行政院國家科學委員會補助專題研究計畫 ☑成果報告 □期中進度報告

同時簽章法的匿名性設計與應用

計畫類別：☑ 個別型計畫　　□ 整合型計畫
計畫編號：NSC 96-2221-E-032 -026
執行期間：96 年 8 月 1 日至　97 年 7 月 31 日

計畫主持人：黃心嘉　淡江大學資工系　副教授
共同主持人：
計畫參與人員：馬瑞澤(淡江大學資工系研究生)、趙健宏(淡江大學資工系研究生)
蔡明璋(淡江大學資工系研究生)、鄭凱隆(淡江大學資工系研究生)
徐碧鴻(淡江大學資工系研究生)

成果報告類型(依經費核定清單規定繳交)：☑精簡報告　□完整報告

本成果報告包括以下應繳交之附件：
□赴國外出差或研習心得報告一份
□赴大陸地區出差或研習心得報告一份
☑出席國際學術會議心得報告及發表之論文各一份
□國際合作研究計畫國外研究報告書一份

處理方式：除產學合作研究計畫、提升產業技術及人才培育研究計畫、列管計畫及下列情形者外，得立即公開查詢
□涉及專利或其他智慧財產權，□一年□二年後可公開查詢

執行單位：淡江大學資工系

中　華　民　國　九十七 年八月三十一日

一、中英文摘要

中文摘要

基於個人隱私的保護，在 2005 年學者 Nguyen 首先提出非對稱式的同時簽章法，也是第一個非採用環簽章設計的同步簽章法。除了正確性、不可偽造性和公平性之外，Nguyen 的方法也滿足匿名性與不可鍊結性。為了滿足匿名性，Nguyen 的同時簽章法具有身分識別的缺失，也就是交換簽章的簽章者無法識別彼此的身分，會使攻擊者可以利用此一缺失，藉由愚弄簽章者而耗盡簽章者的計算資源。然而具有簽章者模糊性的同時簽章法就不會有此一缺失。因此為了匿名的同時簽章法，定義一個新的身分識別的特性。在此一計畫中，提出一個改良的非對稱式的同時簽章法，可以同時滿足身分識別與匿名的特性；此外也滿足不可鍊結性。身分識別性、匿名性與不可鍊結性，就可以保護同時簽章法的簽章者隱私。

關鍵字: 同時簽章、匿名性、身分識別、隱私、數位簽章法

英文摘要

For the privacy protection, Nguyen first proposed an asymmetric concurrent signature scheme without adopting ring signatures in 2005. Except correctness, unforgeability, and fairness, Nguyen's scheme satisfies two new properties: Anonymity and unlinkability. To satisfy the anonymity property, Nguyen's scheme has identification flaw that signers cannot identify each other during the exchange protocol. So an attacker makes use of this flaw to trick signers to exhaust signers' computation resources. However, the concurrent signature schemes with signer-ambiguity do not have the identification flaw. A new property, identification, is defined for the concurrent signature scheme with anonymity. In this project, an improved asymmetric concurrent scheme is proposed to provide both anonymity and identification. Our improved scheme satisfies identification, anonymity, and unlinkability at the same time. With identification, anonymity, and unlinkability, the signers' privacy is protected well without flaws.

關鍵字: Concurrent signatures, anonymity, identification, privacy, signature schemes

二、前言與研究目的

In 2004, Chen et al. first proposed the concurrent signature scheme by adopting the idea of ring signatures [2, 19]. The ring signature scheme is a signature scheme hiding the actual signer among all ring members. When the number of ring members is reduced to two, the ring signature scheme has the singer-ambiguity. For example, *A* signs a ring signature which only contains *A* and *B*'s identities. After validating *A*'s ring signature, *B* is sure that the ring signature is generated by *A* because *B* doesn't sign it. Because there are two identities involved in this ring signature, *B* can't convince the third party that this ring signature is generated by *A*. Susilo et al. [21] point out that Chen et al.'s scheme adopts the same keystone fix to generate both commitments. To cluster two ambiguous signatures with same keystone fix, the signer-ambiguity is removed. To overcome this flaw, Susilo et al. adopt different keystone fix to generate the commitments to propose their scheme, the perfect concurrent signature scheme in 2004 [21]. Wang et al.'s [22] points out that Susilo et al.'s scheme does not satisfy the fairness property in 2006. Therefore, Wang et al. proposed two iperfect concurrent signature schemes satisfying the fairness property. Wang et al.'s scheme also improve the performance. Wang et al.'s scheme satisfies four security properties: correctness, unforgeability, fairness and signer-ambiguity.

Nguyen [17] first proposed asymmetric concurrent signature schemes to provide privacy protection in 2005. Nguyen's scheme not only satisfies three basic security properties but also two additional security properties: Anonymity and unlinkability. Anonymity means that any third party cannot determine the generator of a commitment among all possible singers before the keystone is revealed. So the anonymity property is used to protect the privacy of the transactors before the keystone is released. The unlinkability property means that the relationship between two exchanged concurrent signatures does not exist, as long as the keystone is released. The unlinkability property is used to protect the privacy of the transactors after the keystone is released.

Nguyen's scheme is the first concurrent signature scheme without using ring signature schemes. However, two signature schemes with different structures are used to design Nguyen's scheme. In order to this disadvantage, Chen [8] proposed balanced concurrent signature scheme adopting the signature scheme with the same structure. But Chen's scheme does not satisfy anonymity although Chen's scheme satisfies correctness, unforgeability, fairness, and unlikability.

Anonymity is very important to protect the privacy. However, in Nguyen's scheme, the transactor cannot identify the generator of the commitments due to the anonymity property. By utilizing this disadvantage, attackers trick transactors to exhaust their computation resources. So Nguyen's scheme is vulnerable against denial-of-service attack.

The major flaw of Nguyen's scheme is that the transactor cannot identify commitments' generators. If the concurrent signature satisfies not only anonymity but also identification, this flaw can be removed. So a new property of identification is introduced. Identification means that, during the commitment exchanging protocol, both the transactors are able to identify one another, through acquisition of corroborative evidence. Due to the anonymity property, the third party cannot identify the generator of the commitments. Transactors' privacy is protected by the anonymity property. But the anonymity property in Nguyen's scheme causes identification problem. That is anonymity and identification conflict with each other. Our research goal is to design concurrent signature schemes that not only satisfy anonymity property but also have identification property at the same time. An improved Nguyen's scheme is proposed to satisfy anonymity and identification although these two properties are conflict. Moreover, the improved scheme also satisfies unlinkability and three basic security properties.

三、文獻探討

Promise of Schnorr and Promise of Schnorr-like Signatures

The Schnorr signature scheme is described below.

System Construction: This algorithm selects two large primes p and q such that $q|(p-1)$. Let g be a generator of the multiplicative subgroup of order q in Z_p*. This algorithm also selects a cryptographic hash functions h: $\{0, 1\}^* \rightarrow Z_q*$. Each participant's private key x is chosen randomly from Z_q*. The corresponding public key is $y= g^x \bmod p$.

Signature Generation: The algorithm selects a random secret integer k from Z_q*, and computes $e= h(m||g^k \bmod p)$ and $b= xe + k \bmod q$, where m is a message. The algorithm outputs a Schnorr signature (e, b) on the message m using the private key x.

Signature Verification: On the input <e, b, y, m>, this verification algorithm returns an accepting result if $e= h(m||(g^b y^{-e} \bmod p))$ holds; otherwise, it returns a rejecting result.

If (e, b) is a Schnorr signature on a message m for the public key y, then the tuple (e, β) is the promise of the signature (e, b), where $\beta= g^b \bmod p$. The equation $e= h(m||(\beta y^{-e} \bmod p))$ is used to validate the promise of Schnorr signature (e, β) on the message m by using the public key y. However, anyone is able to use the public key y to forge promises of Schnorr signatures. To prove that (e, β) is indeed a legal promises of Schnorr signature, the signer has to reveal the value

b. Then the signer converts the promise of Schnorr signature (e, β) to a Schnorr signature (e, b) to show that (e, b) is a generated by the signer.

The following shows why anyone can use the public key y to forge promises of Schnorr signatures. On a message m', anyone randomly selects a secret integer k' from Z_q^*, and computes $e'= h(m'||(y^{k'} \bmod p))$ and $\beta= y^{k'+e'} \bmod p$. Then $e'= h(m'||(\beta y^{-e'} \bmod p))$ holds. So the pair (e', β) is a forged promise of Schnorr signature on m' for the public key y.

The Schnorr-like signature scheme proposed by Nguyen [17] is described below.

System Construction: This algorithm selects two large primes p and q such that $q|(p-1)$. Let g be a generator of the multiplicative subgroup of order q in Z_p^*. This algorithm also selects a cryptographic hash functions h: $\{0, 1\}^* \rightarrow Z_q^*$. Each participant's private key x is chosen randomly from Z_q^*. The corresponding public key is $y= g^x \bmod p$.

Signature Generation: The algorithm selects a random secret integer k from Z_q^*, and computes $e= h(m||(g^k \bmod p))$ and $b= (k- e)x^{-1} \bmod q$. The algorithm outputs a Schnorr-like signature (e, b) on the message m using the private key x for the public key y.

Signature Verification: On input the tuple $<e, b, y, m>$, this algorithm returns an accepting result if the equation $e= h(m||(g^e y^b \bmod p))$ holds; otherwise, it returns a rejecting result.

If (e, b) is a Schnorr-like signature on the message m for the public key y, then the tuple (e, b_1, β) is its promise, where $\beta= y^{b-b1}= y^{b2} \bmod p$ and $b= b_1+b_2 \bmod q$. If $e= h(m||(g^e y^{b1}\beta \bmod p))$ holds, then the promise of Schnorr-like signature (e, β) is valid on the message m for the public key y. To convert the promise of Schnorr-like signature to a Schnorr-like signature, the signer reveals the value b_2. Hence $(e, b_1+b_2 \bmod q)$ is indeed a legal Schnorr-like signature that generated by the owner of the public key y.

Using the public key y, everyone can solely generate a valid promise of Schnorr-like signature at will. On the message m' from $\{0, 1\}^*$, anyone randomly selects two integers k' and b_1' from Z_q^*, and computes $e'= h(m'||(g^{k'} y^{b1'} \bmod p))$ and $\beta= g^{k'-e'} \bmod p$. Then the equation $e'= h(m'||(g^{e'} y^{b1'}\beta \bmod p))$ holds. That is the tuple (e', b_1', β) is a forged promise of Schnorr-like signature on message m' without using the private key x.

Asymmetric Concurrent Signature Scheme

Nguyen's concrete asymmetric concurrent signature scheme is described below.

SETUP: For some security parameter l as input, this algorithm selects two large primes p and q such that $q|(p-1)$. Let g be a generator of the multiplicative subgroup of order q in Z_p^*. This algorithm also selects a cryptographic hash function h: $\{0, 1\}^* \rightarrow Z_q^*$. The function *Hash* is defined to be the hash function h. The other functions is defined by $KGEN(t)= g^t \bmod p$, $KGEN_{yi}(t)= y_i^t \bmod p$, and $KTRAN(t, x_i)= t^{xi} \bmod p$. Each participant's private key x_i, $1\leq i\leq n$ is chosen randomly from Z_q^*. The corresponding public key is $y_i= g^{xi} \bmod p$. The public parameters are $<p, q, g>$ along with the descriptions of the spaces F, K, and M, where $F= Z_p^*$, $K= Z_q^*$ and $M= \{0, 1\}^*$.

ISIGN: On input the tuple $<y_i, x_i, m_i>$, this algorithm chooses a random value $r_i \in Z_q^*$ and computes three values as follows: $e_i= h(m_i||(g^{ri} \bmod p))$, $k_i= x_i e_i+ r_i \bmod q$, and $s_i= g^{ki} \bmod p$, where y_i is a public key, x_i is the private key corresponding to y_i, and the message $m_i \in M$. The algorithm outputs a promise of Schnorr signature $\sigma_i= <e_i, s_i>$ on m_i, and a keystone k_i, where $e_i, k_i \in K$, and $s_i \in F$.

MSIGN: On input the tuple $<y_j, x_j, s_j, m_j>$, this algorithm chooses a random value $r_j \in Z_q^*$ and computes two values as follows: $e_j= h(m_j||(g^{rj} s_j \bmod p))$, and $k_j= (r_j- e_j)x_j^{-1} \bmod q$, where y_j is a public key, x_j is the private key corresponding to y_j, and the message $m_j \in M$. The algorithm outputs a promise of Schnorr-like signature $\sigma_j= <e_j, k_j, s_j>$ on m_j, where $e_j, k_j \in K$, and $s_j \in F$.

IVERIFY: On the input $<\sigma_i, y_i, m_i>$, this algorithm returns an accepting result if the equation $e_i= h(m_i||(s_i y_i^{-ei} \bmod p))$ holds; otherwise, it returns a rejecting result. Here $\sigma_i= <e_i, s_i>$, $e_i \in K$, $s_i \in F$, a public key y_i, and the message $m_i \in M$.

MVERIFY: On the input <σ_j, y_j, m_j>, this algorithm returns an accepting result if the equation $e_j = h(m_j||(g^{e_j}y_j^{k_j}s_j \bmod p))$ holds; otherwise, it returns a rejecting result. Here σ_j= <e_j, k_j, s_j>, e_j, $k_j \in K$, $s_j \in F$, a public key y_j, and the message $m_j \in M$.

VERIFY: On the input <k_i, S_i> (or <k_i, S_j>), this algorithm first checks whether or not $KGEN(k_i) = s_i$ (or $KGEN_{y_j}(k_i) = s_j$), where $k_i \in K$, S_i= <σ_i, y_i, m_i>, σ_i= <e_i, s_i> (or σ_j= <e_j, k_j, s_j>), $e_i \in K$, $s_i \in F$ (or e_j, $k_j \in K$, $s_j \in F$), y_i (or y_j) is a public key, and the message m_i (or m_j)$\in M$. If $KGEN(k_i) \neq s_i$ (or $KGEN_{y_j}(k_i) \neq s_j \bmod q$), it outputs a rejecting result; otherwise, it produces its output by using the algorithm IVERIFY(S_i) (or MVERIFY(S_j)).

The asymmetric concurrent signature protocol is described in the following. Initial signer A and matching signer B run SETUP first to set the public parameters and generate their private-public key pairs. Assume that A's key pair is <x_A, y_A> and B's key pair is <x_B, y_B>.

Step 1: A generates his/her promise of Schnorr signature and a keystone k_A by performing the algorithm ISIGN on the message $m_A \in M$ as follows: <k_A, σ_A>= <k_A, <e_A, s_A>>= ISIGN(y_A, x_A, m_A). Then A sends the pair <σ_A, m_A> to B.

Step 2: B performs IVERIFY(σ_A, y_A, m_A) to verify whether or not the promise of Schnorr signature σ_A is valid after receiving A's promise of Schnorr signature σ_A on the message m_A. If IVERIFY(σ_A, y_A, m_A) outputs a rejecting result, then B aborts. Otherwise, B computes $s_B = KTRAN(s_A, x_B)$. Then B performs the algorithm MSIGN with the keystone fix s_B to generate his/her promise of Schnorr-like signature on the message $m_B \in M$ as follows: σ_B= <e_B, k_B, s_B>= MSIGN(y_B, x_B, s_B, m_B). Then B sends A the pair <σ_B, m_B>.

Step 3: A first computes $s_B = KGEN_{y_B}(k_A)$ after receiving B's promise of Schnorr-like signature σ_B on the message m_B. Then, A checks whether or not the computed s_B is equal to the keystone fix s_B from B. If they are the different, then abort. Otherwise, A validates the promise of Schnorr-like signature σ_B by performing MVERIFY(σ_B, y_B, m_B). If MVERIFY(σ_B, y_B, m_B) rejects σ_B, then A aborts; otherwise, A sends the keystone k_A to B.

四、研究方法

Nguyen's asymmetric concurrent scheme is improved to satisfying not only anonymity property but also identification property at the same time in this project. Due to the identification property, the denial-of-service attack on Nguyen's scheme is removed. However, the identification and anonymity properties are conflict. Therefore, the identification property is satisfied only for the match signer and initial singer during the signature exchange. For the other one, it is hard to identify who the match and initial signers are. To achieve this research goal, a new identification way is proposed to improve Nguyen's scheme.

五、結果與討論

In this section, our improved scheme is first proposed. Then the related analysis and discussions are given. Our improved scheme is proposed to remove the identification flaw in Nguyen's scheme. The generic algorithms of our scheme are stated first. Then the protocol and concrete scheme are given.

Generic Algorithms for Our Asymmetric Concurrent Signature Scheme

The generic algorithms used to construct our scheme are specified below.

SETUP: A probabilistic algorithm that, on input a security parameter l, outputs descriptions of the set of participants U, the message space M, the keystone space K, the keystone fix space F, the Diffie-Hellman key space D, a function $Hash$: $M\rightarrow K$ and a function $KGEN$: $K\rightarrow F$. The algorithm also outputs the private-public key pairs $\{x_i, y_i\}$ for all participants, the function families $KGEN_{y_i}$: $K\rightarrow F$, $KGEN_{y_i, y_j}$: $K\rightarrow D$, a keystone transformation function $KTRAN$: $F\times\{x_i\}\rightarrow F$, and a public reducing function $Reduce$: $F\rightarrow K$.

ISIGN: A probabilistic algorithm outputs a commitment $c_i = \langle e_i, s_i\rangle$ and a keystone k_i on the input $\langle y_i, x_i, b_1, m_i\rangle$, where y_i is a public key, x_i is the private key corresponding to y_i, $b_1, e_i, k_i\in K$, $s_i\in F$, the message $m_i\in M$, and $s_i = KGEN(k_i - b_1)$.

MSIGN: A probabilistic algorithm outputs a promise of Schnorr-like signature $\sigma_j = \langle e_j, k_j, s_j\rangle$ on the input $\langle y_j, x_j, s_j, m_j\rangle$, where y_j is a public key, x_j is the private key corresponding to y_j, $e_j, k_j\in K$, $s_j\in F$, and the message $m_j\in M$.

IVERIFY: An algorithm takes $S_i = \langle\sigma_i, y_i, m_i\rangle$ as its input and outputs an accepting or a rejecting result, where $\sigma_i = \langle e_i, s_iKGEN(b_1)\rangle$, $b_1, e_i\in K$, $s_i\in F$, y_i is a public keys, and the message $m_i\in M$.

MVERIFY: An algorithm takes $S_j = \langle\sigma_j, y_j, m_j\rangle$ as its input and outputs an accepting or a rejecting result, where $\sigma_j = \langle e_j, k_j, s_j\rangle$, $e_j, k_j\in K$, $s_j\in F$, y_j is a public keys, and the message $m_j\in M$.

VERIFY: An algorithm takes $\langle k_i, S_i\rangle$ (or $\langle k_i, S_j\rangle$) as its input, where $k_i\in K$ is a keystone and $S_i = \langle\sigma_i, y_i, m_i\rangle$, (or $S_j = \langle\sigma_j, y_j, m_j\rangle$), $\sigma_i = \langle e_i, s_iKGEN(b_1)\rangle$, (or $\sigma_j = \langle e_j, k_j, s_j\rangle$), $b_1, e_i\in K$, $s_i\in F$, $m_i\in M$ (or $e_j, k_j\in K$, $s_j\in F$, $m_j\in M$), and y_i (or y_j) is a public key. This algorithm first checks whether or not $KGEN(k_i) = s_iKGEN(b_1)$ (or $KGEN_{y_j}(k_i) = s_j$). If the equation does not hold, then the algorithm outputs a rejecting result. Otherwise, it produces its output by using the algorithm IVERIFY(S_i) (or MVERIFY(S_j)). If the algorithm VERIFY returns an accepting result, then $\langle k_i, e_i\rangle$ forms a valid signature on m_i for y_i or $\langle k_i + k_j, e_j\rangle$ forms a valid signature on m_j for y_j.

The output $c_i = \langle e_i, s_i\rangle$ of ISIGN is called a commitment while a tuple $\sigma_i = \langle e_i, s_iKGEN(b_1)\rangle$ is called a promise of a Schnorr signature. If IVERIFY(σ_i, y_i, m_i) returns an accepting result, then σ_i is a valid promise of Schnorr signature on m_i for y_i. The output $\sigma_j = \langle e_j, k_j, s_j\rangle$ of MSIGN is called a promise of Schnorr-like signature. If MVERIFY(σ_j, y_j, m_j) returns an accepting result, then σ_j is a valid promise of Schnorr-like signature on m_j for y_j. A promise of Schnorr (or Schnorr-like) signature σ_i (or σ_j) on the message m_i (or m_j) for y_i (or y_j) together with a keystone k_i is called a concurrent signature. Therefore if VERIFY(k_i, $S_i = \langle\sigma_i, y_i, m_i\rangle$) (or VERIFY($k_i$, $S_j = \langle\sigma_j, y_j, m_j\rangle$)) returns an accepting result, then the tuple $\langle k_i, \sigma_i\rangle$ (or the tuple $\langle k_i, \sigma_j\rangle$) is a valid concurrent signature on the message m_i (or the message m_j) using the public key y_i (or y_j). That is $\langle k_i, e_i\rangle$ (or $\langle k_i + k_j, e_j\rangle$) is a valid concurrent signature on the message m_i (or m_j) using the public key y_i (or y_j).

The commitment has the signer-anonymity property that any third part only guesses the identity of the real signer among n possible signers with probability $1/n$. Because the promise of Schnorr signature has the signer-anonymity property, after exposing b_1, the commitment still possesses the signer-anonymity property. After releasing the keystone, the promise signatures do not possess the signer-anonymity property anymore. So anyone utilizes keystones to bind these promise of signatures to their actual signer, and uses the algorithm VERIFY to validate concurrent signatures.

Generic Protocol for Our Asymmetric Concurrent Signature Scheme

Suppose that the initial signer A and the matching signer B run SETUP first to set the public parameters and generate their private-public key pairs. Assume that A's key pair is $\langle x_A, y_A\rangle$ and B's key pair is $\langle x_B, y_B\rangle$. Our asymmetric concurrent signature protocol works as follows:

Step 1: *A sends* her identity ID_A to B over secure **channels**.

Step 2: B randomly chooses a value $t\in K$ and computes $KGEN_{y_A, y_B}(t)$ and $r = KGEN_{y_B}(t)$, where $r\in D$. B generates $b_1 = Reduce(KGEN_{y_A, y_B}(t))$. Then A sends r to B.

Step 3: A computes $KGEN_r(x_A)$ and recovers $b_1 = Reduce(KGEN_r(x_A))$. Then A performs the algorithm ISIGN on the message $m_A \in M$ to generate his/her commitment as follows:

$<k_A, c_A>=<k_A,<e_A, s_A>>=$ ISIGN(y_A, x_A, b_1, m_A)

Then the keystone is k_A. Finally, A sends the pair $<c_A, m_A>$ to B.

Step 4: B performs IVERIFY(σ_A, y_A, m_A) to verify whether or not the promise of Schnorr signature $\sigma_A= <e_A, s_AKGEN(b_1)>$ is valid after receiving A's commitment c_A on the message m_A. If IVERIFY(σ_A, y_A, m_A) outputs a rejecting result, then B aborts. Otherwise, B computes $s_B= KTRAN(s_AKGEN(b_1), x_B)$. Then B performs the algorithm MSIGN with the keystone fix s_B to generate his/her promise of Schnorr-like signature

$\sigma_B = <e_B, k_B, s_B>=$ MSIGN(y_B, x_B, s_B, m_B)

on the message $m_B \in M$. Then B sends A the pair $<\sigma_B, m_B>$.

Step 5: A first computes $s_B= KGEN_{y_B}(k_A)$ after receiving B's promise of Schnorr-like signature σ_B on m_B. Then, A checks whether or not s_B is equal to the keystone fix s_B given by B. If the answer is not, then abort. Otherwise, A validates the promise of Schnorr-like signature σ_B by performing MVERIFY(σ_B, y_B, m_B). If MVERIFY(σ_B, y_B, m_B) rejects σ_B, then A aborts; otherwise, A sends the keystone k_A to B.

S_A is accepted by algorithm IVERIFY after exposing the secret session key b_1, where $S_A= <<e_A, s_AKGEN(b_1)>, y_A, m_A>$. Note that, these two concurrent signatures $<k_A, S_A>$ and $<k_A, S_B>$ will be "concurrently" accepted by algorithm VERIFY after the keystone k_A is revealed, where $S_A= <<e_A, s_AKGEN(b_1)>, y_A, m_A>$ and $S_B= <<e_B, k_B, s_B>, y_B, m_B>$. In other word, after revealing the keystones k_A, these two promise of Schnorr and Schnorr-like signatures $\sigma_A= <e_A, s_AKGEN(b_1)>$ and $\sigma_B= <e_B, k_B, s_B>$ bind their real signers at the same time. The reason why two concurrent signatures $<k_A, e_A>$ and $<k_A+k_B, e_B>$ become valid is that the same keystone k_A is used to generate σ_A and σ_B, where $s_A= KGEN(k_A- b_1)$ and $s_B= KGEN_{y_B}(k_A)$.

Our Concrete Asymmetric Concurrent Signature Scheme with Anonymity and Identification

Our concrete asymmetric concurrent signature scheme with anonymity and identification is described in this section. The algorithms, SETUP, ISIGN, MSIGN, IVERIFY, MVERIFY and VERIFY, are described below.

SETUP: For some security parameter l as input, this algorithm selects two large primes p and q such that $p= 2q + 1$. Let g be a generator of the multiplicative subgroup of order q in Z_p^*. This algorithm also selects a cryptographic hash function h: $\{0, 1\}^* \rightarrow Z_q^*$. The function *Hash* is defined to be the hash function h. Then $KGEN(t)= g^t \bmod p$, $KGEN_{y_i}(t)= y_i^t \bmod p$, $KGEN_{y_iy_j}(t)= y_i^{xjt} \bmod p$, and $KTRAN(t, x_i)= t^{xi} \bmod p$. The reducing function *Reduce* is defined to be $Reduce(t)= t \bmod q$. Each participant's private key x_i, $1\leq i\leq n$ is chosen randomly from Z_q^*. The corresponding public key is $y_i= g^{xi} \bmod p$. The public parameters are $<p, q, g>$ along with the descriptions of the spaces F, D, K and M, where $F= D= Z_p^*$, $K= Z_q^*$, and $M= \{0, 1\}^*$.

ISIGN: On input the tuple $<y_i, x_i, b_1, m_i>$, this algorithm chooses a random value $r_i\in Z_q^*$ and computes three values as follows: $e_i= h(m_i||(g^{ri+b1} \bmod p))$, $k_i= x_ie_i+ r_i+ b_1 \bmod q$, and $s_i= g^{xiei+ri} \bmod p$, where y_i is a public key, x_i is the private key corresponding to y_i, and the message $m_i\in M$. The algorithm outputs a keystone k_i and a commitment $c_i= <e_i, s_i>$ on m_i, where $e_i, b_1, k_i\in K$, $s_i\in F$.

MSIGN: On the input tuple $<y_j, x_j, s_j, m_j>$, this algorithm chooses a random value $r_j\in Z_q^*$ and computes two values as follows: $e_j= h(m_j||(g^{rj}s_j \bmod p))$, and $k_j= (r_j- e_j)x_j^{-1} \bmod q$, where y_j is a public key, x_j is the private key corresponding to y_j, and the message $m_j\in M$. The algorithm outputs a promise of Schnorr-like signature $\sigma_j= <e_j, k_j, s_j>$ on m_j, where $e_j, k_j\in K$, and $s_j\in F$.

IVERIFY: On the input tuple $<\sigma_i, y_i, m_i>$, this algorithm returns an accepting result if the equation $e_i= h(m_i||(s_iKGEN(b_1)y_i^{-ei} \bmod p))$ holds; otherwise, it returns a rejecting result. Here

$\sigma_i = \langle e_i, s_i KGEN(b_1) \rangle$, $e_i, b_1 \in K$, $s_i \in F$, a public key y_i, and the message $m_i \in M$.

MVERIFY: On the input $\langle \sigma_j, y_j, m_j \rangle$, this algorithm returns an accepting result if the equation $e_j = h(m_j \| (g^{e_j} y_j^{k_j} s_j \bmod p))$ holds; otherwise, it returns a rejecting result. Here $\sigma_j = \langle e_j, k_j, s_j \rangle$, $e_j, k_j \in K$, $s_j \in F$, a public key y_j, and the message $m_j \in M$.

VERIFY: On input the tuple $\langle k_i, S_i \rangle$ (or $\langle k_i, S_j \rangle$), this algorithm first checks whether or not $KGEN(k_i) = s_i KGEN(b_1)$ (or $KGEN_{y_j}(k_i) = s_j$), where $k_i \in K$, $S_i = \langle \sigma_i, y_i, m_i \rangle$, $\sigma_i = \langle e_i, s_i KGEN(b_1) \rangle$ (or $\sigma_j = \langle e_j, k_j, s_j \rangle$), $e_i, b_1 \in K$, $s_i \in F$ (or $e_j, k_j \in K$, $s_j \in F$), y_i (or y_j) is a public key, and the message m_i (or m_j) $\in M$. If $KGEN(k_i) \neq s_i KGEN(b_1)$ (or $KGEN_{y_j}(k_i) \neq s_j$), it outputs a rejecting result; otherwise, it produces its output by using the algorithm IVERIFY(S_i) (or MVERIFY(S_j)).

These algorithms together with our proposed protocol described in Section 4.2 can realize the concrete asymmetric concurrent signature scheme. After revealing the keystone k_i, the property of the signer-anonymity would be broken by checking that the equation of $KGEN(k_i) = s_i KGEN(b_1)$ or $KGEN_{y_j}(k_i) = s_j$. That is, the asymmetric concurrent signature can be verified by the algorithm VERIFY.

Performance Analysis and Discussions

In Nguyens's scheme, an attacker pays $2T_E$ to forge a valid promise of Schnorr signature to the matching signer, where T_E denotes the computational cost for one modular exponentation. Because Nguyen's scheme does not have the identification property, the matching signer totally pays $3T_E$ to confirm the promise of Schnorr signature is valid and produces the corresponding promise of Schnorr-like signature to initial signer. Hence Nguyen's protocol is vulnerable against the denial-of-service attack.

Table 1 gives the security comparison between Nguyen's scheme and our improved scheme. It is easily to find that our scheme satisfies identification property. The identification property can be used to guard against the denial-of-service attack for the matching signer's computational resource. Therefore, our scheme removes the identification flaw in Nguyen's scheme. Our scheme satisfies both anonymity and identification at the same time. By using the anonymity, unlinkability, and identification, our scheme provides a practical privacy protection for the initial and matching signer.

Table 1: Security Property Comparison between Nguyen's Scheme and Our Improvement

Schemes / Properties	Nguyen's Scheme	Our Improvement
Correctness	√	√
Unforgeability	√	√
Fairness	√	√
Signer-ambiguity	×	×
Anonymity	√	√
Unlinkability	√	√
Identification	×	√

Table 2 gives the performance comparison between Nguyen's scheme and our improved scheme. Both Nguyen's scheme and our improvement need multi-exponentiation. The multi-exponentiation computational costs for $a_1^{x_1} a_2^{x_2}$ and $a_1^{x_1} a_2^{x_2} a_3^{x_3}$ are about 1.16 T_E and 1.25T_E, respectively [4]. The computational loads of the initial signer A and matching signer B are both $4.16T_E$ and $3T_E$ in Nguyen's scheme. In our scheme, the computational loads of the initial signer A and matching signer B are $5.16T_E$ and $5.32T_E$, respectively. Our computation loads is a little larger than the loads of Nguyen's scheme by 1or 2.32 exponentiations. By paying a little computational overhead, it is valuable to provide the identification property, which can be used to guard against the denial-of-service attack.

Table 2: Performance Comparison between Nguyen's Scheme and Our Improvement

Schemes / Items	Nguyen's Scheme	Our Improvement
Computational Cost of A	$4.16E$	$5.16E$
Computational Cost of B	$3E$	$5.32E$
Computational Cost of Verifier	$2.31E$	$2.31E$
Ambiguous Signature/Commitment Size	$2\|q\|$	$2\|q\|$
Keystone Size	$\|q\|$	$\|q\|$

六、參考文獻

[1] M. Abadi, N. Glew, B. Horne, and B. Pinkas, "Certified E-mail with a Light on-line trusted third party: Design and implementation," *Proc. of the 11th International World Wide Web Conference, WWW 2002,* Honolulu, Hawaii, USA, May 7-11, 2002, pp. 387-395.

[2] M. Abe, M. Ohkubo, and K. Suzuki, "1-out-of-n signatures from a variety of keys," Advances in Cryptology - ASIACRYPT 2002, LNCS, Vol. 2501, New York: Springer-Verlag, 2002, pp. 415-432.

[3] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," *Advances in Cryptology - EUROCRYPT 1998,* LNCS, Vol. 1403, New York: Springer-Verlag, 1998, pp.591-606.

[4] G. Ateniese. "Efficient verifiable encryption (and fair exchange) of digital signature," *Proc. of AMC Conference on Computer and Communications Security (CCS'99)*, ACM Press, New York, U.S.A ., 1999, pp. 138-146.

[5] E. F. Brick*ell, D. Chaum, I.B. Damgård, and J. van de Graaf, "Gradual and Verifiable Release of a secret," Advances in Cryptology - 1987, LNCS, Vol. 293, New York: Springer-Verlag, 1987, pp.156-166.*

[6] C. Cachin and J. Camenisch, *"Optimistic Fair secure computation," Advances in Cryptology - CRYPTO 2000, LNCS, Vol. 1880, New York: Springer-Verlag, 2000, pp.94-112.*

[7] L. Chen, C. Kudla, and K. G. Paterson. "Concurrent signatures," *Advances in Cryptology – EUROCRYPT 2004*, LNCS, Vol. 3207, New York: Springer-Verlag, 2004, pp. 287-305.

[8] Y.-C. Chen, *On the research of fair exchange protocols and micropayment schemes*, Master Thesis, National Central University, Taiwan, R.O.C, 2006.

[9] R. Cleve, "Controlled gradual disclosure schemes for random bits and their applications," *Advances in Cryptology-CRYPTO 1989,* LNCS, Vol. 435, New York: Springer-Verlag, 1990, pp.573-588.

[10] I. B. Damgård, "Practical and provably secure release of a secret and exchange of signatures," *Advances in Cryptology - EUROCRYPT 1993,* LNCS, Vol. 765, New York: Springer-Verlag, 1994, pp. 200-217.

[11] S. Even, O. Goldreich, and A. Lempel, "A randomized protocol for signing contracts," Communications of the ACM, 1985, Vol. 28(6), pp.637-647,.

[12] M. K. Franlin and M. K. Reiter, "Fair exchange with a semi-trusted third party," *Proc. of the 4th ACM Conference on Computer and Communications Security,* Zurich, Switzerland,

1997, pp.1-5.

[13] M. K. Franlin and G. Tsudik, "Secure group barter: Multi-party fair exchange with semi-trusted neutral parties," *Proc. of Financial Cryptology - EUROCRYPT 1998,* LNCS, Vol. 1465, New York: Springer-Verlag, 1998, pp.90-102.

[14] J.A. Garay, M. Jakobsson, and P. MacKenzie, "Abuse-free optimistic contract signing," *Advances in Cryptology - CRYPTO 1999,* LNCS, Vol. 1666, New York: Springer-Verlag, 1999, pp.449-466.

[15] O. Goldreich, "Sending certified mail using oblivious transfer and a threshold scheme," Technical Report, Computer Science Department, Israel Institute of Technology, 1984.

[16] O. Goldreich, "A simple protocol for signing contracts," *Advances in Cryptology-CRYPTO 1983,* New York: Springer-Verlag, 1984, pp.133-136.

[17] K. Nguyen, "Asymmetric concurrent signatures," *Proc. of Information and Communications Security Conference*, ICICS 2004, LNCS, Vol. 3783, New York: Springer-Verlag, 2005, pp. 181-193.

[18] B. Pfitzmann, M. Schunter, and M. Waidner, "Optimal efficiency of optimistic contract signing," *Proc. of the 7th Annual ACM Symposium on Principles of Distributed Computing,* New York, U.S.A., 1998, pp.113-122.

[19] R. L. Rivest, A. Shamir, and Y. Tauman. "How to leak a secret," Asiacrypt' 01, LNCS, Vol. 2248, New York: Springer-Verlag, 2001, pp.552-565.

[20] C. P. Schnorr, "Efficient Identification and Signatures for Smart Cards," *Advances in Cryptology-CRYPTO 1989,* LNCS, Vol. 435, New York: Springer-Verlag,1990, pp.239-252.

[21] W. Susilo, Y. Mu, and F. Zhang, "Perfect concurrent signature schemes," *Proc. of Information and Communications Security Conference*, ICICS 2004, LNCS Vol. 3269, New York: Springer-Verlag, 2004, pp. 14-26.

[22] G. Wang, F. Bao, and J. Zhou, "The Fairness of Perfect Concurrent Signatures," *The 8th International Conference on Information and Communications Security (ICICS 2006),* LNCS, Vol. 4307, New York: Springer-Verlag, 2006, pp. 435-451.

七、計畫成果自評

In this project, our improved scheme is proposed to provide anonymity and identification at the same time. Due to Table 1, our improved scheme satisfies not only the three basic security properties but also anonymity, unlinkability and identification. Since our improved scheme satisfies anonymity and unlinkability, the privacy is protected well in the improved scheme. Even though our improved scheme satisfies anonymity, our scheme also satisfies the identification property to guard against denial of service attack on computational resource. Our improved scheme is better than Nguyen's scheme. Since our improved schemes satisfy the identification property, our schemes are more practical in the real world. The project goal is completed.