LITESET: a Light-Weight Secure Electronic Transaction Protocol

Goichiro Hanaoka[1], Yuliang Zheng[2] and Hideki Imai[1]

[1] The 3rd Department, Institute of Industrial Science
the University of Tokyo
Roppongi 7-22-1, Minato-ku, Tokyo 106, Japan
Phone & Fax: +81-3-3402-7365
E-Mail: {hanaoka,imai}@imailab.iis.u-tokyo.ac.jp

[2] The Peninsula School of Computing and Information Technology
Monash University, McMahons Road, Frankston
Melbourne, VIC 3199, Australia
Email: `yzheng@fcit.monash.edu.au`
URL: `http://www-pscit.fcit.monash.edu.au/~yuliang/`
Phone: +61 3 9904 4196, Fax: +61 3 9904 4124

Abstract. The past few years 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 worldwide. 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 shows a 53.1/53.7% 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.

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 [7]. 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 represents a 53.1% reduction in the computational time in message generation, a 53.7% 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.

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 initiates a payment with the merchant. The merchant then has the payment authorized; 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[3]. There messages are summarized in Table 1. Among these messages, among which the most important and transmitted at the highest frequency are the following six [2],[4]: PInitReq, PInitRes, PReq, PRes, AuthReq and AuthRes. Each abbreviated message is summarized in Table 2. Other messages are used mainly for administrative purposes, such as creating certificates, canceling messages, registration, error handling etc. Hence these message are transmitted at a far 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 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[3],[4] which can be verified by either entity. Dual signature used in SET is constructed as illustrated in Table 4.

On 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.

On 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.

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.

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

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

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"[5], the main computational cost for one public-key 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 ($n = 1024bits$). Part of Table 9 shows computational costs for message generations and verifications in SET, respectively.

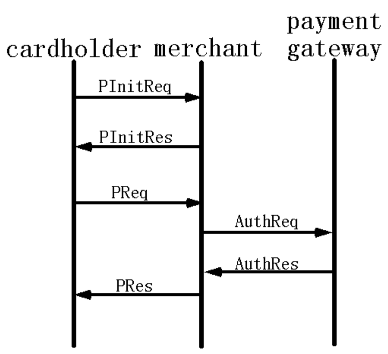

Fig. 1. Flows of SET messages

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.

Turning now to message or communication overhead, digital signatures and public-key encrypted session keys are regarded as the main overhead. Besides them, hashed variables (160bits) 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 the message overhead is estimated to be 2008 bits (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). Part of Table 10 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

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	$E_{Pv_C}\{H(H($PIData$),H($OIData$))\}$

Table 5. Structure of AuthReq/Req.

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

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 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). In the following, we define the public key of entity e as $Pb_e = g^{Pv_e} \bmod p$.

4.2 LinkedData

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

Table 6. Structure of PRes.

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

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

· sign the message M_1,
· encrypt it with the recipient (R)'s public key,
· and show the relationship between M_1 and 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.

To efficiently apply 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 to show 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.

As one can see immediately, 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 are described as *LinkedData*.

4.3 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 $D_{Pb_S}(Dualsignature)$ 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 $M_1(M_2)$ which is received by one recipient is not hashed to be $H(M_1(M_2))$ 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,
· 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:

– $CoupledData_{S,Pb_{R'}}(M_1, M_2) = \{CSC_{S,Pb_{R'},M_2}(M_1), CSig_{S,M_1}(M_2)\}$
  ◇ $CSig_{S,M_1}(M_2) = \{s_1, r_1, M_2, H(M_1)\}$
    $x_1 \in_R [1, \cdots, q-1]$
    $r_1 = H(g^{x_1}, H(M_1), H(M_2)[, etc])$
    $s_1 = \frac{x_1}{r_1 + Pv_S} \bmod q$
    On 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:

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

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

$x_1 \in_R [1, \cdots, q-1]$
$(k_3, k_4) = H({Pb_R}^{x_1} \bmod p)$
$s_1 = \frac{x_1}{r_1 + Pv_S} \bmod q$
$r_1 = KH_{k_3}(H(M_1), H(M_2)[, etc])$
$c_1 = E_{k_4}(M_1)$
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 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.

4.4 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. The six main messages in LITESET are described in Table 8

Table 8. Six Main Messages of LITESET.

message	message factor
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	$\{LinkedData_{P,Pb_M}$(AuthReqData, CapToken)$\}$
PRes	$\{Sig_M$(PResData)$\}$

For other messages, operations mentioned above are adapted similarly according to their message type. A detailed description of the messages will be given in the final version of this paper.

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 known to be secure against adaptively chosen ciphertext attacks (the most powerful attacks that one can conceive in the real world). 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 against that of SET. Here, we compare LITESET with SET based on RSA, which is the most common implementation. Elliptic cryptosystems[1] are known as a quite efficient cryptgraphical technology. But, we don't investigate them here.

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 that for $(Pb_{e_1} \cdot g^r)^{s \cdot Pv_{e_2}}$ is $\frac{7}{4} \cdot |q|$. In conventional SET, 1024-bit RSA composite is used. To achieve the same security level, $|q| = 160bits$ and $|p| = 1024bits$ should be chosen for our scheme [1]. Table 9(a) shows the costs of message generation and verification of the six main messages. We see that the computational costs are saved over 50%[2]. For other messages, Table 9(b) shows the costs of message generation and verification respectively where we can also see the significant cost reduction.

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%.

[1] *Signcryption on elliptic curves*[8] has been already proposed, and we can realize LITESET on elliptic curves easily.

[2] It is difficult to make quantitative analysis of computational costs involved in certificate verification, which heavily depends on the structure of a cerrification infrastructure employed. Thus, we don't investigate them here.

5.2 Message overhead

In our evaluation, digital signature and public key encrypted session key are regarded as message overhead. Namely, for our scheme, $r(|r| = 80bits)$, $s(|s| = 160bits)$ and hashed variables$(|H(t)| = 160bits)$ for message linking are message overhead. Table 10(a) shows the message overhead of the six main messages. We see that message overhead is saved over 70% for each message. Table 10(b) shows the message overhead of other messages; hence the reduction of message overhead is also significant.

6 Conclusion

In this paper, a new and very practical method which reduces computational cost and message overhead of SET messages is proposed by applying 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 apparent.

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*, page 165-179, 1997. Springer-Verlag.
2. MasterCard and Visa, " Secure electronic transaction (SET) specification book 1: Business Decryption", May 1997.
3. MasterCard and Visa, "Secure electronic transaction (SET) specification book 2: Programmer's Guide", May 1997.
4. Donal O'Mahony, Michael Peirce and Hitesh Tewari, "Electronic Payment Systems", Artech House Publishers, 1997
5. 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.
6. 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.
7. R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptsystems," *Communications of the ACM,* 21(2):120-128, 1978.
8. Y. Zheng and H. Imai, "Efficient Signcryption Schemes on Elliptic Curves," Proc. of IFIP SEC'98 , Chapman & Hall, Sept 1998, Vienna. (to appear)

Table 9. Computational cost for message generation/verification.

(a)main messages

message	conventional scheme	our scheme	$\frac{ourscheme}{conventionalscheme}$
PInitReq	-/-	-/-	-/-
PInitRes	384/384	240/280	0.625/0.729
PReq	768/384	480/280	0.625/0.729
AuthReq	768/1536	240/560	0.313/0.365
AuthRes	1536/768	480/280	0.313/0.365
PRes	384/384	240/280	0.625/0.729
Total	3072/3456	1440/1600	0.469/0.463

(b)other messages

message	conventional scheme	our scheme	$\frac{ourscheme}{conventionalscheme}$
AuthRevReq	768/1536	240/560	0.313/0.365
AuthRevRes	768/768	240/280	0.313/0.365
CapReq	768/768	240/280	0.313/0.365
CapRes	768/768	240/280	0.313/0.365
CapRevReq	768/768	240/280	0.313/0.365
CapRevRes	768/768	240/280	0.313/0.365
CredReq	768/768	240/280	0.313/0.365
CredRes	768/768	240/280	0.313/0.365
CredRevReq	768/768	240/280	0.313/0.365
CredRevRes	768/768	240/280	0.313/0.365
PCertReq	384/384	240/280	0.313/0.729
PCertRes	384/384	240/280	0.313/0.729
BatchAdminReq	768/768	240/280	0.313/0.365
BatchAdminRes	768/768	240/280	0.313/0.365
CardCInitReq	-/-	-/-	-/-
CardCInitRes	384/384	240/280	0.313/0.729
Me-AqCInitReq	-/-	-/-	-/-
Me-AqcInitRes	384/384	240/280	0.625/0.729
RegFormReq	384/384	240/280	0.625/0.729
RegFormRes	384/384	240/280	0.625/0.729
CertReq	768/768	240/280	0.313/0.365
CertRes	384/384	240/280	0.625/0.729
CertInqReq	384/384	240/280	0.625/0.729
CertInqRes	384/384	240/280	0.625/0.729

Table 10. Message overhead.

(a)main messages

message	conventional scheme	our scheme	$\frac{\text{ourscheme}}{\text{conventionalscheme}}$
PInitReq	-	-	-
PInitRes	1024bit	320bit	0.313
PReq	2008bit	720bit	0.359
AuthReq	4056bit	640bit	0.158
AuthRes	4256bit	480bit	0.113
PRes	1024bit	320bit	0.313
Total	12368bit	2480bit	0.201

(b)other messages

message	conventional scheme	our scheme	$\frac{\text{ourscheme}}{\text{conventionalscheme}}$
AuthRevReq	6114bit	880bit	0.144
AuthRevRes	4256bit	480bit	0.113
CapReq	2208 +(2048·n)bit	240 +(240·n)bit	≃0.12
CapRes	2048bit	240bit	0.117
CapRevReq	2208 +(2048·n)bit	240 +(240·n)bit	≃0.12
CapRevRes	2048bit	240bit	0.117
CredReq	2208 +(2048·n)bit	240 +(240·n)bit	≃0.12
CredRes	2048bit	240bit	0.117
CredRevReq	2208 +(2048·n)bit	240 +(240·n)bit	≃0.12
CredRevRes	2048bit	240bit	0.117
PCertReq	1024bit	320bit	0.313
PCertRes	1024bit	320bit	0.313
BatchAdminReq	2048bit	240bit	0.117
BatchAdminRes	2048bit	240bit	0.117
CardCInitReq	-	-	-
CardCInitRes	1024bit	320bit	0.313
Me-AqCInitReq	-	-	-
Me-AqcInitRes	2048bit	240bit	0.117
RegFormReq	1184bit	872bit	0.736
RegFormRes	1024bit	320bit	0.313
CertReq	1528bit	240bit	0.157
CertRes	1024bit	320bit	0.313
CertInqReq	1024bit	320bit	0.313
CertInqRes	1024bit	320bit	0.313