Design a security concurrent signature scheme applied to e-commerce

Pin-Chang Su Chia-Lieh Chang*

***Abstract*—In order that electronic medical records have the same legal effects as the entity records, electronic signature must be able to ensure the integrity, identity identification and repudiation. Although the existing electronic medical records for made, save modify are norms, the protection of privacy for patients is still weak. In particular, medical information is passed through Internet. Once the computer network is under attack, the whole privacy information will fall into the mire of the crisis. Therefore, information security increasingly national attention. Even if more and more companies to provide services through the Internet, but for services on the Internet, is still a very scruples. Especially sensitive personal data and financial information transmitted on the Internet might be stolen, malicious depredation, forgery attack and so on. It is difficult to establish security and fair transactions. Therefore fair exchange protocol make users getting information of each other in a fair way, so becomes an extensively studied topic in research related to personal information protection applications. Concurrent signatures were introduced as an alternative approach to solving the problem of fair exchange of signatures by Chen et al. in 2004. In these concurrent signature schemes, two parties can produce two ambiguous signatures. These signatures bind to their true signers concurrently only when the keystone is released by one of the parties. Zhang et al. improved a concurrent signature scheme based on identity in 2011. However, there is a security problem of identity authentication mechanism in their scheme. Therefore, we enhance identity authentication mechanism and prevent forged identity attack by a self-certified scheme. We propose a concurrent signature based on bilinear pairings and self-certified scheme.**

***Index Terms*—Bilinear pairings, Concurrent Signature, Fair Exchange, Self-Certified.**

I. INTRODUCTION

WITH the rapid development of the Internet, more and more information through network to communicate. Therefore, the existence of the network is subject to security concerns involving identity management, peer authentication, personal privacy and so forth. In the study of cryptography, the fair exchange protocol is attempt to solve this problem so that untrusted parties can exchange information fairly over the network. In exchange protocol, fairness means that at the end of the agreement, each party can obtain the expected items, or both of them do not obtain any useful information.

Manuscript received October 21, 2014.

P. C. Su is with the Department of Information Management, National Defense University, Taiwan, R.O.C (e-mail: spc.cg@msa.hinet.net).

C. L. Chang* is with the Department of Information Management, National Defense University, Taiwan, R.O.C (corresponding author to provide phone: 0982276143; e-mail:xyzheart11@yahoo.com.tw).

Many previous literatures studied the problem of fair exchange. Specifically, after the concept of optimistic fair signatures was proposed by Asokan et al. in 1998[1], numerous studies [2]-[5] have proposed fair exchange schemes that enable offline trusted third party (TTP). Chen et al. [6] observed that a fair exchange signature scheme is not required for numerous applications. They found a mechanism that enables more conflict resolution without the participation of a TTP, namely, concurrent signatures. To enhance the anonymity of the concurrent signature scheme proposed by Chen et al. [6], Susilo et al. [7] proposed perfect concurrent signatures (PCS). However, these schemes were unfair because 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 [8] consulted a PCS based on identity authentication. Subsequently, asymmetrical concurrent signatures [9], three-party concurrent signatures [10], multi-party concurrent signatures [11], the improved perfect concurrent signature [12], and fair concurrent signature scheme based on identity [13] were successively proposed. Unfortunate, previous studies [6], [7], [9], [10], [12] have found a weakness named message substitute attack in signature protocols. Either party can create multiple ambiguous signatures containing differing messages that can also be bound by the same keystone. Based on these observations, the security characteristic of accountability was suggested in literature [14]. The characteristic of accountability refers to the ability of any third party to confirm the accuracy of the signature through the VERIFY algorithm of the concurrent signature after the keystone is announced, thereby determining the uniqueness of the ambiguous signature. Because the improvements proposed in [13] did not achieve the security on demand, Zhang et al. [15] recommended including the messages that Alice and Bob were to exchange in the keystone fix to achieve accountability. However, researchers found that both parties did not perform identity authentication before communication will suffer identity forgery attack.

In the section, we will first expound a little more issues regarding concurrent signature. In consideration of this, we

import self-certification mechanism to strengthen peer identity authentication. In Section 2, we review bilinear pairings, concurrent signature scheme, and self-certified scheme. These techniques and means of security thereof are the focus of this paper and are described in Sections 3 and Section 4. Finally, Section 5 provides the research conclusion.

II. LITERATURE REVIEW

In this section, we firstly explain the concept of bilinear pairings and complexity assumption, secondly review concurrent signature algorithms. At last, we describe self-certified scheme.

A. Bilinear Pairings and Complexity Assumption

Bilinear pairings have been found to be very useful for various applications in cryptography. They were originally brought to the cryptographic community by Menezes et al. [16] with their MOV attack. This attack reduces the discrete logarithm problem on some elliptic or hyperelliptic curves to the discrete logarithm problem in a finite field. Joux [17] used the pairings to propose the first one round tripartite key agreement protocol in 2000. And then a number of cryptosystems from pairings have been proposed in cryptography. We describe pairings and the related mathematics in a more general format here.

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 $\hat{e}: G_1 \times G_1 \to G_2$ with the following properties：

(1)Bilinear：For all $P, Q, R \in G_1$,
$\hat{e}(P+Q,R) = \hat{e}(P,R) \cdot \hat{e}(Q,R)$,
$\hat{e}(P,Q+R) = \hat{e}(P,Q) \cdot \hat{e}(P,R)$.
And for all $a, b \in Z_q^*$,
$\hat{e}(aP,bQ) = \hat{e}(P,Q)^{ab} = \hat{e}(abP,Q) = \hat{e}(P,abQ)$.

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

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

(4)Computational Co-Diffie-Hellman (Co-CDH) problem: Given a randomly chosen $\{P, aP, bP\}$, where $a, b \in Z_q^*$, and a, b are unknown, compute $abP \in G_2$. For every probabilistic polynomial-time algorithm *A*, the advantage of *A* to solve Co-CDH-Problem is negligible.

B. Concurrent Signature

Concurrent signature schemes were proposed by Chen et al. [6] which allow both signing parties to produce and exchange ambiguous signatures, with third parties not learning the identity of the original signer until an additional keystone is announced by one of the two parties. Subsequently, the third party can use this information to verify the identity of the ambiguous signature signer.

Zhang et al. [15] improved a concurrent signature scheme based on identity in 2011. The concurrent signature scheme is composed of five parts: **SETUP**, **KEYGEN**, **ASIGN**, **AVERIFY**, and **VERIFY** algorithms.

(1)**SETUP**: The Key Generation Center (KGC) chooses $(G_1, G_2, P, q, \hat{e})$ as he above subsection and selects two cryptographic hash functions $H: (0,1)^* \to G_1$ and $h: (0,1)^* \to Z_q^*$. KGC selects a random number $s \in Z_q^*$ and sets $P_{Pub} = s \cdot P$ and publishes system parameters $\{G_1, G_2, P, q, \hat{e}, H, h, P_{Pub}\}$, and keeps s as the master private key. The algorithm also sets $\mathcal{M} = \mathcal{K} = \mathcal{F} = Z_q^*$.

(2)**KEYGEN**: The signer U_{I} submits his or her identity ID_{I} to KGC. KGC sets $U_{\mathrm{i}}'s$ public key $P_{\mathrm{i}} = H_{\mathrm{i}}(ID_{\mathrm{I}})$ and computes the signer's private key $s_i = s \cdot P_{\mathrm{i}}$

(3)**ASIGN**: The Asign algorithm accepts the following parameters $\{P_{\mathrm{A}}, P_{\mathrm{B}}, s_A, f, m\}$, where P_A, P_B are public keys, s_A is the private key associated with P_{A}, and $m \in \mathcal{M}$ is the message. The initial signer randomly chooses a key stone $k \in \mathcal{K}$ and $\alpha_A \in Z_q^*$. This algorithm performs the following parameter calculations:
(a) $\beta_1 = \hat{e}(P, P_{Pub})^{\alpha_A}$
(b) $\beta_2 = h(\hat{e}(P_{Pub}, P_B)^{\alpha_A})$
(c) $\mathrm{c} = m_A \oplus \beta_2$
(d) $f = h(k \parallel \beta_1 \parallel c) \in \mathcal{F}$
(e) $S = \alpha_A \cdot P_{Pub} - f \cdot x_A$
(f)Outputs $\sigma_A = \mathrm{ASIGN}\ (P_A, P_B, x_A, f, m_A)$ as the ambiguous signature.

(4)**AVERIFY**: The Averify algorithm accepts the following parameters $\{m, \sigma, P_A, P_B, P_{Pub}\}$, and checks whether the following parameter calculations holds with equality:
(a) $\beta_1 = \hat{e}(P,S)\hat{e}(P_{Pub}, P_A)^f$
(b) $\beta_2 = h(\hat{e}(S, P_B)\hat{e}(P_A, S_B)^f)$
(c) $m_A = c \oplus \beta_2$
If true, it output "accept"; otherwise, it output "reject".

(5)**VERIFY**: The Verify algorithm accepts the following parameters $\{k, m, \sigma, P_A, P_B, P_{Pub}\}$. The algorithm verifies whether the keystone k is valid. If the output is accepted, then VERIFY outputs "accept". If not, VERIFY outputs "reject."

C. Self-Certified

A sophisticated approach, first introduced by Girault [18], is called self-certified public key (SCPK), which can be regarded as an intermediate between the identity-based approaches and the traditional PKI approaches. Using a RSA cryptosystem a user chooses his or her private key, computes the corresponding public key, and gives it to a certificate authority. Then the certificate authority computes certificate parameters for the user, which satisfies a computationally unforgeable relationship with the public key and the identity of the user. A verifier can compute the public key from the identity and the certificate parameters. In 1997, Saeednia [19] successfully combined those merits with the inherency in both the ID-based and the self-certified systems. However, Wu et al. [20] showed that the original version of Saeednia's scheme [19] is not secure enough against withstanding the impersonated attack. Subsequently, Saeednia [21] further proved that it is possible to make the attack ineffective by taking additional precautions. The latter resulting model presents great loss of the merits compared to the original model and has no longer to meet the primary contribution of the self-certified notion. Tsaur [22] extended

Girault's works to ECC-based cryptosystems which are quite suitable for electronic transactions. A main problem of SCPK schemes is that they only provide implicit authentication, i.e., the validity of a SCPK is verified only after a successful communication. Another characteristic of the proposed self-certified signature scheme is based on bilinear pairings [23]. The self-certified signature scheme is composed of four parts: **KeyGen**, **Extract**, **Sign**, and **Verify**.

(1)**KeyGen**: KGC chooses $\{G_1, G_2, P, q, \hat{e}\}$ as he above subsection and selects a random number $s \in Z_q^*$ and sets $P_{Pub} = s \cdot P$ as its public key. It selects two cryptographic hash functions $H: (0,1)^* \to G_1$ and $h: (0,1)^* \to Z_q^*$. Each client given identity $ID \in (0,1)^*$, picks a random number $x_{ID} \in Z_q^*$ as its partial private key and sets $Y_{ID} = x_{ID} \cdot P$ as its partial public key. KGC publishes system parameters $\{G_1, G_2, P, q, \hat{e}, H, h, P_{Pub}\}$, and keeps s as the master private key.

(2)**EXTRACT**: Each client sends (ID, Y_{ID}) securely to KGC, after authenticating himself to KGC. KGC computes $H_{ID} = H(P_{Pub} \parallel ID \parallel Y_{ID}) \in G_1$, and sets the partial private key $d_{ID} = s \cdot H_{ID}$. Then KGC choose a random number $r_{ID} \in Z_q^*$ and computes $U = r_{ID} \cdot P, V = r_{ID} \cdot Y_{ID} + d_{ID}$. Finally KGC sends (U, V), to the client over a public channel.

The client first recovers $d_{ID} = V - x_{ID} \cdot U$. Then the client verifies d_{ID} by checking the following equations:

(a) $H_{ID} = H(P_{Pub} \parallel ID \parallel Y_{ID})$

(b) $\hat{e}(d_{ID}, P) =? \hat{e}(H_{ID}, P_{Pub})$

Thus the client obtains his actual private key (x_{ID}, d_{ID}). Hence, the certificate of the actual public key is used as the private key for signing.

(3)**SIGN**: To sign a message M, the signer A randomly chooses a integer $k \in Z_q^*$ and computes:

(a) $H_{ID} = H(P_{Pub} \parallel ID \parallel Y_{ID})$

(b) $v = \hat{e}(k \cdot H_{ID}, P)$

(c) $f = \mathrm{F}(M, v, H_{ID})$

(d) $V = k \cdot H_{ID} + f \cdot x_{ID} \cdot H_{ID} + f^2 \cdot d_{ID}$

Then the signer A sends the signature (f, V) together with the public key P_{Pub} of the certificate authority, its partial public key P_{ID} and identifier ID to a verifier B.

(4)**VERIFY**: To verify the signature (f, V), the verifier B requests (P_{Pub}, ID, Y_{ID}) and computes:

(a) $H_{ID} = H(P_{Pub} \parallel ID \parallel Y_{ID})$

(b) $v' = \hat{e}(V, \mathrm{P}) \cdot \hat{e}(-H_{ID}, f \cdot Y_{ID} + f^2 \cdot P_{Pub})$

Finally, the verifier B checks the equation:

(c) $f = \hat{e}(M, v', H_{ID})$

Because:

(d) $\hat{e}(V, \mathrm{P}) = v' \cdot \hat{e}(H_{ID}, f \cdot Y_{ID} + f^2 \cdot P_{Pub})$,

$v' = \hat{e}(V, \mathrm{P}) \cdot \hat{e}(-H_{ID}, f \cdot Y_{ID} + f^2 \cdot P_{Pub}) = v$

Thus $\mathrm{F}(M, v', H_{ID}) = \mathrm{F}(M, v, H_{ID}) = f$

Hence, if the two clients A and B follow this protocol, the verifier B will always accept the signature (f, V) and be convinced of the authenticity of the partial public key of the signer A.

III. Design of Concurrent Signature Method

We indicate that the existence of the network is subject to security concerns involving identity management, peer authentication, personal privacy and so forth. Particularly, identity management and peer authentication are flawed with unauthorized system access rights and disruption actions. In regard to this we enhance identity authentication mechanism and prevent forged identity attack by introducing of a self-certified scheme. In this section further proposes a concurrent signature based on bilinear pairings and self-certified scheme. This scheme consists of five phase: the initial phase, the key generation phase, the authentication phase, the signature sign and verify phase, and the verify phase. The overall operation sequence proposed in this study is shown in Fig. 1. Table 1 is the definitions of the given notations. Different phases are stared as follows.

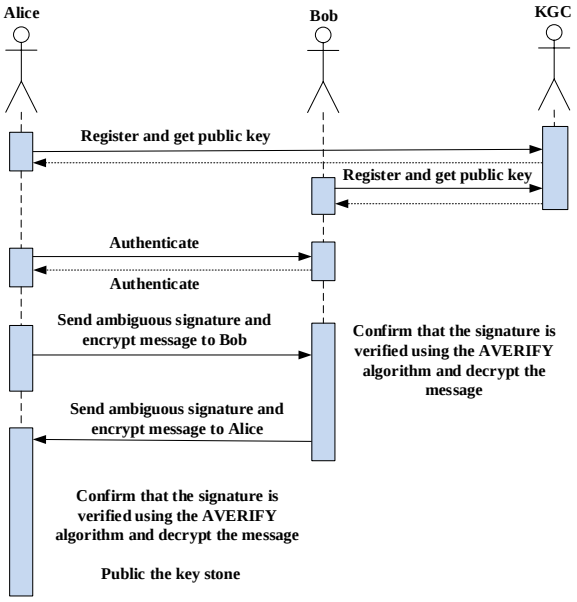

Fig 1. Diagram of the overall operation sequence

TABLE I
DESCRIPTION OF NOTATIONS USED IN THE SYSYTEM

Notation	Description
G_1	A cyclic additive group
G_2	A cyclic multiplicative group
$\hat{e}$	A bilinear pairing：$G_1 \times G_1 \to G_2$
P	Base point of the elliptical curve
q	Order of G_1 and G_2
s	The master secret key of the system
KGC	Key generation center
P_{Pub}	KGC's public key
$H()$	One way hash function: $(0,1)^* \to G_1$
$h_1()$、$h_2()$、$h_3()$	One way hash function: $(0,1)^* \to Z_q^*$.
ID_A、ID_B	The user's identity of Alice and Bob
x_{A1}、x_{A2}	Random value selected by Alice
Q_A、Q_B	Alice's and Bob's public key
D_A	Alice's partial private key
R_A	Alice's private key
α_A、α_B	Random value selected by Alice and Bob
k	Keystone
σ_A、σ_B	Ambiguous signature of Alice and Bob
m_A, m_B	Message of Alice and Bob

A. Initial Phase

The key generation center (KGC) generates G_1, G_2 of prime order q, G_1 is a cyclic additive group, G_2 is a cyclic multiplicative group, $P \in G_1$ is a generator. Let $\hat{e} : G_1 \times G_1 \to G_2$ is a bilinear pairing.

KGC selects the parameter $s \in Z_q^*$ as the master secret key of the system and computers the public key

$$P_{Pub} = s \cdot P. \quad (3\text{-}1)$$

After that KGC defines four secure one-way hash functions: $H: (0,1)^* \to G_1$ and h_1、h_2、$h_3: (0,1)^* \to Z_q^*$.

The public system parameters are $\{G_1, G_2, P, q, \hat{e}, H, h_1, h_2, h_3, P_{Pub}\}$

B. Key Generation Phase

(a)Alice ID_A selects a secure value $x_{A1} \in Z_q^*$, computes

$$Y_A = x_{A1} \cdot P \quad (3\text{-}2)$$

Alice keeps the x_{A1} secret and then sends the ID_A and Y_A to KGC.

(b)KGC computers Alice's public key H_A and partial private key D_A through the following equations:

$$Q_A = H(P_{Pub} \parallel ID_A \parallel Y_A) \quad (3\text{-}3)$$

$$D_A = s \cdot Q_A \quad (3\text{-}4)$$

(c) KGC selects a secure value $r \in Z_q^*$, and computes

$$U = r \cdot P \quad (3\text{-}5)$$

$$V = D_A + r \cdot Y_A \quad (3\text{-}6)$$

KGC sends the U and V to Alice.

(d) After receiving U and V, Alice recovers D_A by computing

$$D_A = V - x_{A1} \cdot U \quad (3\text{-}7)$$

Then Alice verifies D_A by checking following equations:

$$Q_A = H(P_{Pub} \parallel ID_A \parallel Y_A) \quad (3\text{-}8)$$

$$\hat{e}(D_A, P) =? \hat{e}(H_A, P_{Pub}) \quad (3\text{-}9)$$

If it holds, Alice receives (x_{A1}, D_A) as her actual private key.

(e) Alice randomly chooses a integer $x_{A2} \in Z_q^*$, and computes:

$$v = \hat{e}(x_{A2} \cdot Q_A, P) \quad (3\text{-}10)$$

$$w = h_1(v \parallel Q_A) \quad (3\text{-}11)$$

$$Z = x_{A2} \cdot Q_A + w \cdot x_{A1} \cdot Q_A + w^2 \cdot D_A \quad (3\text{-}12)$$

Then Alice sends the signature (w, Z) together with the public key P_{Pub} of KGC, its partial public key Y_A and identifier ID_A to the verifier Bob.

C. Authentication Phase

After Alice and Bob obtain valid identity from KGC. They could authenticate each other before transmitting messages. In order to verify the signature (w, Z), Bob requests (P_{Pub}, ID_A, Y_A) and computes

$$Q_A = H(P_{Pub} \parallel ID_A \parallel Y_A) \quad (3\text{-}13)$$

$$v' = \hat{e}(Z, P) \cdot \hat{e}(-Q_A, w \cdot Y_{ID} + w^2 \cdot P_{Pub}) \quad (3\text{-}14)$$

Finally, Bob checks the equation:

$$w = h_1(v' \parallel Q_A) \quad (3\text{-}15)$$

Because

$$\begin{aligned} \hat{e}(Z, \mathrm{P}) &= \hat{e}(x_{A2} \cdot Q_A + w \cdot x_{A1} \cdot Q_A + w^2 \cdot D_A, \mathrm{P}) \\ &= \hat{e}(x_{A2}, \mathrm{P}) \cdot \hat{e}\big((w \cdot x_{A1} + w^2 \cdot s) \cdot Q_A, \mathrm{P}\big) \\ &= v \cdot \hat{e}(Q_A, (w \cdot x_{A1} + w^2 \cdot s) \cdot \mathrm{P}) \\ &= v \cdot \hat{e}(Q_A, w \cdot Y_A + w^2 \cdot P_{Pub}) \end{aligned} \quad (3\text{-}16)$$

$$v' = \hat{e}(Z, P) \cdot \hat{e}(-Q_A, w \cdot Y_{ID} + w^2 \cdot P_{Pub}) = v \quad (3\text{-}17)$$

D. Signature Sign and Verify Phase

The initial signer Alice performs the following actions:

(a) Randomly select a parameter $\alpha_A \in Z_q^*$, a keystone $k \in K$ and message $m_A \in M$, calculate

$$\beta_1 = \hat{e}(P, P_{Pub})^{\alpha_A} \quad (3\text{-}18)$$

$$\beta_2 = h_2(\hat{e}(P_{Pub}, Q_B)^{\alpha_A}) \quad (3\text{-}19)$$

$$\mathrm{c} = m_A \oplus \beta_2 \quad (3\text{-}20)$$

$$f = h_3(k \parallel \beta_1 \parallel c) \in F \quad (3\text{-}21)$$

$$S = \alpha_A \cdot P_{Pub} - f \cdot d_A \quad (3\text{-}22)$$

(b) Perform an ambiguous signature

$$\sigma_A = ASIGN\ (Q_A, Q_B, d_A, f, m_A) \quad (3\text{-}23)$$

(c) Send σ_A, c, and S to the matching signer Bob.

After receiving σ_A, c, and S, Bob performs the following actions:

Calculate $\beta_1 = \hat{e}(P, S)\hat{e}(P_{Pub}, Q_A)^f$. (3-24)

Calculate $\beta_2 = h_2(\hat{e}(S, Q_B)\hat{e}(Q_A, S_B)^f)$ (3-25)

Calculate $m_A = c \oplus \beta_2$ (3-26)

Confirm that the signature is verified using the AVERIFY$(m_A, \sigma_A, Q_A, Q_B, P_{Pub})$. (3-27)

If the verification fails, Bob terminates the agreement. If not, Bob performs the following actions:

(f) Randomly select a parameter $\alpha_B \in Z_q^*$ and $m_B \in M$, calculates

$$\beta_3 = \hat{e}(P, P_{Pub})^{\alpha_B} \quad (3\text{-}28)$$

$$\beta_4 = h_2(\hat{e}(P_{Pub}, Q_A)^{\alpha_B}) \quad (3\text{-}29)$$

$$c' = m_B \oplus \beta_4 \quad (3\text{-}30)$$

$$f' = h_3(f \parallel \beta_3 \parallel c') \in F \quad (3\text{-}31)$$

$$S' = \alpha_B \cdot P_{Pub} - f' \cdot d_B \quad (3\text{-}32)$$

(g) Perform $\sigma_B = \mathrm{ASIGN}(Q_A, Q_B, d_B, f', m_B)$ (3-33)

(h) Send σ_B, c′, and S′ to the initial signer Alice.

(i) After receiving the signature, Alice performs the following actions:

Calculate $\beta_3 = \hat{e}(P, S')\hat{e}(P_{Pub}, Q_B)^{f'}$ (3-34)

Calculate $\beta_4 = h_2\left(\hat{e}(S', Q_A)\hat{e}(Q_B, S_A)^{f'}\right)$ (3-35)

Calculate $m_B = \mathrm{c}' \oplus \beta_4$ (3-36)

(j) Confirm that the signature is verified using the AVERIFY$(m_B, \sigma_B, Q_A, Q_B, P_{Pub})$ (3-37)

(k) Test whether $f' = h_3(f \parallel \beta_3 \parallel c')$ is true. If not, Alice terminates the protocol.

(l) If AVERIFY verification is successful, then Alice publically releases the keystone k, simultaneously binding the concurrent signature $(m_A, \sigma_A, c, S, m_B, \sigma_B, c', S')$ into effect.

E. Publically Verify Phase

After the keystone k is publically released, if $f = h_3(k \parallel \beta_1 \parallel c)$ and $f' = h_3(f \parallel \beta_3 \parallel c')$ are true, and VERIFY $(k, m_A, \sigma_A, c, S, m_B, \sigma_B, c', S')$ verification is successful, anyone can confirm that the ambiguous signatures σ_A and σ_B were signed by either Alice or Bob.

IV. Security Analyses

The security encryption mechanism proposed in this study is primarily based on the bilinear pairings, asymmetric encryption methods, concurrent signature scheme, and one-way hash function to achieve the information security management requirements. The proposed mechanism satisfies the correctness, ambiguity, unforgeability, fairness, confidentiality, non-repudiation, accountability, and self-certified approach concurrent signature security requirements recommended by Chen et al. [6] and International Organization for Standard. Security analysis are explored in the following section.

A. Correctness

According to Chen et al. [6], if the scheme passes through the ASIGN algorithm and AVERIFY = accept is true, both the message and ambiguous signature are correct.

Because (3-27) AVERIFY$(m_A, \sigma_A, P_A, P_B, P_{Pub}) =$ accept, and (3-37) AVERIFY$(m_B, \sigma_B, P_A, P_B, P_{Pub}) =$ accept is true, this study satisfies the correctness requirement.

B. Ambiguity

For ambiguous signatures under the concurrent signature scheme, third parties are unable to know the original signer of an ambiguous signature until the keystone is released by one of the two parties. Subsequently, third parties can use the released information to verify the signer of the ambiguous signature.

For the ambiguous signatures of (3-23) and (3-33) in this study, because the messages of the two transaction parties m_A and m_B are processed through the one-way no collision hash function calculation in (3-19) and (3-29), third parties cannot determine the identity of the signer; thus, signature ambiguity is achieved.

C. Unforgeability

Unforgeability refers to the mechanism that precludes data content from being maliciously tampered or altered by third parties during data transmission and ensures data integrity and accuracy at the receiving end. Regardless the number of packets intercepted during transmission, the original ciphertext or plaintext data are precluded from deciphering or recovery, thereby preventing tampering and forgery.

In this study, if Bob seeks to produce another ambiguous signature for his message $\bar{m}_B$, and the forged signature $\bar{\sigma}_B$ is bound and comes into effect with σ_A after the release of the keystone, Bob will fail. The correctness of the forged signature $(\bar{\sigma}_B, \bar{m}_B)$ signed by Bob is subject to VERIFY algorithm verification by anyone following the release of the keystone *k*.

D. Fairness

Concurrent signature fairness refers to the ability of anyone to confirm that the signatures were signed by Alice and Bob following the completion of a protocol.

This study assumes that Bob is the deceiving party. Because the messages of both parties have already been bound in the keystone fix of (3-21) $f = h(\mathrm{k} \parallel \beta_1 \parallel c)$ and (3-31) $f' = h(\mathrm{f} \parallel \beta_3 \parallel \mathrm{c}')$, the false signature signed by Bob is subject to VERIFY algorithm verification by anyone following the release of the keystone *k*.

E. Confidentiality

Confidentiality refers to the characteristic that information may not be obtained by or disclosed to unauthorized individuals, entities, or programs. After a message is successfully transmitted to the destination, all message exchanges are encrypted.

Using the method proposed in this study, transmitted messages are encrypted. Thus, if a malicious third party steals the ciphertext (c, c'), when attempting to decode (3-26) $m_A = c \oplus \beta_2$ and (3-36) $m_B = \mathrm{c}' \oplus \beta_4$ the decrypter will face the difficult Co-CDH and one-way hash function problem because the value of β_2 and β_4 must be deducted from (3-25) $\beta_2 = h(\hat{e}(S, P_B)\hat{e}(P_A, S_B)^f)$ and (3-35) $\beta_4 = h\left(\hat{e}(S', P_A)\hat{e}(P_B, S_A)^{f'}\right)$.

F. Non-repudiation

Non-repudiation refers to the evidence of an action or event that has already occurred, precluding the deniability of the action or event. In key management, users can obtain personal public and private keys for encryption and decryption operations after being verified by the KGC.

When the concurrent signature protocol is completed and the keystone k is released, any third party can use the VERIFY algorithm to confirm the identity of each signer, thereby achieving the characteristic of non-repudiation.

Algorithm	Chen et al., 2004	Wang et al., 2010	Hang et al. 2010	Zhang et al. 2011	This study
Correctness	V	V	V	V	V
Ambiguity	V	V	V	V	V
Unforgeability	V	V	X	V	V
Fairness	V	V	V	V	V
Confidentiality	X	X	V	V	V
Non-repudiation	X	X	V	V	V
Accountability	X	V	V	V	V
Identity authentication	X	X	X	X	V

TABLE II
COMPARISON OF SECURITY MECHANISMS OF THE PROPOSED SCHEME AND VARIOUS CONCURRENT SIGNATURE SCHEMES

G. Accountability

Accountability requires the signers could convince themselves and any third party that the messages signed in the ambiguous signature are the unique set generated in one protocol run. Any signer could not generate ambiguous signature for any messages other than the one he send to other signers in his ambiguous signature, which could satisfy the VERIFY and AVERIFY algorithm.

This study, we bind the messages of Alice and Bob to keystone. If anyone cheats by a fake message, it cannot pass the VERIFY and AVERIFY algorithm.

H. Identity authentication

Our scheme introduces a self-certified approach and makes it resistant against the forgery attacks. In this certification mechanism, each user obtains a valid certificate along with corresponding identity information, and holds one session key of the participants. The session key ensures users' communication against any possible attacks from the KGC collude. Messages sent between the communication users are self-certified, and hence, the certificates can be used to verify the identities applicable or not. Additionally, the proposed measure supports off-line identity. Each user can rely on KGC's public key to reliably verify the authenticity of each participant identity. It is effective to avoid untrustworthy KGC abuse of the user's secret key.

We say that a self-certified scheme is presently counterfeited against adaptive chosen message attack if no polynomial bounded adversary A has a non-negligible advantage against the challenger in the following game: The challenger takes a security parameter $r_A{}'$ and runs the setup algorithm. It gives the adversary the resulting system parameters and a public key Q_A of the certificate authority. If an attacker attempts to carry out an attack by revealing the private key from the public key of Q_A then he or she can play the role of Q_A to forge. In case of that, the attacker must solve the Co-CDH given by Q_A. Table 2 presents a summary comparison of various security mechanisms for concurrent signature schemes.

V. CONCLUSION

Because the concurrent signature schemes proposed in previous literature were all vulnerable to identity forgery attack. In this study, we applied the self-certified scheme to achieve the mutual authentication between the communication entities and proposed a new scheme for preventing such attacks and it can provide higher security in Internet where two parties are mutually dishonest. Thus, this study prevents medical records being stolen, and protect patient privacy. Unless the patient publics the keystone, or any third party impossible to know medical information.

REFERENCES

[1] N. Asokan, V. Shoup, and M Waidner, "Optimistic fair exchange of signatures," EUROCRYPT—LNCS 1403, 1998, pp. 591-606.
[2] F. Bao, R. H. Deng, and W. Mao, "Efficient and practical fair exchange protocols with off-line TTP," IEEE Symposium on Security and Privacy, 1998, pp. 77-85.
[3] J. Garay, M. Jakobsson, and P. MacKenzie, "Abuse-free optimistic contract signing," CRYPTO—LNCS 1666, 1999, pp. 449-466.
[4] N. Asokan, V. Shoup, and M. Waidner, "Optimistic fair exchange of digital signatures," IEEE Journal on SelectedAreas in Communication, vol. 18, no. 4, 2000, pp. 593-610.
[5] D. Boneh, C. Gentry, B. Lynn, and H. Shacham, "Aggregate and variably encrypted signatures from bilinear maps," EUROCRYPT—LNCS 2656, 2003, pp. 416-432.
[6] L. Chen, C. Kudla, and K.G. Paterson, "Concurrent signatures," EUROCRYPT—LNCS 3027, 2004, pp. 287-305.
[7] W. Susilo, Y. Mu, and F. Zhang, "Perfect concurrent signature schemes," ICICS, LNCS 3269, 2004, pp. 14-26.
[8] S. Chow, and W. Susilo, "Generic construction of (identity-based) perfect concurrent signatures," 7th International Conference on Information and Communications Security, LNCS 3783, 2005, pp. 194-206.
[9] K. Nguyen, "Asymmetric Concurrent Signatures," ICICS, LNCS 3783, 2005, pp. 181-193.
[10] W. Susilo, and Y. Mu, "Tripartite concurrent signatures," IFIP/SEC, 2005, pp. 425–441.
[11] D. Tonien, W. Susilo, and R.S. Naini, "Multi-party Concurrent Signatures," ISC, LNCS 4176, 2006, pp. 131-145.
[12] W.G. Lin, B. Feng, and Z.J. Ying, "The fairness of perfect concurrent signatures," ICICS, LNCS 4307, 2006, pp. 435-451.
[13] X.F. Huang, and L.C. Wang, "A fair concurrent signature scheme based on identity," 2nd International Conference on High-performance Computing and Applications LNCS 5938, 2010, pp. 198-205.
[14] L. Yunfeng, H. Dake, and L. Xianhui, "Accountability of Perfect Concurrent Signature," International Conference on Computer and Electrical Engineering, 2008, pp. 773-7.
[15] Z. Zhang, and Xu. Shuo, "Cryptanalysis and Improvement of a Concurrent Signature Scheme Based on Identity," 2nd International Conference on Software Engineering and Service Science, 2011, pp. 453-6.
[16] A. Menezes, S. Vanstone, and T, Okamoto, "Reducing elliptic curve logarithms to logarithms in a finite field," IEEE Transactions on Information Theory, vol. 39, no. 5, 1993, pp. 1639-46.
[17] A. Joux, "A one round protocol for tripartite Diffie–Hellman," 4th International symposium on algorithmic number theory, LNCS 1838, 2000, pp. 385-94.
[18] M. Girault, "Self-certified public keys," EUROCRYPT—LNCS 547, 1991, pp. 490-497.
[19] S. Saeednia, "Identity-based and self-certified key exchange protocols," Proceedings of Second Australasian Conference on Information Security and Privacy, LNCS 1270, 1997, pp. 303-313.

[20] T.C. Wu, Y.S. Chang, and T.Y. Lin, “Improvement of Saeednia’s self-certified key,” Electronics Letters, vol. 34, no. 17, 1998, pp. 1094-1095.
[21] S. Saeednia, “A note on Girault’s self-certified model,” Information Processing Letters, vol. 86, no. 6, 2003, pp. 323-327.
[22] W.J. Tsaur, “Several security schemes constructed using ECC-based self-certified public key cryptosystems,” Applied Mathematics and Computation, vol. 168, no. 1, 2005, pp. 447-464.
[23] Z. Shao, “Self-certified signature scheme from pairings,” Journal of Systems and Software, vol. 80, no. 3, 2007, pp. 388-395.