ISSN: 2278 – 1323
International Journal of Advanced Research in Computer Engineering & Technology (IJARCET)
Volume 2, No 5, May 2013

Fairness of Optimistic ID-Based Concurrent Signature Schemes

Pearl Ei Phyu

***Abstract*— Concurrent signature scheme was introduced by Chen, Kudla and Paterson in Eurocrypt 2004. In these concurrent signature schemes, two parties can produce two ambiguous signatures. These signatures bind to their true signers concurrently only when an extra piece of information (namely the keystone) is released by either of the two parties. The concept of original concurrent signature schemes is that both parties must have the true fairness in exchanging the signatures mutually. Huang, Chen and Wang improved two ID-based perfect concurrent signature schemes in 2007. However, this paper points out that their schemes are unfair because the initial signer can cheat the matching signer. Therefore, the initial signer has more advantages than the matching signer. According to this observation, we modify these schemes to achieve the true fairness of two ID-based perfect concurrent signature schemes and also propose these schemes to get the accountability property.**

***Index Terms*—Fair exchange, Concurrent signature, Bilinear pairings, accountability, ID-Based signatures.**

I. INTRODUCTION

Concurrent signatures contribute an alternative approach for the traditional problem in fair exchange of signatures. The principle of fair exchange is that both parties get the other party's item, or no party gets the other party's item at the end of a process. To achieve fair exchange over Internet, in which two parties are mutually dishonest, is an important task. In concurrent signature, both parties can produce their ambiguous signatures. Thus, any third party cannot distinguish which signature is signed by any party until one of the two parties releases the keystone publicly.

In concurrent signature, there are two parties acting in the protocol. They are known as the initial signer and the matching signer. The initial signer is the party who computes a keystone and sends the first signature to the matching signer. The matching signer is also the party who responds to the initial signature by creating another signature.

In [3], Chen, Kudla and Paterson introduced original concurrent signature schemes that used the same keystone fix in producing the ambiguous signatures. However, these schemes were unfair because only the party who can create the keystone can get more advantages over the other party.

Manuscript received May, 2013.

***Pearl Ei Phyu**, Department of Information and Communication Technology (ICT), University Technology (Yatanarpon Cyber City), Pyin Oo Lwin, Myanmar, +959402535605*

Susilo, Mu and Zhang [4] proposed perfect concurrent signatures, but the initiator could generate the two keystones independently which enable the initiator could bind different ambiguous signature (neither the one send to the matching signer) with the one created by the matching signer. Therefore, these schemes cannot provide perfect ambiguity.

To overcome these weak points, Chow and Susilo provided identity-based perfect concurrent signatures [5]. As both keystones (k_I and k_M) were produced by the initial signer, it may cause unfair. To give true fairness, Huang, Chen, Lin and R.Huang improved ID-based concurrent signature scheme by producing the keystones from both parties [7]. However, the protocol breaks down fairness in step (2) and step (3).

Moreover, ID-based perfect concurrent signatures suffered from the message substitute attack. Because the keystone fix does not contain the exchange messages (such as m_A and m_B) in their concurrent signature schemes. Therefore, we propose the modified ID-based concurrent signature schemes in order to overcome message substitute attack and to get the true fairness.

II. REVIEW OF HUANG, CHEN, LIN AND R.HUANG'S SCHEMES

A. Bilinear Pairings and Complexity Assumption

Let G_1 be a cyclic additive group generated by P with order prime q and G_2 be a cyclic multiplicative group with the same order q. A bilinear pairing is a map

$\hat{e}: G1 \times G1 \longrightarrow G_2$ with the following properties:

Bilinear: For all $P, P_1, P_2, Q, Q_1, Q_2 \in G_1$,

$\hat{e}(P_1 + P_2, Q) = \hat{e}(P_1, Q)\,\hat{e}(P_2, Q)$,

$\hat{e}(P, Q_1 + Q_2) = \hat{e}(P, Q_1)\,\hat{e}(P, Q_2)$.

Non-degenerate: There exits $P,Q \in G_1$ such that $\hat{e}(P, Q) \neq 1$;

Computable: There is an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P, Q \in G_1$.

Modified Weil pairing and Tate pairings are examples of bilinear maps.

Computational Co-Diffie-Hellman (Co-CDH) Problem: Given a randomly chosen (P_1, P_2, aP_1, bP_2), where $P_1, P_2 \in G_1$, $a, b \in Z_q^*$, and a, b are unknown, compute $abP_2 \in G_2$.

Co-CDH Assumption: For every probabilistic polynomial-time algorithm A, the advantage of A to solve Co-CDH-Problem is negligible.[7]

B. ID-Based Perfect Concurrent Signature Algorithms

– SETUP:

- Choose $(G_1, G_2, e\hat{}, q, P)$ as Section *A*. The Private Key Generator (PKG) selects a random number $s \in Z_q^*$ and sets $P_{pub} = sP$. It selects three cryptographic hash functions $H_0 : \{0, 1\}^* \rightarrow G_1$ and $H_1 : \{0, 1\}^* \rightarrow Z_q$ and $H_2 : \{0, 1\}^* \rightarrow G_1$. It publishes system parameters $params = \{G_1, G_2, e\hat{}, q, P, P_{pub}, H_0, H_1, H_2\}$, and keeps s as the master private key. The algorithm also sets $\mathcal{M} = \mathcal{F} = Z_q$, and $F = K' = G_1$.
- Sets $K_I = K_M = G_2$.
- Sets $F_I : G_2 \longrightarrow Z_q$ be a one-way permutation.
- Sets $F_M (x, y) = F_I (x) + y \pmod q$.
- Sets $Enc(k) = kP$.
- Sets $Dec(K', K'') = e\hat{}(K', K'')$.

– EXTRACT: The EXTRACT algorithm is defined as follows.

- A user $\mathcal{U}_i$ submits his or her identity ID_i to the PKG.
- The PKG generates $\mathcal{U}_i$'s private key as $S_{IDi} = sQ_{IDi}$, where $Q_{IDi} = H_0 (ID_i)$.

– ASIGN: The algorithm accepts $(ID_i, ID_j, S_{IDi}, f_i, m_i)$ and performs the following.

- Selects a random point $Z \in G_1$.
- Computes $u_0 = H_1 (H_0 (m) \| (ID_i \oplus ID_j) \| e\hat{} (Z, P) e\hat{} (f_i Q_{IDj}, P_{pub}))$.
- Computes $V = u_0^{-1}(Z – (u_0 – u_j) S_{IDi})$.
- Sets $u_i = u_0 – f_i \pmod q$, $u_j = f_i$.
- Outputs $\sigma = (u_i, u_j, V)$ as the signature on message m.

– AVERIFY: The algorithm accepts (σ, ID_i, ID_j, m), where $\sigma = (u_i, u_j, V)$, and verifies whether
$u_{i+} u_j =^? H_1(H_2(m) \| (ID_i \oplus ID_j) \| e\hat{}(V, P)^{ui + uj} e\hat{}(u_i Q_{IDi}, P_{pub}) e\hat{}(u_j Q_{IDj}, P_{pub}))$
holds with the equality. If so, then output *accept*. Otherwise, output *reject*.

– VERIFY: The algorithm accepts (k_i, k_j, S'), where $k_i \in K_I$ and $k_j \in K_M$ are the keystones and $S' = (\sigma_i, \sigma_j, ID_i, ID_j, m_i, m_j)$. The algorithm verifies whether $f_i = F_I (k_i)$, $f_j = F_I (k_j) + f_i \pmod q$. If not, then outputs *reject*. Otherwise, run AVERIFY on σ_i and σ_j respectively. If both outputs are *accept*, then outputs *accept*. Otherwise, outputs *reject*.[7]

C. ID-Based Perfect Concurrent Signature Protocol

1. Alice performs the following

– Picks a random keystone $k_I \in G_2$, computes keystone fix $f_I = F_I (k_I)$.

– Selects a message $m_I \in \mathcal{M}$, computes her ambiguous signature as

$\sigma_I = (u_I, u_M, V) \longleftarrow ASIGN (ID_I, ID_M, S_{IDI}, f_I, m_I)$.

– Sends σ_I to Bob.

2. Bob performs the following

– Verifies the signature σ_I by testing whether $AVERIFY (\sigma_I, ID_I, ID_M, m_I) = accept$. Aborts if the equation does not hold.

– Picks a random number $k \in Z_q$, computes keystone $k_M = e\hat{} (P_{pub}, Q_{IDI})^k$.

– Computes encrypted keystone $K_M = kP$.

– Computes matching keystone fix $f_M = F_I (k_M) + u_j \pmod q$.

– Selects a message $m_M \in \mathcal{M}$, and computes his ambiguous signature as

$\sigma_M = (u'_M, u'_I, V') \longleftarrow ASIGN (ID_M, ID_I, S_{IDM}, f_M, m_M)$.

– Sends σ_M and K_M to Alice.

3. Alice verifies σ_M by testing whether

– $u'_I = F_I (e\hat{} (K_M, S_{IDI})) + u_M \pmod q$

– $AVERIFY (\sigma_M, ID_M, ID_I, m_M) = accept$.

If not, then Alice aborts. Otherwise, Alice computes keystone $k_M = e\hat{} (K_M, S_{IDI})$ and releases the keystone (k_I, k_M), then both signatures are binding concurrently.[7]

In their concurrent signature protocol, Alice can cheat Bob without releasing her keystone in Step (3). In this situation, Bob can't afford to get her keystone. Therefore, their schemes are unfair. Moreover, they did not consider the exchanged messages altogether in generating the keystone fix. Therefore, their schemes may suffer from message substitute attack. Due to this observation, we propose the modified ID-based concurrent signature schemes to get the true fairness and also overcome the message substitute attack.

III. PROPOSED SCHEME

In this section, we describe a fair ID-based perfect concurrent signature scheme to achieve the true fairness by using an off-line Trusted Third Party (off-line TTP). Our scheme also prevents message substitute attack.

The protocol consists of two sub protocols, main protocol and recovery protocol, respectively. In main protocol, two parties exchange their ambiguous signature. After executing the main protocol, each party fairly gets respective signature if both of them is honest. If someone is dishonest in the exchange protocol, TTP can resolve this case by executing the recovery protocol.

Assume that m_A and m_B are the messages that Alice and Bob want to exchange.

A. Improved ID-based perfect concurrent signature algorithm

– SETUP: The same as that of the original scheme.
– EXTRACT: The same as that of the original scheme.
– ASIGN: The algorithm accepts (ID_A, ID_B, S_{IDA}, f_A, m_A) and performs the following.
  - Selects a random point $Z \in G_1$.
  - Computes $u_0 = H_1 (H_0 (m_A \oplus m_B) \| (ID_A \oplus ID_B) \| e\hat{} (Z, P) e\hat{} (f_A Q_{IDB}, P_{pub}))$.
  - Computes $V = u_0^{-1}(Z – (u_0 – u_B) S_{IDA})$.
  - Sets $u_A = u_0 – f_A \pmod q$, $u_B = f_A$.
  - Outputs $\sigma_A = (u_A, u_B, V)$ as the signature on message m_A and m_B.
– AVERIFY: The same as that of the original scheme.
– VERIFY: The same as that of the original scheme.

B. Improved ID-based Perfect Concurrent Signature Protocol (Main Protocol)

1. Alice performs the following
   – Picks a random keystone $k_A \in G_2$, and computes keystone fix $f_A = F_A (k_A \| (m_A \oplus m_B))$.
   – Selects a message $m_A \in \mathcal{M}$ and computes her ambiguous signature as
   $\sigma_A = (u_A, u_B, V) \longleftarrow$ ASIGN (ID_A, ID_B, S_{IDA}, f_A, m_A).
   – Sends $[\sigma_A, E_{sAT}(k_A \| (m_A \oplus m_B))]$ to Bob.

2. Bob performs the following
   – Verifies the signature σ_A by testing whether AVERIFY (σ_A, ID_A, ID_B, m_A) = accept. Aborts if the equation does not hold.
   – Picks a random number $k \in Z_q$, computes keystone $k_B = e\hat{} (P_{pub}, Q_{IDA})^k$.
   – Computes encrypted keystone $K_B = kP$.
   – Computes matching keystone fix $f_B = F_A (k_B \| (m_A \oplus m_B)) + u_B \pmod q$.
   – Selects a message $m_B \in \mathcal{M}$, and computes his ambiguous signature as
   $\sigma_B = (u'_B, u'_A, V') \longleftarrow$ ASIGN (ID_B, ID_A, S_{IDB}, f_B, m_B).
   – Sends σ_B and K_B to Alice.

3. Alice verifies σ_B by testing whether
   – $u'_A = F_A(e\hat{} (K_B, S_{IDA})) + u_B \pmod q$
   – AVERIFY (σ_B, ID_B, ID_A, m_B) = accept.

   If not, then Alice aborts. Otherwise, Alice computes keystone $k_B = e\hat{} (K_B, S_{IDA})$ and releases the keystone (k_A, k_B), then both signatures are binding concurrently.

C. Recovery for B

If Alice is dishonest, when she receives Bob's signature (σ_B) and K_B in Step (2), she may refuse to release the keystone pair (k_A, k_B) in Step (3). Therefore, the two ambiguous signatures cannot bind to their true signers concurrently. At that time, Bob run the following recovery protocol to ask the keystone k from TTP.

B $\longrightarrow$ TTP : $[\sigma_A, E_{sAT}(k_A \| (m_A \oplus m_B))], [\sigma_B, K_B]$
TTP $\longrightarrow$ B : k_A

D. Recovery for A

If Bob is dishonest, as soon as he receives Alice's signature (σ_A) in Step (1), he may cheat to get the keystone k from TTP. In this situation, Alice can run the following recovery protocol to ask Bob's signature and his encrypted keystone from TTP.

A $\longrightarrow$ TTP : $F_A (k_A \| (m_A \oplus m_B))$
TTP $\longrightarrow$ A : $[\sigma_B, K_B]$

E. Initialization Phase

In initialization phase, Alice and Bob must download their secret keys with respect to TTP (namely s_{AT} and s_{BT}) before they run the concurrent signature exchange protocol.

IV. Security Proofs

Fairness: Our proposed scheme satisfies the true fairness property.

Fairness Proof: There are two possible cases to prove the fairness of our proposed scheme.

Case 1: Alice is dishonest but Bob is honest.
Alice receives Bob's signature in Step (2) but she refuses to release the keystone to Bob in Step (3). In this situation, Bob can ask the help of TTP by running the recovery protocol for B. When TTP receives Bob's messages, TTP check whether all messages are valid. If all are valid, TTP decrypts $E_{sAT}(k_A \| (m_A \oplus m_B))$ and releases the keystone k_A to let the two signatures bind to their true signers concurrently. Therefore, the proposed protocol is fair.

Case 2: Alice is honest but Bob is dishonest.
Bob receives Alice's signature in Step (1) but he can cheat to get the keystone k_A from TTP without sending back his signature to Alice. In this situation, Alice can get Bob's signature and K_B from TTP by running the recovery protocol for A. Therefore, the proposed protocol is still fair.

Accountability: Our proposed scheme satisfies the accountability property. Our proposed scheme inherits the definition of accountability property.[9]
Proof: In our proposed scheme, the keystone fix contains the exchange messages. Therefore, the initial signer cannot use this keystone in other messages except the exchange messages. Furthermore, any signer could not generate

ambiguous signature for any messages other than the one send to other signers in his ambiguous signature, which could satisfy the VERIFY and AVERIFY algorithms.

Message Substitute Attack: Our proposed scheme can avoid the message substitute attack.

Proof: In original protocol, they suffered this attack because the keystone fix does not contain the exchange messages. For example, Alice is a customer and Bob is a merchant. When Alice want to buy the item (price -$100), she agrees to exchange signature with Bob. In this situation, Alice can cheat to get the item (price - $200) because Alice can produce the two signatures about these items. Because they do not compute the exchange messages in generating their keystone fix. The initiator could bind different ambiguous signature (neither the one send to the matching signer) with the one created by the matching signer. Therefore, we consider to compute the keystone fix including the exchange messages.

V. CONCLUSION

A In this paper, we present an ID-based perfect concurrent signature scheme in order to ensure that both parties can get the valid signature or neither of them receives any useful information. As both parties have the equal opportunities at the end of the exchange protocol, our scheme achieves the true fairness and also has the accountability property. Furthermore, our proposed scheme can prevent the message substitute attack.

ACKNOWLEDGMENT

The author would like to give special thanks to Dr. Aung Win, Principal of University of Technology (Yatanarpon Cyber City), Pyin Oo Lwin, Myanmar, Dr. Soe Soe Khaing, Professor, Head of ICT, University of Technology (Yatanarpon Cyber City), Pyin Oo Lwin, Myanmar and my supervisor, Dr. Khin Khat Khat Kyaw, Assistant Professor, Faculty of ICT, University of Technology (Yatanarpon Cyber City), Pyin Oo Lwin, Myanmar, for their invaluable suggestions and general guidances. The author would like to thank to all my teachers, my parents and all my colleagues.

REFERENCES

[1] Shamir, A., "Identity-base cryptosystems and signature schemes". *In Advances in Cryptology – CryptoÆ84, Lecture Notes in Computer Science*, Vol. 196. Springer-Verlag, Berlin, (1985), pp. 47–53.

[2] Asokan N., V. Shoup and M.Waidner. "Optimistic fair exchange of signatures". *In Advances in Cryptology-EUROCRYPT'98, lecture Notes in Computer Science*, vol. 1403. Springer-Verlag, Berlin, (1998), pp. 591-606.

[3] L. Chen, C.J. Kudla and K.G. Paterson, "Concurrent Signatures". *In C. Cachin and J. Camenisch (eds.), EUROCRYPT 2004, lecture Notes in Computer Science*, vol. 3027, Springer-Verlag, (2004), pp.287-305.

[4] W. Susilo, Y. Mu, and F. Zhang. "Perfect concurrent signature schemes". *In: ICICS'04, LNCS*, Vol. 3269, (2004), pp. 14-26. Springer, Belin.

[5] S.S.M. Chow and W. Susilo, "Generic construction of (identity-based) perfect concurrent signatures", *ICICS'05, LNCS*, Vol. 3783, (2005), pp. 194-206, Springer, Berlin.

[6] Wang Gui-lin, Bao Feng, Zhou Jian-ying. "The fairness of perfect concurrent signatures", *In: ICICS'06, LNCS*, Vol. 4307, (2006), pp. 435-451, Springer, Berlin.

[7] Zhenjie Huang, Kefei Chen, Xuanzhi Lin, Rufen Huang, "Analysis and Improvements of Two Identity-Based Perfect Concurrent Signature Schemes", *INFORMATICA*, Vol. 18, No. 3, (2007), 375-394@2007 Institute of Mathematics and Informatics, Vilnius.

[8] Li Yunfeng, He Dake, Lu Xianhui, "Accountability of Perfect CS", *In: IEEE, International Conference on Computer & Electircal Engineering*, (2008), pp. 773—777, IEEE Press.

[9] Wang, C.H., Chen, C.C., "Identity-Based Concurrent Signature Scheme with Improved Accountability". *In: 5th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, (2011), pp. 514—519. IEEE Press.