implementation of SET may impose heavy computation and message overhead on a system that employs SET, primarily due to its use of the RSA digital signature and encryption scheme [11]. This article makes an attempt to improve the efficiency of SET by using a new cryptographic technology called *signcryption*[1], which simultaneously fulfills both the functions of digital signature and public-key encryption in a logically single step. We show how to incorporate signcryption into SET, and evaluate the efficiency of our implementation. Our improved SET will be called "LITESET" or a light-weight Secure Electronic Transaction protocol.

Detailed analysis and comparison shows that LITESET provides a 56.2% reduction in the computational time in message generation, a 51.4% reduction in the computational time in message verification, and a 79.9% reduction in communication overhead.

Section 2 gives a brief review of the SET protocol. Problems with the efficiency of SET are summarized in Section 3. Section 4 proposes an adaptation of signcryption for SET. Our LITESET protocol is also specified in the same section. This is followed by Section 5 where significant improvements of LITESET over SET are presented. Section 6 closes the paper with some concluding remarks.

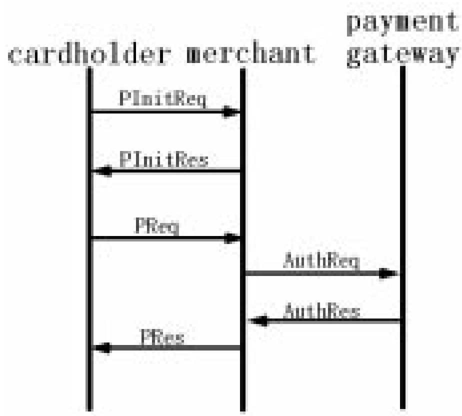

Fig. 1 Flows of the main SET messages.

2. An Overview of SET

The payment model on which SET is based consists of three participants: a cardholder, a merchant, and a payment gateway. The card holder (C) initiates a payment with the merchant (M). The merchant then has to authorize the payment; the payment gateway acts as the front end to the existing financial network, and through this the card issuer can be contacted to explicitly authorize each and every transaction that takes place. In the SET protocol, there are in total 32 different types of messages[4]. These messages are summarized in Table 1. Among the messages, the most important ones and transmitted at the highest frequency are the following six [3],[5]: PInitReq, PInitRes, PReq, PRes, AuthReq and AuthRes. Other messages are used mainly for administrative purposes, such as

Manuscript received xxxx xx, xxxx.
Manuscript revised xxxx xx, xxxx.
†3rd Department, Institute of Industrial Science the University of Tokyo Roppongi 7-22-1, Minato-ku, Tokyo 106, Japan.
The first author is supported by a Research Fellowship from Japan Society for the Promotion of Science(JSPS).
††The Peninsula School of Computing and Information Technology Monash University, McMahons Road, Frankston Melbourne, VIC 3199, Australia
**An extended abstract of this paper was presented at Third Australasian Conference on Information Security and Privacy (ACISP '98)[6]

Table 1 SET messages.

PInitReq,PInitRes	Purchase initialization request/response.
PReq,PRes	Purchase request/response.
AuthReq,AuthRes	Authorization request/response.
AuthRevReq, AuthRevRes	Authorization reversal request/response.
InqReq,InqRes	Inquiry request/response.
CapReq,CapRes	Capture request/response.
CapRevReq, CapRevRes	Capture reversal request/response.
CredReq,CredRes	Credit request/response.
CredRevReq, CredRevRes	Credit reversal request/response.
PCertReq,PCertRes	Payment gateway's certificate request/response.
BatchAdminReq, BatchAdminRes	Batch Administration request/response.
CardCInitReq, CardCInitRes	Cardholder's certificate initialization request/response.
Me-AqCInitReq, Me-AqCInitRes	Merchant's or acquirer's certificate initialization request/response.
RegFormReq, RegFormRes	Registration form request/response.
CertReq,CertRes	Certificate request/response.
CertInqReq, CertInqRes	Certificate inquiry request/response.

creating certificates, canceling messages, registration, error handling etc. Hence these messages are transmitted with significantly smaller frequency than the above mentioned six messages, which in turn implies that any attempt to improve the efficiency of SET must focus on the six main messages. The flow of the six main messages is shown in Figure 1.

Next we discuss in detail the functions of the six dominant messages. A few frequently used notations are summarized in Table 2.

The SET protocol starts with Purchase Initialization (implementation of PInitReq and PInitRes is shown in Table 3). Purchase Request is then executed conforming to the structure described in Table 4. In PReq, PI and OI are destined to different entities but sent in the same cryptographic envelope. They share a signature called *dual signature*[4],[5] which can be verified by either entity. Dual signature used in SET is constructed as illustrated in Table 4.

After receiving PReq, the merchant verifies it (especially, Dual signature). If it is valid, he produces AuthReq and sends it to the payment gateway (P). AuthRseq includes AuthReqData and PI, where PI is copied from PReq.

Upon receiving AuthReq, the payment gateway verifies it. If successful, the payment gateway sends AuthRes back to the merchant. AuthRes includes CapToken and AuthResData, which shows the state of the transaction. If the verification of AuthReq fails, only AuthResData is sent as AuthRes. Table 5 shows the structure of AuthReq/Res.

Finally, the protocol is finished with PRes produced by the merchant (the structure of PRes is shown in Table 6).

Table 2 Notations.

$E_k(t)$	to encrypt t by using a key k.
$D_k(t)$	to decrypt t by using a key k.
$H(t)$	to hash t.
Pv_e	participant e's private key.
Pb_e	participant e's public key.

Table 3 Structure of PInitReq/Res.

message	message factor
PInitReq	{RRPID,LID-C,Chall_C,BrandID,BIN}
PInitRes	{PInitResData, $E_{Pv_M}(H(\text{PInitResData}))$}
RRPID	UniqueID for one pair of request and response.
LID-C	LocalID of cardholder's transaction.
Chall_C	Cardholder's challenge.
BIN	Cardholder's account number.
PInitResData	{TransID,RRPID, Chall_C, Chall_M,PEThumb}
TransID	TransactionID.
Chall_M	Merchant's challenge.
BrandID	Brand of card.
PEThumb	Thumbprint of payment gateway public key certificate.

Table 4 Structure of PReq.

message	message factor
PReq	{PI,OI}
PI	{$E_{Pb_P}(k$,PANData, nonce), E_k(PI-OILink,H(PANData,nonce)), **Dual signature** }
OI	{ OIData,H(PIData) }
PANData	Primary account number data.
PIData	Purchase instruction data.
OIData	Order information data.
PI-OILink	{PIData(except PANData),H(OIData)}
Dual signature	$E_{Pv_C}\{H(H(\text{PIData}),H(\text{OIData}))\}$

Table 5 Structure of AuthReq/Req.

message	massage factor
AuthReq	{$E_{Pb_P}(k)$, PI, E_k(AuthReqData,H(PI), E_{Pv_M}(H(AuthReqData,H(PI))))}
AuthRes	{$E_{Pb_M}(k)$, CapToken, E_k(AuthResData,H(Captoken), E_{Pv_P}(H(AuthResData,H(CapToken))) }
AuthReqData	Authorization request data.
AuthResData	Authorization response data.

3. Problems with the Efficiency of SET

As mentioned above, all the public-key encryption and digital signature used in SET are based on the RSA scheme. RSA requires a relatively large computational cost and large message overhead. Based on "square-and-multiply" and "simultaneous multiple exponentiation"[9], the main computational cost for one public-key

Table 6 Structure of PRes.

message	message factor
PRes	{PResData,$E_{Pv_M}(H(\text{PResData}))$}
PResData	Purchase response data.

encryption or one digital signature generation is estimated to be $\frac{1.5}{4} \cdot |n|$ modulo multiplications where n is a composite of the RSA scheme. For PReq generation, for example, one public-key encryption and one digital signature generation are required, therefore the computational cost is estimated to be 768 modulo multiplications ($|n| = 1024$bit). Part of Table 9 shows computational costs for message generations and verifications in SET, respectively.

Turning now to message or communication overhead, digital signatures and public-key encrypted session keys are regarded as the main overhead. In addition, the hash variables (160bit) for message linking are also regarded as message overhead. The message overhead for one digital signature or public-key encrypted session key is estimated to be $|n|$. Hence, as an example, for PReq generation, there are one public-key encryption, one digital signature, and three hashed variables, so that the message overhead is estimated to be 2008 bit (PANData and the session key are altogether encrypted with the cardholder's public key, so that the message overhead is less than the total amount mentioned above). Table 11 shows the message overhead in SET.

4. LITESET — a Light-Weight Version of SET

In this section, we will show how to improve SET in terms of efficiency: specifically, how to adapt signcryption for SET. The most important part of this work is how to link a message to another message. In our improvement, there are two kinds of efficient linking: LinkedData and CoupledData. The details appear in the following subsection.

4.1 Difficulty of applying signcryption to SET

As it is well known, since signcryption executes completely different two procedures simultoneously, there often occur several problems in the implementations of certain secrity systems. In SET, the problem of straightforwardly applying signcryption is as follows: signcryption dose not provide efficient message linking though this function is very often required in SET. For example, in PReq the relationship between the informtion for the payment and that for the order must be guaranteed. Namely, PIData and OIData are linked with each other in the message. In conventional SET, this requirement is fulfilled by the dual signature. However, it is very difficult to provide the same property of the dual signature by signcryption for the above reason; alghouth a dual signature can be veirfied by both the merchant and the payment gateway, a signcrypted message cannot be (assuming usual computational costs). Therefore, straightforwardly applied signcryption is not suitable for SET. Hence, in order to apply signcryption to SET we need to construct modified signcryption schemes which provide the function of message linking. In this section, we show the modified signcryptions which fulfill two kinds of message linking in SET.

Table 7 Parameters for LITESET messages.

$KH_k(t)$	to hash t with a key k.
p	a large prime.
q	a large prime factor of $p-1$.
g	an integer in $[1, \cdots, p-1]$ with order q modulo p.

4.2 Notation

Table 7 shows the parameters which are used in this paper (notice that $E_x(t)$, $D_x(t)$, $H(t)$, Pv_e and Pb_e are defined in Table 2). We define the public key of entity e as $Pb_e = g^{Pv_e} \bmod p$.

4.3 LinkedData

In SET, we often find a situation where the sender (S) has to

· sign the message M_1,

· encrypt it with the recipient(R)'s public key,

· and show the relationship between M_1 and ceratain M_2.

In conventional SET, to satisfy such demands, $H(M_2)$ is attached to M_1, and these messages are signed by using S's private key and then encrypted by using R's public key. Then, R can verify the linking between M_1 and M_2 by checking the value of $H(M_2)$. Namely, if someone falsifies M_2, R can find that M_2 is falsified.

For efficient application of signcryption scheme, we use hashed M_2 in the verification of the signcrypted M_1. These linked messages are referred to as *LinkedData*.

Now let us proceed by showing how to construct *LinkedData*. The message to be sent by S to R is $LinkedData_{S,Pb_R}(M_1, M_2)$ which is composed as follows:

- $LinkedData_{S,Pb_R}(M_1, M_2)$
  $= \{LSC_{S,Pb_R,M_2}(M_1), M_2\}$
  where $LSC_{S,Pb_R,M_2}(M_1) = \{r, s, c\}$, and r, s, c are defined by:

  $x \in_R [1, \cdots, q-1]$

  $(k_1, k_2) = H({Pb_R}^x \bmod p)$

  $r = KH_{k_1}(H(M_1), H(M_2))$

  $s = \frac{x}{r+Pv_S} \bmod q$

$c = E_{k_2}(M_1)$

On receiving $LinkedData_{S,Pb_R}(M_1, M_2)$, R verifies it as follows:

1. $(k_1, k_2) = H((Pb_S \cdot g^r)^{s \cdot Pv_R} \bmod p)$
2. $M_1 = D_{k_2}(c)$
3. If $r = KH_{k_1}(H(M_1), H(M_2))$, R accepts M_1, M_2.

Accordingly, in order to be able to verify the message M_1, unfalsified $H(M_2)$ is required. Thus, if someone falsifies M_2, R can detect that it is indeed falsified. As examples, AuthReq and AuthRes can be described as *LinkedData*.

4.4 CoupledData

Generally, dual signature is used for linking two messages whose recipients are different. Thus, although one recipient can only see the contents of the message M_1 he receives, he can be confident of the digest $H(M_2)$ of the other message M_2. Hence, if one recipient wants to confirm the linking of the two messages, the two recipients send dual signatures $E_{Pv_S}(H(H(M_1), H(M_2)))$, messages and message digests they received to a reliable institution. By using them and sender's public key, the reliable institution can detect a dishonest act. If D_{Pb_S}(dual signature) is not identical to $H(H(M_1), H(M_2))$ which is made from components sent by one recipient, the reliable institution knows this recipient forged M_1 and/or $H(M_2)$. And, if dual signatures are valid and M_1(or M_2) which is received by one recipient is not hashed to be $H(M_1$(or $M_2))$ which is received by the other recipient, the reliable institution knows the sender conducted a dishonest act.

Here we show how to realize the function of dual signature by applying signcryption. Let the messages which are linked by using this scheme be called *CoupledData*.

When S sends PReq to R, S must

· sign the messages, M_1 and M_2,

· encrypt only M_1 by using R''s public key,

· send M_1 and M_2 to R,

· let R send M_1 to R' with keeping M_1 unread,

· and show the relationship between M_1 and M_2

where R' is the true recipient of M_1. In SET, C acts S, M acts R, and P acts R'.

In our implementation, S send $CoupledData_{S,Pb_{R'}}(M_1, M_2)$ to R as follows:

- $CoupledData_{S,Pb_{R'}}(M_1, M_2)$
  $= \{CSC_{S,Pb_{R'},M_2}(M_1), CSig_{S,M_1}(M_2)\}$

  ◇ $CSig_{S,M_1}(M_2) = \{s_1, r_1, M_2, H(M_1)\}$

  $x_1 \in_R [1, \cdots, q-1]$

  $r_1 = H(g^{x_1}, H(M_1), H(M_2)[, etc])$

  $s_1 = \frac{x_1}{r_1 + Pv_S} \bmod q$

  Upon receiving $CoupledData_{S,Pb_{R'}}(M_1, M_2)$, R verifies it as follows:

  1. $(g^{x_1}) = H((Pb_S \cdot g^{r_1})^{s_1} \bmod p)$
  2. If $r_1 = H(g^{x_1}, H(M_1), H(M_2)[, etc])$, R accepts M_2, and sends $CSC_{S,Pb_{R'},M_2}(M_1)$ and $H(M_2)$ to R'.

  ◇ $CSC_{S,Pb_{R'},M_2}(M_1) = \{r_2, s_2, c_2\}$

  $x_2 \in_R [1, \cdots, q-1]$

  $(k_1, k_2) = H({Pb_{R'}}^{x_2} \bmod p)$

  $r_2 = KH_{k_1}(H(M_1), H(M_2)[, etc])$

  $s_2 = \frac{x_2}{r_2 + Pv_S} \bmod q$

  $c_2 = E_{k_2}(M_1)$

  R' verifies $CSC_{S,Pb_{R'},M_2}(M_1)$ as follows:

  1. $(k_1, k_2) = H((Pb_S \cdot g^{r_2})^{s_2 \cdot Pv_{R'}} \bmod p)$
  2. $\{M_1\} = D_{k_2}(c_2)$
  3. If $r_2 = KH_{k_1}(H(M_1), H(M_2)[, etc])$, R' accepts M_1.

If S wants to designate the recipient of the message, S should put the recipient's public key in *etc*.

If S wants to encrypt M_2, S should send *CoupledData* as follows:

- $CoupledData_{S,Pb_{R'},Pb_R}(M_1, M_2)$
  $= \{CSC_{S,Pb_{R'},M_2}(M_1), CSC_{S,Pb_R,M_1}(M_2)\}$

  ◇ $CSC_{S,Pb_R,M_1}(M_2) = \{s_1, r_1, c_1\}$

  $x_1 \in_R [1, \cdots, q-1]$

  $(k_3, k_4) = H({Pb_R}^{x_1} \bmod p)$

  $s_1 = \frac{x_1}{r_1 + Pv_S} \bmod q$

  $r_1 = KH_{k_3}(H(M_1), H(M_2)[, etc])$

  $c_1 = E_{k_4}(M_1)$

  R verifies $CSC_{S,Pb_R,M_1}(M_2) = \{s_1, r_1, c_1\}$ as follows:

  1. $(k_3, k_4) = H((Pb_S \cdot g^{r_1})^{s_1 \cdot Pv_R} \bmod p)$
  2. $\{M_2\} = D_{k_4}(c_1)$
  3. If $r_1 = KH_{k_3}(H(M_1), H(M_2)[, etc])$, R accepts M_1 (of course, S has to send $H(M_1)$ with $CoupledData_{S,Pb_{R'},Pb_R}(M_1, M_2)$), and should send $CSC_{S,Pb_{R'},M_2}(M_1)$ and $H(M_2)$.

Although dishonest acts are detected in almost the same way as in dual signature scheme, there exist several differences. (1) recipient's private keys are required for detection. (2) although the two recipients can be confident that they have received the same signature in

Table 8 Message structures of LITESET for main messages.

message	message structure
PInitReq	{RRPID,LID-C,Chall_C,BrandID,BIN}
PInitRes	$\{Sig_M$(PInitResData)$\}$
PReq	$\{CoupledData_{C,Pb_P}$(PIData,OIData)$\}$ If OIData is encrypted, $\{CoupledData_{C,Pb_P,Pb_M}$(PIData,OIData), H(PIData)$\}$
AuthReq	$\{LinkedData_{M,Pb_P}$(AuthReqData, $\{CSC_{S,Pb_P,OIData}$(PIData),H(OIData)$\}$)$\}$
AuthRes	$\{LinkedData_{P,Pb_M}$(AuthResData, CapToken)$\}$
PRes	$\{Sig_M$(PResData)$\}$

the conventional SET, recipients cannot be confident of the signature which is received by the other recipient in our scheme. With our scheme, more computational costs need to be invested to detect dishonest acts. However, as the need of detection of dishonest acts should arise in very rare situations, we believe that the extra computational costs for detecting dishonest acts with our scheme should not be a disadvantage in practice.

4.5 Messages in LITESET

Embodying *LinkedData* and *CoupledData* in SET results is a light weight version of the protocol called LITESET. For the six main messages, *LinkedData* is adapted to AuthReq ((M_1, M_2) =(AuthReqData, PI)) and AuthRes ((M_1, M_2) = (AuthResData, CapToken)), and *CoupledData* is adapted to PReq ((M_1, M_2) = (PIData, OIData)). Moreover, to sign only, such as PInitRes and PRes, SDSS1[1] is adapted to such messages. Accordingly, the six main messages in LITESET are described in Table 8.

For other messages, operations mentioned above are adapted appropriately to their message type, employing a similar approach. A detailed description of these messages is shown in Table A·1. Here, we show only their structure, and message factors in them are not discussed.

5. LITESET v.s. SET

LITESET relies for its security on the computational infeasibility of the discrete logarithm problem. Assuming the difficulty of computing the discrete logarithm, the signcryption scheme embodied in LITESET has been proven secure against adaptively chosen ciphertext attacks (the most powerful attacks that one can conceive in the real world)[2], [8]. This means the security level of LITESET is same as the conventional SET with *optimal asymmetric encryption padding*(OAEP)[7]. Similar to the original SET protocol, the LITESET protocol is secure in practice.

The rest of this section is devoted to a detailed comparison of the efficiency of LITESET and SET. Here, we compare LITESET with SET based on RSA,

Table 9 Computational cost for message generation/verification of main messages (discrete-logarithm based LITESET).

message	conventional scheme	our scheme	saving
PInitReq	-/-	-/-	-/-
PInitRes	384/384	240/280	37.5%/27.1%
PReq	768/384	480/280	37.5%/27.1%
AuthReq	768/1536	240/560	68.7%/63.5%
AuthRes	1536/768	480/280	68.7%/63.5%
PRes	384/384	240/280	37.5%/27.1%
Total	3840/3456	1680/1680	56.2%/51.4%

Table 10 Computational cost for message generation/verification of main messages (LITESET on elliptic curves).

message	conventional scheme	our scheme on elliptic curves	saving
PInitReq	-/-	-/-	-/-
PInitRes	384/384	24/28	93.7%/92.7%
PReq	768/384	48/28	93.7%/92.7%
AuthReq	768/1536	24/56	96.9%/96.3%
AuthRes	1536/768	48/28	96.9%/96.3%
PRes	384/384	24/28	93.7%/92.7%
Total	3840/3456	168/168	95.6%/95.1%

Table 11 Message overhead of main messages.

message	conventional scheme	our scheme	saving
PInitReq	-	-	-
PInitRes	1024bit	320bit	68.7%
PReq	2008bit	720bit	64.1%
AuthReq	4056bit	640bit	84.2%
AuthRes	4256bit	480bit	88.7%
PRes	1024bit	320bit	68.7%
Total	12368bit	2480bit	79.9%

which is the most common implementation. Of course elliptic cryptosystems are known as quite efficient cryptographical technologies. *Signcryption on elliptic curves*[12] has been already proposed, and we can realize LITESET on elliptic curves easily. Therefore, we also evaluate the performance of LITESET on elliptic curves.

5.1 Computational costs

The computational cost depends mainly on modulo exponentiations in encryption or signature generation. Hence, the number of modulo multiplications in modulo exponentiation can be used as the computational cost. We estimate the number of modulo multiplications by using "square-and-multiply" and "simultaneous multiple exponentiation". Namely, the number of modulo multiplications for one g^x or ${Pb_e}^x$ is $1.5 \cdot |q|$, and for $(Pb_{e_1} \cdot g^r)^{s \cdot Pv_{e_2}}$ it is equal to $\frac{7}{4} \cdot |q|$. In conventional SET, 1024bit RSA composite is used. To achieve the same security level, $|q| = 160$bit and $|p| = 1024$bit should be chosen for our scheme[1]. Table 9 shows the costs of message generation and verification for the six main messages. We can see that the computational costs are saved over 50%[††]. For other messages, Table A·2 shows the costs of message generation and verification, respectively, where we can also see the significant cost reduction. Note that LITESET can be applied to almost all of the computers and that we do not assume any specifed computers. The actual time depends on the particular computer used in SET and LITESET. As an example, on M16C processor[13] the computational time for PReq generation in the conventional SET is estimated to be 10sec approximately. Therefore, that in LITESET becomes 5sec on the same processor. Additionally, we roughly estimate the performance of LITESET on elliptic curves, assuming that the computational cost for point addition on elliptic curves is 1/10 of that for modulo multiplication in the conventional discrete-logarithm based cryptosystems. In Table 10, the computational cost of LITESET on elliptic curves is shown. The cost reduction can be considered as significant.

In a most probable situation, cardholder's computer is much slower than merchant's and payment gateway's. Hence, the efficiency depends largely on the load on cardholder's computer. Our proposal reduces this load significantly; PReq(generation), PInitRes(verification) and PRes(verification) are managed on cardholder's computer, and their computational costs are saved as much as 37.0%.

On the implementaion on IC cards, since *coprocessors* are well-optimized for modulo multiplication, modulo division, e.g., s in *LinkedData*, is not desirable. However, in LITESET the number of modulo divisions is significantly smaller than that of modulo multiplications. Hence, we consider that the inefficiency of the modulo division can be ignored.

5.2 Message overhead

In our evaluation, digital signature and public key encrypted session key are regarded as message overhead. Namely, for our scheme, $r(|r| = 80\text{bit})$, $s(|s| = 160\text{bit})$ and hashed variables$(|H(t)| = 160\text{bit})$ for message linking are message overhead. Table 11 shows the message overhead of the six main messages. We see that message overhead is saved over 70% for each message. Table A·3 shows the message overhead of other messages; hence the reduction of message overhead is also significant. Note that in LITESET on elliptic curves the message overhead is same as that in the discrete-logarithm based LITESET.

[††]It is difficult to make quantitative analysis of computational costs involved in certificate verification, which heavily depends on the structure of a certification infrastructure employed. Thus, we do not investigate them here.

5.3 Future parameters

We should also consider situations that require larger security parameters. On account of the continuing developments in computer technologies, we will certainly need larger security parameters in the future. Even at the present time, we can often reach specific situations that the payment should be done more safely. Table 12 shows the advantage of LITESET over RSA-based SET with larger parameters. Here, LITESET's advantage is estimated by using the average computational cost and message overhead for six main messages assuming different security parameters. We can find that LITESET's advantage will be more significant in the future.

6. Conclusion

In this paper, a new and very practical method which reduces computational cost and message overhead of SET messages is proposed based on signcryption. In SET, messages are often signed, encrypted and linked to other messages. With the help of signcryption, all of these functions are fulfilled, but with a far smaller cost than that required by SET. In the future, security parameters will be larger to compensate advances in cryptanalysis, and the advantages of our proposed LITESET over the current version of SET, based on RSA, will be more significant.

Acknowledgments

We would like to thank Chandana Gamage and Kanta Matsuura for their help in preparing this paper.

References

[1] Y. Zheng, "Digital signcryption or how to achieve cost (signature & encryption) << cost (signature) + cost (encryption)," In Advances in *Cryptology - CRYPTO'97*, volume 1294 of *Lecture Notes in Computer Science*, pp.165-179, Springer-Verlag, Berlin, 1997.
[2] Y. Zheng, "Signcryption or how to achieve cost (signature & encryption) << cost (signature) + cost (encryption)," manuscript, 1999.
[3] MasterCard and Visa, " Secure electronic transaction (SET) specification book 1: Business Decryption," May 1997.
[4] MasterCard and Visa, "Secure electronic transaction (SET) specification book 2: Programmer's Guide," May 1997.
[5] Donal O'Mahony, Michael Peirce and Hitesh Tewari, "Electronic Payment Systems," Artech House Publishers, 1997.
[6] G. Hanaoka, Y. Zheng and H. Imai, "LITESET: a Light-Weight Secure Electronic Transaction," Proc. of ACISP '98, volume 1438 of *Lecture Notes in Computer Science*, pp.215-226, Springer-Verlag, Berlin, 1998.
[7] M. Bellare and P. Rogaway, "Optimal Asymmetric Encryption," Proc. of Eurocrypt'94, volume 950 of *Lecture Notes in Computer Science*, pp.92-111, Springer-Verlag, Berlin, 1994.
[8] D. Pointcheval and J. Stern, "Security Proofs for Signature

Table 12 Saving in computational cost (for message generation/verification) and message overhead of LITESET over the RSA-based SET for future parameters.

$\lvert p\rvert = \lvert n\rvert$	$\lvert q\rvert$	$\lvert KH(\cdot)\rvert$	computational cost for message generation/verification	message overhead
1024bit	160bit	80bit	56.2%/51.4%	79.9%
1536bit	176bit	88bit	67.9%/64.4%	85.4%
2048bit	192bit	96bit	73.7%/70.8%	88.0%
3072bit	224bit	112bit	79.6%/77.3%	90.7%
4096bit	256bit	128bit	82.5%/80.6%	92.0%
5120bit	288bit	144bit	84.2%/82.5%	92.8%
8192bit	320bit	160bit	89.1%/87.8%	95.0%

Schemes," Proc. of Eurocrypto'96, volume 1070 of *Lecture Notes in Computer Science*, pp.387-398, Springer-Verlag, Berlin, 1996.

[9] T. ElGamal. "A Public Key Cryptosystem and a Signature Scheme Based on discrete Logarithms," *IEEE Trans. Information Theory,* Vol. IT-31, no. 4, pp.468-472, 1985.

[10] National Institute for Standards and Technology, "Specifications for a digital signature standard (DSS)," Federal Information Processing Standard Publication 186, U.S Department of Commerce, May 1994.

[11] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptsystems," *Communications of the ACM,* 21(2), pp.120-128, 1978.

[12] Y. Zheng and H. Imai, "Efficient Signcryption Schemes on Elliptic Curves," Proc. of IFIP SEC'98 (*CD-ROM*), Chapman & Hall, Sept. 1998, Vienna.

[13] "User Manual of M16C/60 Series," Mitsubishi Electric Corporation, 1996.

Appendix A: Signcryption, SDSS1 and RSA with OAEP

This appendix is intended to give a brief summary of signcryption[1], a shortened digital signature scheme called SDSS1[1], and the RSA with OAEP scheme[7], [11]. The reader is directed to the original references for further details of the schemes.

A.1: Signcryption

Signcryption is a new cryptographic technology that can reduce computational cost and message overhead by using an idea to manage digital signature and public key encryption simultaneously. For example, it can be implemented as follows[1]. We define the public key of an entity e as $Pb_e = g^{Pv_e} \bmod p$. When the sender(S) sends a $message$ to the recipient(R), S sends the $message$ in a signcrypted form $SC_{S,Pb_R}(message) = r, s, c$ where

- $x \in_R [1, \cdots, q-1]$
  $(k_1, k_2) = H({Pb_R}^x \bmod p)$
  $r = KH_{k_1}(H(message))$
  $s = \frac{x}{r+Pv_S} \bmod q$
  $c = E_{k_2}(message)$

  On receiving $SC_{S,Pb_R}(message)$, R verifies it as follows:

  1. $(k_1, k_2) = H((Pb_S \cdot g^r)^{s \cdot Pv_R} \bmod p)$
  2. $message = D_{k_2}(c)$
  3. If $r = KH_{k_1}(H(message))$,
     R accepts $message$.

A.2: SDSS1 — A Shortened Digital Signature Scheme

SDSS1 proposed in [1] is an improvement of DSS[10]. If S wants to sign $message$, S sends $Sig_S(message)$ as follows:

- $Sig_S(message) = \{s, r, message\}$
  $x \in_R [1, \cdots, q-1]$
  $s = \frac{x}{r+Pv_S} \bmod q$
  $r = H(g^x, message)$

  R verifies $Sig_S(message) = \{s, r, message\}$ as follows:

  1. $(g^x) = H((Pb_S \cdot g^r)^s \bmod p)$
  2. If $r = H(g^x, message)$,
     R accepts $message$.

A.3: The RSA with OAEP cryptosystem

Suppose n_S is the enough large composite with factoring difficulty, S calculates two integers e_S and d_S each having roughly the same size and satisfying $e_S d_S = 1 \bmod \lambda(n_S)$, where $\lambda()$ is the Carmichael function. Then, S uses (e_S, n_S) for S's public key and (d_S) for S's private key. S's signature on $message$ is defined as $s = H(message)^{d_S} \bmod n_S$. Other user can verify whether s is S's valid signature on $message$ by checking whether $H(message)$ is identical to $s^{e_S} \bmod n_S$.

Similarly to S, R can create R's public key (e_R, n_R) and secret key d_R. Let G and F be random oracles $G: \{0,1\}^{k_0} \rightarrow \{0,1\}^{n+k_1}$ and $F: \{0,1\}^{n+k_1} \rightarrow \{0,1\}^{k_0}$, respectively, where $n = \lvert$message-encryption key$\rvert$ and $n + k_0 + k_1 = \lvert n_R \rvert$. To send $message$ to R in a secure way, S picks random message-encryption key k and calculates $z = (k \| 0^{k_1}) \oplus G(r)$, where r is a k_0-bit random number. Then S sends to R $c_1 = E_k(message)$

Table A·1 Message structures of LITESET for other messages.

message	structure
AuthRevReq	$\{LinkedData_{M,Pb_P}$(AuthRevReqData, {PI,CapToken})}
AuthRevRes	$\{LinkedData_{P,Pb_M}$(AuthRevResData, {CapTokenNew, AuthTokenNew})}
CapReq	$\{LinkedData_{M,Pb_P}$(CapReqData, CapTokenSeq)}
CapRes	$\{SC_{P,Pb_M}$(CapResData)}
CapRevReq	$\{LinkedData_{M,Pb_P}$(CapRevData, CapTokenSeq)}
CapRevRes	$\{SC_{P,Pb_M}$(CapRevResData)}
CredReq	$\{LinkedData_{M,Pb_P}$(CredReqData, CapTokenSeq)}
CredRes	$\{SC_{P,Pb_M}$(CredResData)}
CredRevReq	$\{LinkedData_{M,Pb_P}$(CredRevReqData, CapTokenSeq)}
CredRevRes	$\{SC_{P,Pb_M}$(CredRevResData)}
PCertReq	$\{Sig_M$(PCertReqData)}
PCertRes	$\{Sig_P$(PCertResData)}
BatchAdminReq	$\{SC_{M,Pb_P}$(BatchAdminReqData)}
BatchAdminRes	$\{SC_{P,Pb_M}$(BatchAdminResData)}
CardCInitReq	{RRPID,LID-EE,Chall_EE,BrandID}
CardCInitRes	$\{Sig_{CA}$(CardCInitResTBS)}
Me-AqCInitReq	{RRPID,LID-EE,Chall_EE,RequestType, IDData, BrandID, Language}
Me-AqcInitRes	$\{Sig_{CA}$(Me-AqCInitResTBS)}
RegFormReq	$\{SC_{EE,Pb_{CA}}$({RegFormReqData, PANOnly})}
RegFormRes	$\{Sig_{CA}$(RegFormResTBS)}
CertReq	$\{SC_{EE,Pb_{CA}}$({CertReqData,AcctInfo})}
CertRes	$\{Sig_{CA}$(CertResData)}
CertInqReq	$\{Sig_{CA}$(Me-AqCInitResTBS)}
CertInqRes	$\{SC_{EE,Pb_{CA}}$({CertReqData,AcctInfo})}

and $c_2 = \{z\|(r \oplus F(z))\}^{e_R} \bmod n_R$. Upon receiving c_1 and c_2, R can retrieve k by calculating $z = [{c_2}^{d_R} \bmod n_R]^{n+k_1}$, $r = [{c_2}^{d_R} \bmod n_R]_{k_0} \oplus F(z)$ and $k = [z \oplus G(r)]^n$, employing it he can decrypt c_1.

Appendix B: Evealuation of other messages

In this appendix, we show message structures, computational cost and message overhead of LITESET messages except for the main six messages. Table A·1, Table A·2 and Table A·3 show the message structure, the computational cost and the message overhead, respectively.

Table A·2 Computational cost for message generation/verification for other messages (discrete-logarithm based LITESET).

message	conventional scheme	our scheme	saving
AuthRevReq	768/1536	240/560	68.7%/63.5%
AuthRevRes	768/768	240/280	68.7%/63.5%
CapReq	768/768	240/280	68.7%/63.5%
CapRes	768/768	240/280	68.7%/63.5%
CapRevReq	768/768	240/280	68.7%/63.5%
CapRevRes	768/768	240/280	68.7%/63.5%
CredReq	768/768	240/280	68.7%/63.5%
CredRes	768/768	240/280	68.7%/63.5%
CredRevReq	768/768	240/280	68.7%/63.5%
CredRevRes	768/768	240/280	68.7%/63.5%
PCertReq	384/384	240/280	68.7%/27.1%
PCertRes	384/384	240/280	68.7%/27.1%
BatchAdminReq	768/768	240/280	68.7%/63.5%
BatchAdminRes	768/768	240/280	68.7%/63.5%
CardCInitReq	-/-	-/-	-/-
CardCInitRes	384/384	240/280	68.7%/27.1%
Me-AqCInitReq	-/-	-/-	-/-
Me-AqcInitRes	384/384	240/280	37.5%/27.1%
RegFormReq	384/384	240/280	37.5%/27.1%
RegFormRes	384/384	240/280	37.5%/27.1%
CertReq	768/768	240/280	68.7%/63.5%
CertRes	384/384	240/280	37.5%/27.1%
CertInqReq	384/384	240/280	37.5%/27.1%
CertInqRes	384/384	240/280	37.5%/27.1%

Table A·3 Message overhead for other messages.

message	conventional scheme	our scheme	saving
AuthRevReq	6114bit	880bit	85.6%
AuthRevRes	4256bit	480bit	88.7%
CapReq	2208 +(2048·n)bit	240 +(240·n)bit	≅88.3%
CapRes	2048bit	240bit	88.3%
CapRevReq	2208 +(2048·n)bit	240 +(240·n)bit	≅88.3%
CapRevRes	2048bit	240bit	88.3%
CredReq	2208 +(2048·n)bit	240 +(240·n)bit	≅88.3%
CredRes	2048bit	240bit	88.3%
CredRevReq	2208 +(2048·n)bit	240 +(240·n)bit	≅88.3%
CredRevRes	2048bit	240bit	88.3%
PCertReq	1024bit	320bit	68.7%
PCertRes	1024bit	320bit	68.7%
BatchAdminReq	2048bit	240bit	88.3%
BatchAdminRes	2048bit	240bit	88.3%
CardCInitReq	-	-	-
CardCInitRes	1024bit	320bit	68.7%
Me-AqCInitReq	-	-	-
Me-AqcInitRes	2048bit	240bit	88.3%
RegFormReq	1184bit	872bit	26.4%
RegFormRes	1024bit	320bit	68.7%
CertReq	1528bit	240bit	84.3%
CertRes	1024bit	320bit	68.7%
CertInqReq	1024bit	320bit	68.7%
CertInqRes	1024bit	320bit	68.7%

Goichiro Hanaoka is a Ph.D. student in electronic information engineering at The University of Tokyo, Tokyo, Japan. He received his bachelors and Masters degrees in electronic engineering and electronic information engineering also from The University of Tokyo in 1997 and 1999, respectively. His research interests are in

the fields of unconditionally secure cryptography, electronic payments and network security.

Yuliang Zheng received his B.Sc. degree in computer science from Nanjing Institute of Technology, China, in 1982, and the M.E. and Ph.D. degrees, both in electrical and computer engineering, from Yokohama National University, Japan, in 1988 and 1991 respectively. From 1982 to 1984 he was with the Guangzhou Research Institute for Communications, Guangzhou (Canton), China. Since 1991 he has worked for a number of academic institutions in Australia. Currently he is Reader of the Faculty of Information Technology, Monash University, in Melbourne, and heads Monash's Laboratory for Information and Network Security (LINKS). He served as the program committee co-chair of the 1998, 1999 and 2000 International Workshops on Practice and Theory in Public Key Cryptography. His research interests include cryptography and its applications secure electronic commerce. Dr. Zheng is a member of IACR, ACM and IEEE.

Hideki Imai was born in Shimane, Japan on May 31, 1943. He received the B.E., M.E. and Ph.D. degrees in electrical engineering from the University of Tokyo, Japan, in 1966, 1968 and 1971, respectively. From 1971 to 1992 he was on the faculty of Yokohama National University. In 1992 he joined the faculty of the University of Tokyo, where he is currently a Full Professor in the Institute of Industrial Science. His current research interests include information theory, coding theory, cryptography, spread spectrum systems and their applications. He received Excellent Book Awards from IEICE in 1976 and 1991. He also received the Best Paper Award (Yonezawa Memorial Award) from IEICE in 1992, the Distinguished Services Award from the Association for Telecommunication Promotion in 1994, the Telecom System Technology Prize from the Telecommunication Advancement Foundation and Achievement Award from IEICE in 1995. In 1998 he was awarded Golden Jubilee Paper Award by the IEEE Information Theory Society. He was elected an IEEE Fellow for his contributions to the theory of coded modulation and two-dimensional codes in 1992. He chaired several committees of scientific societies such as the IEICE Professional Group on Information Theory. He served as the editor of several scientific journals of IEICE, IEEE etc. He chaired a lot of international conferences such as 1993 IEEE International Theory Workshop and 1994 International Symposium on Information Theory and Its Applications (ISITA'94). Dr. Imai has been on the board of IEICE, the IEEE Information Theory Society, Japan Society of Security Management (JSSM) and the Society of Information Theory and Its Applications (SITA). At present he serves as President of the IEICE Engineering Sciences Society.