# Improved Fair Exchange Protocol Based on Signcryption-Based Concurrent Signature

Qing Ye[1,2]
[1]Information Security Center, Beijing University of Posts and Telecommunications, Beijing 100876, China
Email: yeqing1981_0@yahoo.com.cn

Hongfu Guo[2], Chunli Yang[1], Wenji Li[1]
[2]College of Computer Science and Technology, Henan Polytechnic University, Jiaozuo 454000, China
Email: {504112476, 2343212, 4543234}@qq.com

***Abstract*—Through analysis, we point out Luo et al.'s and Sun et al.'s signcryption-based concurrent signature schemes have the same defect in ambiguity and therefore the fair exchange protocols based on their schemes are not fair. Thus based on the notions of signcryption and concurrent signature, a new signcryption-based concurrent signature scheme from bilinear pairing is presented, and based on this scheme, a new fair exchange protocol is proposed. Since we adopt a new method to construct the new signcryption-based concurrent signature scheme, the new scheme redresses the flaw of Luo et al.'s and Sun et al.'s schemes, and the fair exchange protocol based on the new scheme is also fair. Besides, due to the new scheme's independence of the ring signature and simplification of encryption operations, the new scheme has the advantage of short signatures and low computation cost. We improve Luo et al.'s definition of the security model of a signcryption-based concurrent signature scheme and prove the proposed scheme and protocol are secure under the computational Diffie-Hellman assumption in the random oracle model.**

***Index Terms*—signcryption, concurrent signature, fair exchange protocol, ambiguity, bilinear pairing**

## I. Introduction

Fair exchange protocols [1-5] are mainly used to ensure the security and fairness of online transactions and they are very important. A good fair exchange protocol can make both parties exchange information in a fair way, i.e., after the completion of exchange, either each party gets the other's information or neither party does. Early fair exchange protocols are usually gradual exchange protocols [1-2], and they always take both parties many steps to finish the exchange and are very inefficient. Another class of fair exchange protocols [3-5] use a trusted third party (TTP) to achieve fairness, but since these protocols require the TTP must be fully trusted, it is sometimes hard for us to find such a TTP. In 2004, Chen et al. [6] introduced the notion of concurrent signature. In a concurrent signature scheme, two signers use a piece of information (called keystone) to make sure their signatures are exchanged in a fair way, i.e., before the keystone is released by one of the signers (initial signer), those two signatures are ambiguous, i.e., they may be issued by either of the two signers; however, after the keystone is released, both signatures are bound to their signers concurrently. Concurrent signature can realize the fair exchange of signatures without the help of a TTP and provide a new approach for designing fair exchange protocols without TTP.

But Susilo et al. [7] pointed out in Chen et al.'s [6] scheme, if both users are honest, any third party can identify who is the true signer of the ambiguous signature even before the keystone is released. In order to strengthen the ambiguity, [7] presented the concept of perfect concurrent signature and gave two concrete perfect concurrent signature schemes based on Schnorr ring signature [8] and bilinear pairing [9-10]. However, Wang et al. [11] pointed out the schemes in [7] are not fair, and actually they are more favorable to the initial signer, so Wang et al. provided two improved perfect concurrent signature schemes. Recently, Chow et al. [12] and Huang et al. [13] respectively developed ID-based perfect concurrent signature schemes, and [14] proposed a certificate-based perfect concurrent signature scheme.

However, all above concurrent signature schemes can only be used for the exchange of signatures and can't be used for physical goods transactions. Li et al. [15] first used concurrent signature [6] to design a fair exchange protocol which can be used for digital goods transactions, but their protocol is not fair and vulnerable to the attack in [11]. Chen et al. [16] used the concurrent signature scheme [11] based on Schnorr ring signature to design a truly fair exchange protocol which can be used for digital goods transactions. Li et al. [17] proposed an abuse-free fair exchange protocol based on [16]. However, there is a common defect in Chen et al.'s [16] and Li et al.'s [17] protocols, i.e., they require the communication channel between two parties should be secure, which greatly limits the application scope of the protocols. Zheng [18] first proposed the concept of signcryption which can simultaneously fulfill the functions of digital signature and public key encryption in a logically single step with a

This work was supported by National Natural Science Foundation of China (No. 61003285), the Fundamental Research Funds for the Central Universities (BUPT2012RC0219). Corresponding author: Qing Ye.

cost lower than that required by "signature followed by encryption". Based on the concept of signcryption [18-20] and the perfect concurrent signature scheme [7] from bilinear pairing, Luo et al. [21] designed a signcryption-based concurrent signature scheme, and based on this scheme, they proposed a fair exchange protocol which doesn't need secure communication channel. However, Sun et al. [22] pointed out Luo et al.'s scheme doesn't satisfy unforgeability, therefore the exchange protocol based on their scheme doesn't satisfy confidentiality and fairness. So Sun et al. proposed an improved signcryption-based concurrent signature scheme based on Luo et al.'s scheme, and they claimed the fair exchange protocol based on their proposed scheme satisfied confidentiality and fairness.

Through analysis, we point out although Sun et al. improved the unforgeability of Luo et al.'s scheme, their scheme still has the same defect in ambiguity as Luo et al.'s, therefore the fair exchange protocols based on the two schemes are both not fair. More specifically, in their schemes, as soon as user $i$ releases his keystone $k_i$, any third party can bind $i$'s ambiguous signature to $i$ and the signature receiver $j$ can decrypt $i$'s ambiguous signature, so in the fair exchange protocols based on their schemes, as soon as $i$ releases his keystone $k_i$, $j$ can obtain expected $i$'s valid signature, then it's entirely possible for $j$ to refuse to release his keystone $k_j$, and this is obviously unfair for $i$.

In this paper, based on the notions of signcryption and concurrent signature, we propose a new signcryption-based concurrent signature scheme, and based on this scheme, we present a fair exchange protocol. Compared with Luo et al.'s and Sun et al.'s schemes, our scheme has following characteristics: (1) In our scheme, $i$'s ambiguous signature will become binding if and only if both $i$ and $j$ release their keystones $k_i$, $k_j$. This means if $j$ doesn't release $k_j$, the $i$'s signature he has won't become binding, therefore $j$ is forced to release $k_j$, which avoids the occurrence of unfair phenomenon. (2) Besides, because our scheme doesn't involve the concept of ring signature which always results in long signatures and simplifies encryption operations, our scheme has the advantage of short signatures and low computation cost.

## II. Bilinear Pairing

Let $G_1$ be a cyclic additive group with order prime $q$, which is generated by $P$, and $G_2$ be a cyclic multiplicative group with the same order $q$. A bilinear pairing is a map $e$: $G_1 \times G_1 \rightarrow G_2$ with the following properties:

(1)**Bilinearity:** For all $P,Q \in G_1$, $\alpha,\beta \in Z_q^*$, there is $e(\alpha P,\beta Q) = e(P,Q)^{\alpha\beta}$.

(2)**Non-degeneracy:** There exists $P,Q \in G_1$ such that $e(P,Q) \neq 1$.

(3)**Computability:** There exists an efficient algorithm to compute $e(P,Q)$ for all $P,Q \in G_1$.

**Definition 1. Computational Diffie-Hellman (CDH) Problem.**

Given a randomly chosen $(P,aP,bP)$, where $P \in G_1$, $a,b \in Z_q^*$, and $a,b$ are unknown, compute $abP$.

**Definition 2. CDH Assumption.**

For every probabilistic polynomial-time algorithm $\mathcal{B}$, the advantage of $\mathcal{B}$ to solve CDH problem is negligible.

## III. Fair Exchange Protocol Based on Signcryption-Based Concurrent Signature

### *A. Luo et al's Signcryption-Based Concurrent Signature Scheme*

Luo et al's [21] signcryption-based concurrent signature scheme includes seven basic algorithms and they are described as follows.

(1)**Setup($k$):** The algorithm selects a bilinear map $e$:$G_1 \times G_1 \rightarrow G_2$, where $G_1$, $G_2$ are groups with order $q$ ($q>2^k$) and $G_1$ is generated by $P$. It also selects four cryptographic hash functions: $H_1$:$(G_1)^3 \rightarrow \{0,1\}^n$,$H_2$:$G_1 \rightarrow \{0,1\}^n$,$H_3$:$(G_1)^5 \times G_2 \rightarrow \{0,1\}^n$,$H_4$:$\{0,1\}^n \times (G_1)^4 \rightarrow \{0,1\}^n$. It publishes system parameters $S$ = $<G_1,G_2,e, q,P,H_1,H_2,H_3,H_4>$.

(2)**UserKeyGen($S$):**User $i$ selects a random number $x_i \in Z_q^*$ and sets $X_i = x_iP$. $i$'s secret key is $x_i$, and $i$'s public key is $X_i$.

(3)**Signcrypt($X_i$,$X_j$,$x_i$,$m_i$):**This algorithm first uses the public keys ($X_i$,$X_j$) and the signer's secret key $x_i$ to encrypt message $m_i$, and then signs on the ciphertext of $m_i$. The detailed process is as follows.

This algorithm first selects a random number $k \in Z_q^*$, and then computes $R_1=kP$, $h_1=H_1(X_i,X_j,R_1)$, keystone $k_i=x_ih_1X_i$, $h_i=H_2(k_i)$, $R_2=h_iX_i - h_1X_j$, $u=e(h_iX_i,X_j)^k$, session key $key=H_3(X_i,X_j,R_1,R_2,x_iX_j,u)$, ciphertext $y=key \oplus m_i$, $h_4=H_4(y,X_i,X_j,R_1,R_2)$ and $\sigma=k^{-1}h_4(h_1+h_i)X_i$. Finally, it outputs an ambiguous signature $C_i=(y,\sigma,R_1,R_2)$.

(4)**AVerify($X_i$,$X_j$,$C_i$):** The algorithm performs the following to verify an ambiguous signature $C_i$.

It first computes $h_1=H_1(X_i,X_j,R_1)$, $h_4=H_4(y,X_i,X_j,R_1,R_2)$ and then verifies whether $e(R_1,\sigma) = e(h_4P, R_2+h_1X_i+h_1X_j)$. If not, it outputs "$\perp$". Otherwise, it accepts the signature.

(5)**KSVerify($X_i$,$X_j$,$k_i$,$k_j$):** The algorithm is used to verify the validity of a keystone pair ($k_i$,$k_j$) after users $i$,$j$ release $k_i$, $k_j$ respectively. For user $i$, it verifies whether $e(k_i,P) = e(h_1X_i, X_i)$. If not, it shows $i$ is dishonest. Otherwise, it runs algorithm Verify. As for user $j$, the verification process is similar.

(6)**Verify($X_i$,$X_j$,$k_i$,$k_j$,$C_i$,$C_j$):** This algorithm is used to bind the ambiguous signatures $C_i$,$C_j$ to their true signers after the keystones $k_i$,$k_j$ are released. For each ambiguous signature $C_i$, the algorithm verifies whether $e(R_1,\sigma) = e(h_4P, h_1X_i+h_iX_i)$. If $C_i$,$C_j$ are bound to $i$, $j$ respectively, it outputs "accept". Otherwise, it outputs "$\perp$".

(7)**Decrypt($X_i$,$X_j$,$k_i$,$C_i$,$x_j$):** This algorithm uses the signature receiver's secret key $x_j$ and the signature signer's keystone $k_i$ to decrypt the ambiguous signature $C_i$. It first computes $u=e(R_1,x_jh_iX_i)$, $key$ = $H_3(X_i,X_j,R_1,R_2,x_jX_i,u)$, and then obtains the message $m_i=y \oplus key$.

### *B. Luo et al's Fair Exchange Protocol*

Based on above scheme, Luo et al. [21] designed a fair exchange protocol which doesn't need secure communication channel. Suppose $A$ is a buyer, $B$ is a seller, and $A$, $B$ both have an account in a bank. The protocol is briefly described as follows.

**Step 1.** $A\rightarrow B$:$C_A$=($y_A$,$\sigma_A$,$R_{A1}$,$R_{A2}$).

**Step 2.** $B\rightarrow A$:$C_B$=($y_B$,$\sigma_B$,$R_{B1}$,$R_{B2}$).

**Step 3.** $A\rightarrow B$:$k_A$.

**Step 4.** $B\rightarrow A$:$k_B$.

In above protocol, $y_A$ denotes the ciphertext of $A$'s electronic cheque $m_A$ whose value is equal to the price of the digital goods, $C_A$ denotes $A$'s ambiguous signature on $y_A$, $y_B$ denotes the ciphertext of the receipt $m_B$ generated by $B$, $C_B$ denotes $B$'s ambiguous signature on $y_B$, and $k_A$,$k_B$ denote $A$'s and $B$'s keystones respectively. For more details, please refer to [21].

### C. Sun et al.'s Signcryption-Based Concurrent Signature Scheme

Sun et al.'s [22] improved signcryption-based concurrent signature scheme still contains seven basic algorithms, and they are briefly described as follows.

**Setup:** Compared with Luo et al.'s Setup algorithm, it only adds a hash function $H_5$:$G_1\rightarrow\{0,1\}^n$.

**Signcrypt:** Compared with Luo et al.'s Signcrypt algorithm, it adds the computation of $h_i'=H_5(k_i)$ prior to computing $u$, and it modifies $u=e(h_iX_i,X_j)^k$ into $u=e(h_i'X_i,X_j)^k$.

**Decrypt:** After $i$ releases his keystone $k_i$, the signature receiver $j$ computes $h_i'=H_5(k_i)$, $u=e(R_1,x_jh_i'X_i)$, session key $key=H_3(X_i,X_j,R_1,R_2,x_jX_i,u)$ and obtains the message $m_i=key\oplus y$.

The other four algorithms are the same as Luo et al.'s.

## IV. Security Analysis on Luo et. al.'s and Sun et al's Schemes

We found Luo et al.'s [21] and Sun et al.'s [22] signcryption-based concurrent signature schemes have the same defect in ambiguity, therefore the fair exchange protocols based on their schemes are not fair. More specifically, in their schemes, as soon as user $i$ releases his keystone $k_i$, any third party can bind $i$'s ambiguous signature to $i$ and the signature receiver $j$ can decrypt $i$'s ambiguous signature, so in the fair exchange protocols based on their schemes, as soon as the buyer $A$ releases his keystone $k_A$, the seller $B$ can obtain $A$'s valid signature on the electronic cheque, then it's entirely possible for $B$ to refuse to release his keystone $k_B$ and quit the protocol. At this moment, $B$ has $A$'s valid signature on the expected electronic cheque, but $A$ doesn't have $B$'s valid signature on the corresponding receipt and can't decrypt the encrypted digital goods. Obviously, this is not fair to $A$.

If we want the signcryption-based concurrent signature scheme will not cause above fairness problem, we should relate the ambiguity of $i$'s signature not to the release of $k_i$ but to the release of ($k_i$,$k_j$).

## V. New Signcryption-Based Concurrent Signature Scheme

### A. Security Notions

Luo et al. [21] require a secure signcryption-based concurrent signature scheme should satisfy confidentiality, unforgeability, forward security, temporary key security, and they defined unforgeability in detail in [21]. But we think their definition (Luo et al. in fact require an outside attacker who doesn't know either of $A$'s and $B$'s secret keys can't forge a signature which can pass the verification of algorithm AVerify) of unforgeability isn't proper, and the unforgeability for a signcryption-based concurrent signature scheme should be defined as the improbability of an outside attacker forging a signature which can pass the verification of algorithm Verify, and it is formally redefined as follows.

**Definition 3.** We say a signcryption-based concurrent signature scheme is existentially unforgeable under a chosen message attack if the probability of success of any polynomially bounded adversary in the following game is negligible.

**Game 1:**

**Setup.** The challenger $C$ runs algorithm Setup and gives the system parameter $S$ to the adversary $E$.

**Query.** $C$ answers $E$'s various queries as follows.

(1)**Public key query.** $E$ submits an identity $u$, $C$ looks for the tuple $<u,X_u,x_u>$ in a secret key list which is called $L$-list. If it does exist, $C$ returns $X_u$ to $E$. Otherwise, $C$ runs algorithm UserKeyGen for $u$, generates $u$'s public key and secret key ($X_u$,$x_u$), and adds the tuple $<u,X_u,x_u>$ to $L$-list and returns $X_u$ to $E$.

(2)**Secret key query.** $E$ submits an identity $u$, $C$ looks for the tuple $<u,X_u,x_u>$ in $L$-list. If it exists, $C$ returns $x_u$ to $E$. Otherwise, $C$ runs algorithm UserKeyGen for $u$, generates $u$'s public key and secret key($X_u$,$x_u$), and adds the tuple $<u,X_u,x_u>$ to $L$-list and returns $x_u$ to $E$.

(3)**Hash query.** $C$ keeps a $H_i$-list for every random oracle $H_i$. When $E$ queries the random oracle $H_i$, $C$ looks for the corresponding tuple in $H_i$-list. If it exists, $C$ returns the corresponding hash value to $E$. Otherwise, $C$ chooses a random number $t$ from $H_i$'s hash space, adds the input and $t$ to $H_i$-list and returns $t$ to $E$.

(4)**Signcryption query.** $E$ submits an identity $u$, a message $m$, a session key $key$ and a keystone fix pair ($f_1$,$f_2$), $C$ first conducts a secret key query for $u$, then runs algorithm Signcrypt with an input ($m$,$x_u$,$key$,$f_1$,$f_2$) and sends the output to $E$.

**Forge.** Suppose $E$ outputs $i$'s signature ($c_i$,$v_i$) with a keystone pair ($k_1$,$k_2$), and $E$ didn't conduct a secret key query for $i$, and $E$'s output results are not produced by the signcryption oracle. We say $E$ wins the game if Verify($c_i$,$v_i$,$k_1$,$k_2$,$X_i$)=accept.

### B. Proposed Scheme

To improve the security of previous signcryption-based concurrent signature schemes, we use bilinear pairing to reconstruct a signcryption-based concurrent signature scheme. It contains six basic algorithms and they are described as follows.

(1)**Setup($k$):** The algorithm selects a bilinear map $e$:$G_1\times G_1\rightarrow G_2$, where $G_1$, $G_2$ are groups with order $q$ ($q>2^k$) and $G_1$ is generated by $P$. It sets the message space $\mathcal{M}=\{0,1\}^n$, the keystone space $\mathcal{K}=\{0,1\}^*$, the keystone fix space $\mathcal{F}=Z_q$. It also selects four cryptographic hash functions: $H_1:\{0,1\}^*\rightarrow Z_q$, $H_2:(G_1)^2\rightarrow\{0,1\}^n$,$H_3:\{0,1\}^n\times(Z_q)^2\rightarrow G_1$,$H_4:G_1\rightarrow Z_q$. It publishes system parameters $S=<G_1,G_2,e,q,n,P,H_1,H_2,H_3,H_4>$.

(2)**UserKeyGen(*S*).** User $i$ selects a random number $x_i\in Z_q^*$ and computes $X_i=x_iP$. $i$'s secret key is $x_i$, and $i$'s public key is $X_i$.

(3)**Signcrypt($m_i$,$x_i$,$key_i$,$f_i$,$f_j$).** This algorithm first uses the signer $i$'s session key $key_i$ to encrypt message $m_i$, and then uses $i$'s secret key $x_i$ and a keystone fix pair $(f_i,f_j)$ to sign the ciphertext of $m_i$. It finally outputs an ambiguous signature $(c_i,v_i)$, where

$c_i=m_i\oplus key_i$,

$v_i=x_iH_3(c_i,f_i,f_j)$.

(4)**Decrypt($c_j$,$v_j$,$key_j$).** The algorithm uses the signer $j$'s session key $key_j$ to decrypt $j$'s ambiguous signature $(c_j,v_j)$, and it outputs message $m_j$, where

$m_j=c_j\oplus key_j$.

(5)**Insiderverify($c_j$,$v_j$,$f_i$,$f_j$,$X_j$).** The algorithm uses the keystone fix pair $(f_i,f_j)$ to verify whether the ambiguous signature $(c_j,v_j)$ is correctly generated by user $j$ as follows.

If $e(P,v_j)=e(X_j,H_3[c_j,f_i,f_j])$, it outputs "accept". Otherwise, it outputs "reject".

(6)**Verify($c_i$,$v_i$,$c_j$,$v_j$,$k_i$,$k_j$,$X_i$,$X_j$).** After the keystone $(k_i,k_j)$ is released, anyone can use this algorithm to bind the signatures $(c_i,v_i)$,$(c_j,v_j)$ to their true signers as follows.

It verifies whether $e(P,v_i)=e(X_i,H_3[c_i,H_1(k_i),H_1(k_j)])$ and $e(P,v_j)=e(X_j,H_3[c_j,H_1(k_i),H_1(k_j)])$. If both the equations hold, it outputs "accept". Otherwise, it outputs "reject".

### *C. Unforgeability Analysis on the Proposed Scheme*

In this section, we prove the unforgeability of the proposed signcryption-based concurrent signature scheme in detail. As for the analyses of confidentiality, forward security and temporary key security, please refer to Section 6.

**Theorem 1.** If there exists a polynomially bounded adversary $E$ who successfully performs an existential forgery under a chosen message attack against the proposed scheme in Section 4.2 with a non-negligible probability $\varepsilon$, then there exists another algorithm $C$ which can solve the CDH problem in group $G_1$ with probability at least $[1-1/q_k]^{q_s}\cdot 1/q_k\cdot\varepsilon$, where $q_k$ denotes $E$ can make at most $q_k$ public key queries and $q_s$ denotes $E$ can make at most $q_s$ signcryption queries.

**Proof.** We will show given $(P,xP,yP)$, $C$ can figure out $xyP$ by simulating the challenger in Game 1 and interacting with forger $E$. $C$ performs as follows.

$C$ starts by running algorithm Setup and gives system parameters $S$ to $E$, and then $C$ establishes $L$-list, $H_1$-list, $H_2$-list, $H_3$-list, $H_4$-list. Next, $C$ answers $E$'s various queries as follows.

(1)**Public key query.** $E$ submits an identity $u$, $C$ looks for the corresponding public key $X_u$ in $L$-list. If it exists, $C$ returns $X_u$ to $E$. Otherwise, $C$ performs as follows.

①If $u\neq i$, $C$ randomly chooses $u$'s secret key $x_u\in Z_q^*$, computes $u$'s public key $X_u=x_uP$, and adds the tuple $<u,X_u,x_u>$ to $L$-list and returns $X_u$ to $E$.

②Otherwise, $C$ randomly chooses $s\in Z_q^*$, computes $u$'s public key $X_u=sxP$, adds the tuple $<u,X_u,\perp>$ to $L$-list and returns $X_u$ to $E$.

(2)**Secret key query.** Suppose $E$ has conducted the corresponding public key query before $E$ conducts a secret key query. If the identity $E$ submits is $u=i$, $C$ returns "$\perp$". Otherwise, $C$ looks for the corresponding tuple $<u,X_u,x_u>$ in $L$-list and returns $x_u$ to $E$.

(3)**Hash Query.** When $E$ conducts a $H_1$ query, if the query already appears in some tuple in $H_1$-list, $C$ finds it and outputs corresponding $t$. Otherwise, $C$ randomly chooses $t\in Z_q$, adds the input and $t$ to $H_1$-list and returns $t$ to $E$. When $E$ conducts $H_2$, $H_4$ queries, $C$ performs similarly. When $E$ conducts a $H_3$ query with an input $(c,f_1,f_2)$, where $c\in\{0,1\}^n$, $f_1,f_2\in Z_q$, $C$ performs as follows.

①If $(c,f_1,f_2)$ is already in some tuple $<c,f_1,f_2,t,p>$ in $H_3$-list, $C$ checks the value of $p$. If $p$=0, $C$ outputs $tyP$. If $p$=1, $C$ outputs $tP$. If $(c,f_1,f_2)$ doesn't exist, continue.

②$C$ generates a random coin $p\in\{0,1\}$ so that $\Pr[p=0]=1/q_k$.

③$C$ picks a random number $t\in Z_q^*$. If $p$=0, $C$ outputs $tyP$. If $p$=1, $C$ outputs $tP$. $C$ adds the tuple $<c,f_1,f_2,t,p>$ to $H_3$-list.

(4)**Signcryption query.** Since $E$ can request the secret key for user $u\neq i$, $E$ can signcrypt any message on behalf of user $u\neq i$. In addition, $E$ can ask $C$ for user $i$'s signcryption on any message, and $C$ responds as follows.

When $E$ conducts a signcryption query with an input $(m,key,f_1,f_2)$, $C$ first computes $c=m\oplus key$, and then conducts a $H_3$ query with an input $(c,f_1,f_2)$. Suppose the tuple $C$ obtains through the $H_3$ query is $<c,f_1,f_2,t,p>$. If $p$=0, $C$ aborts. If $p$=1, $C$ outputs a signature $(c,v)$, where $v=tX_i$.

Suppose $E$ outputs $i$'s signature $(c_i,v_i)$ with a keystone pair $(k_1,k_2)$ after $E$ conducted above queries, and $E$'s output results are not produced by the signcryption oracle, and the equation $e(P,v_i)=e(X_i,H_3[c_i,H_1(k_1),H_1(k_2)])$ holds.

Since the output of $H_3$ is totally random, the adversary must utilize previous signcryption queries or have directly queried random oracle $H_3$ with an input $(c_i,H_1(k_1),H_1(k_2))$. Suppose $E$ has conducted a signcryption query with an input $(m',key',f_1',f_2')$, where $f_j'=H_1(k_j')$, and due to this query, $C$ adds a tuple $<c_i',f_1',f_2',t,p>$ to $H_3$-list, where $c_i'=m'\oplus key'$. Since $H_3$ is an one-way function, it's impossible for $E$ to find another tuple $(c_i,k_1,k_2)$ such that $H_3(c_i,H_1(k_1),H_1(k_2))=t$. So we can conclude that $E$'s success in forging $i$'s signature stems from not previous signcryption queries but a direct $H_3$ query with the input $(c_i,H_1(k_1),H_1(k_2))$. Suppose this $H_3$ query corresponds to the tuple $<c_i,f_1,f_2,t_i,p_i>$ in $H_3$-list. If $p_i$=1, $C$ aborts. If $p_i$=0, it means $H_3($ $(c_i,H_1(k_1),H_1(k_2))$ $)=t_iyP$, in addition, since

$X_i = sxP$, the equation $e(P,v_i) = e(X_i, H_3[c_i, H_1(k_1), H_1(k_2)])$ is equivalent to $e(P,v_i) = e(sxP, t_i yP)$. Since for $C$, $s,t_i$ is known, $C$ can calculate $xyP=(st_i)^{-1}v_i$, which solves the CDH problem in group $G_1$.

Next, we calculate the probability of $C$'s success. First, we define three events:

$\eta_1$: $E$'s signcryption queries doesn't cause $C$ to stop.

$\eta_2$: $E$ successfully forges $i$'s signature with a keystone pair, and the forgery results aren't the output of the signcryption oracle.

$\eta_3$: $\eta_2$ occurs, and $p_i$=0.

$C$'s success means above three events occur simultaneously, so the probability of $C$'s success is $\Pr[\eta_1 \wedge \eta_2 \wedge \eta_3]=\Pr[\eta_1]\Pr[\eta_2|\eta_1]\Pr[\eta_3|\eta_2 \wedge \eta_1]$.

**Proposition 1.** The probability that $C$ does not abort as a result of $E$'s signcryption queries is at least $[1-1/q_k]^{q_s}$. Hence, $\Pr[\eta_1]\geq[1-1/q_k]^{q_s}$.

**Proof.** We assume $E$ conducted $n(n\in[1,q_s])$ signcryption queries. Let $\xi_j(j\in[1,n])$ represents the event that $E$'s $j$'th signcryption query doesn't cause $C$ to stop. Then the event that $C$ doesn't stop after $E$ conducted $n$ signcryption queries is equivalent to the event that $\xi_1,\xi_2,\ldots,\xi_n$ occur simultaneously, and the probability that it occurs is equal to $\Pr(\xi_1 \wedge \xi_2 \wedge \ldots \wedge \xi_n)$ $=\Pr(\xi_1)\Pr(\xi_2|\xi_1)\ldots\Pr(\xi_n|\xi_1 \wedge \xi_2 \wedge \ldots \wedge \xi_{n-1})$.

We assume $E$ doesn't make the same signcryption query twice, so $\xi_1,\xi_2,\ldots,\xi_n$ are independent events and $\Pr(\xi_1 \wedge \xi_2 \wedge \ldots \wedge \xi_n)=\Pr(\xi_1)\ \Pr(\xi_2)\ \ldots\ \Pr(\xi_n) = \Pr[p_1=1]$ $\Pr[p_2=1]\ldots\Pr[p_n=1]=[1-1/q_k]^n$. Suppose we allow $E$ can make at most $q_s$ signcryption queries, then the probability that $C$ does not abort as a result of $E$'s signcryption queries is at least $[1-1/q_k]^{q_s}$.

**Proposition 2.** If algorithm $C$ does not abort as a result of $E$'s queries then algorithm $E$'s view is identical to its view in the real attack. Hence, $\Pr[\eta_2|\eta_1]\geq\varepsilon$.

**Proposition 3.** The probability that algorithm $C$ does not abort after $E$ performs a successful forgery is $1/q_k$. Hence, $\Pr[\eta_3|\eta_2 \wedge \eta_1]=1/q_k$.

**Proof.** We have analyzed that if $E$ can perform a successful forgery, he must have conducted a $H_3$ query with an input $(c_i,H_1(k_1), H_1(k_2))$. Suppose this query corresponds to the tuple $<c_i,f_1,f_2,t_i,p_i>$ in $H_3$-list, then the probability that $C$ does not abort after $E$ performs a successful forgery is equal to the probability of $p_i$=0, i.e., $1/q_k$.

To sum up, the probability of $C$'s success is $\Pr[\eta_1 \wedge \eta_2 \wedge \eta_3]=\Pr[\eta_1]\Pr[\eta_2|\eta_1]\Pr[\eta_3|\eta_2 \wedge \eta_1]\geq$ $[1-1/q_k]^{q_s} \bullet \varepsilon \bullet 1/q_k$.

## VI. New Fair Exchange Protocol

Based on the proposed scheme in Section 4.2, we design a fair exchange protocol which doesn't need security channel and could be used to exchange digital goods. Suppose $A$ is a buyer, $B$ is a seller, and they both have an account in a bank (the bank is only an ordinary bank and doesn't have the function of a trusted third party). The protocol is described as follows.

(1)$A$ first selects a digital goods (such as an audio or video file) to buy on $B$'s website and generates the corresponding order. Then $B$ selects his keystone $k_B\in\mathcal{K}$ and a random number $s_B\in Z_q^*$, computes $f_B=H_1(k_B)$, $K_B=s_BX_A$, $r_B=f_B+H_4(K_B)$, $S_B=s_BP$, and sends $(r_B,S_B)$ to $A$.

(2)Upon receiving $(r_B,S_B)$, $A$ first selects his keystone $k_A\in\mathcal{K}$ and a random number $s_A\in Z_q^*$, computes $K=x_AX_B$, $K_A=s_AX_B$, $key_A=H_2(K,K_A)$, $f_A=H_1(k_A)$, $K_B=x_AS_B$, $f_B=r_B-H_4(K_B)$, and generates an electronic check $m_A$ whose value is equal to the price of the digital goods. $A$ then runs $(c_A,v_A)\leftarrow$Signcrypt $(m_A,x_A,key_A,f_A, f_B)$, calculates $r_A=f_A+H_4(K_A)$, $S_A=s_AP$, and sends $(c_A,v_A,r_A,S_A)$ to $B$.

(3)Upon receiving $(c_A,v_A,r_A,S_A)$, $B$ computes $K=x_BX_A$, $K_A=x_BS_A$, $key_A=H_2(K,K_A)$, runs $m_A$=Decrypt$(c_A,v_A,key_A)$, and checks the validity of $m_A$. If the electronic cheque $m_A$ is invalid, $B$ aborts. Otherwise, $B$ computes $f_A=r_A-H_4(K_A)$, and checks whether Insiderverify $(c_A,v_A,f_B,f_A,X_A)$=accept. If not, $B$ aborts. Otherwise, $B$ computes $key_B=H_2(K,K_B)$, and generates the corresponding receipt $m_B$, runs $(c_B,v_B)$=Signcrypt $(m_B,x_B,key_B,f_B,f_A)$, and sends $(c_B,v_B)$ to $A$. Meanwhile, $B$ encrypts the digital goods specified in $A$'s order with key $k=key_B||k_B$, and uploads the encrypted digital goods to the website specified in $A$'s order.

(4)Upon receiving $(c_B,v_B)$, $A$ computes $key_B=H_2(K,K_B)$, runs $m_B$=Decrypt$(c_B,v_B,key_B)$, and checks the validity of $m_B$. If the receipt $m_B$ is not correct, $A$ aborts. Otherwise, $A$ checks whether Insiderverify $(c_B,v_B,f_A,f_B,X_B)$=accept. If not, $A$ aborts. Otherwise, $A$ releases $k_A$.

(5)$B$ checks whether $f_A=H_1(k_A)$. If not, $B$ aborts. Otherwise, $B$ releases $k_B$.

At this moment, $A$ can download the encrypted digital goods and decrypt the goods with key $k=key_B||k_B$. Meanwhile, anyone can run algorithm Verify to bind the ambiguous signatures $(c_A,v_A)$, $(c_B,v_B)$ to $A$, $B$ respectively.

## VII. Performance Analysis

In this section, we provide the security and efficiency analysis on the proposed protocol in detail.

### A. Security Analysis

**Effectiveness.** If $A$, $B$ both perform according to the protocol, then when the protocol is finished, both keystones have been released and both signatures become binding. So the seller $B$ can submit $A$'s binding signature to the bank to get $A$'s payment, and the buyer $A$ can obtain $B$'s key (which can be used to decrypt the digital goods) and the corresponding receipt.

**Confidentiality.** Suppose an outside attacker $E$ (other than $A$ and $B$) intercepts $i$'s ambiguous signature $(c_i,v_i)$ on message $m_i$, it's infeasible for $E$ to get the message $m_i$. Because $E$ doesn't know $s_i$, $x_i$ and $x_j$, when $E$ calculates $s_iX_j$ or $x_jS_i$, he will face the CDH problem in group $G_1$, so it's infeasible for $E$ to calculate the session key $key_i=(x_iX_j,s_iX_j)$, and therefore the message $m_i$ can't be acquired. So the proposed protocol satisfies confidentiality.

**Forward security.** Suppose an outside attacker $E$ (other than $A$ and $B$) has obtained $i$'s $r$'th session key $key_r=H_2(x_iX_j,s_{ir}X_j)$ and $i$'s secret key $x_i$, it's infeasible for $E$ to calculate $i$'s $t$'th ($t\neq r$) session key $key_t=H_2(x_iX_j,s_{it}X_j)$. That is because when $E$ computes $s_{it}X_j$, he'll face the CDH problem in group $G_1$. Therefore, the proposed protocol satisfies forward security.

**Temporary key security.** Suppose an outside attacker $E$ (other than $A$ and $B$) has obtained $i$'s ambiguous signature $(c_i,v_i)$ on message $m_i$ and $i$'s temporary key $s_i$, it's infeasible for $E$ to calculate the session key $key_i=(x_iX_j,s_iX_j)$. This is because when $E$ calculates $x_iX_j$, he will encounter the CDH problem in group $G_1$. Therefore, the proposed protocol also satisfies temporary key security.

**Non-repudiation.** Before the keystone $(k_A,k_B)$ is released, $(c_i,v_i)$ is ambiguous and there is no way for a third party to identify who is its true signer, but as soon as $(k_A,k_B)$ is released, anyone can bind $(c_i,v_i)$ to its true signer by running algorithm Verify. Therefore, the proposed protocol satisfies non-repudiation.

**Ambiguity.** Obviously, in the proposed protocol, only when the keystone $(k_A,k_B)$ or the keystone fix $(f_A,f_B)$ is acquired, can one run algorithm Verify or Insiderverify to bind $(c_i,v_i)$ to user $i$. Before $(k_A,k_B)$ is released, an outside attacker $E$ only can intercept $(r_i,S_i)$. However, since $E$ doesn't have the signature receiver $j$'s secret key $x_j$ or the signature signer $i$'s temporary key $s_i$, $E$ can't figure out $f_i$ from $(r_i,S_i)$. So before $(k_A,k_B)$ is released, the outside attacker has no way to bind $(c_i,v_i)$ to its true signer. Therefore, the proposed protocol satisfies ambiguity.

**Fairness.** Suppose the seller $A$ is dishonest, the buyer $B$ is honest. $A$ has two chances to cheat $B$. In step (2), if $A$ doesn't send the signature $(c_A,v_A)$, then the transaction does not exist and there is no loss for both sides. If $A$ sends a false signature, then as soon as $B$ receives the signature, he will do a series of examinations. When he finds the message isn't correct or the signature is invalid, $B$ will quit the protocol and $A$ can't make a profit. In step (4), if $A$ refuses to release $k_A$ or releases a false $k_A$, $B$ won't release his keystone $k_B$, then $A$ can't decrypt the digital goods and the signature $(c_B,v_B)$ $A$ has is still an ambiguous signature.

Suppose the seller $B$ is dishonest, the buyer $A$ is honest. $B$ also has two chances to cheat $A$. In step (3), if $B$ doesn't send $(c_B,v_B)$, A won't release his keystone $k_A$, then the signature $(c_A,v_A)$ $B$ has is still an ambiguous signature, and $B$ can't obtain $A$'s payment. If $B$ sends a false signature, after $A$ receives the signature, he will do a series of examinations, when he finds the message isn't correct or the signature is invalid, $A$ will quit the protocol, and $B$ can't make a profit. In step (5), if $B$ doesn't release $k_B$ or releases a false $k_B$, the bank can't bind $(c_A,v_A)$ to $A$, and $B$ can't obtain $A$'s payment.

## *B. Efficiency Analysis*

Table 1 gives the efficiency comparison for Luo et al's [21] protocol, Sun et al's [22] protocol and the proposed protocol. As the main computational overheads, we only consider pairings (denote by P), exponentiations (denote by E), scalar multiplications (denote by S), and hash-to-point operations (denote by H). As for communication cost, we calculate the total length of data exchanged between $A$ and $B$. Suppose the communication cost of $(y_i,\sigma_i,R_{i1},R_{i2})$ or $(c_i,v_i,r_i,S_i)$ is 4, and the communication cost of $k_i$ is 1.

TABLE I.
EFFICIENCY COMPARISON

| Protocol | Comp.Cost of $A$ | Comp.Cost of $B$ | Comp.Cost of Verifier | Communi-cation Cost |
|---|---|---|---|---|
| Protocol in [21] | 6P+E+13S | 6P+E+13S | 4P+6S | 10 |
| Protocol in [22] | 6P+E+13S | 6P+E+13S | 4P+6S | 10 |
| Our Protocol | 2P+2H+5S | 2P+2H+5S | 4P+2H | 10 |

From table 1, we can see that the proposed protocol greatly reduces the computational cost of each party in the protocol while keeping the communication cost unchanged. Therefore, it is an efficient fair exchange protocol.

## VIII. CONCLUSION

After detailed analysis on Luo et al.'s and Sun et al.'s signcryption-based concurrent signature schemes, we point out their schemes have the same defect in ambiguity and therefore the fair exchange protocols based on their schemes can't achieve fairness. Thus we reconstructed a signcryption-based concurrent signature scheme which redresses the flaw of Luo et al.'s and Sun et al.'s schemes, and based on this scheme, a new fair exchange protocol which can be used for exchanging digital goods and doesn't need secure communication channel was designed. Analysis results show the new protocol not only satisfies effectiveness, confidentiality, non-repudiation, ambiguity and fairness, but also is more efficient than Luo et al.'s and Sun et al.'s protocols.

## REFERENCES

[1] M. Blum, "How to exchange(secret)keys", *ACM Transactions on Computer Systems*, vol. 1, no. 2, pp. 175-193, 1983.

[2] S. Even, O. Goldreich, A. Lempel, "A randomized protocol for signing contracts", *Communications of the ACM*, vol. 28, no. 6, pp. 637-647, 1985.

[3] M.K. Franklin, M.K. Reiter, "Fair exchange with a semi-trusted third party", in *the 4th ACM Conference on Computer and Communications Security*, ACM Press,1997, pp. 1-5.

[4] N. Zhang, Q Shi, "Achieving non-repudiation of receipt", *The Computer Journal*, vol. 39, no. 10, pp. 844-853, 1996.

[5] G. Draper-Gil, J. Zhou, J.L. Ferrer-Gomila, et al., "An optimistic fair exchange protocol with active intermediaries", *International Journal of Information Security*, online first articles, pp. 1-20, 2013.

[6] L. Chen,C. Kudla,K.G. Paterson, "Concurrent signature", in *Eurocrypt 2004*, Berlin: Springer-Verlag, 2004, pp. 287-305.

[7] W. Susilo, Y. Mu, F. Zhang, "Perfect concurrent signature schemes", in *ICICS '04*, LNCS 3269, Springer-Verlag, 2004, pp. 14–26.

[8] M. Abe, M. Ohkubo, K. Suzuki, "1-out-of-n signatures from a variety of keys", in *Advances in Cryptology - Asiacrypt*

*2002*, LNCS 2501, 2002, pp. 415 – 432.
[9] X.G. Cheng, S.J. Zhou, L.F. Guo, "An ID-based short group signature scheme", *Journal of software*, vol. 8, no. 3, pp. 554-559, 2013.
[10]J. Qin, H.W. Zhao, "*k* out of *n* oblivious transfer protocols from bilinear pairings", *Journal of software*, vol. 5, no. 1, pp. 65-72, 2009.
[11]G. Wang, F. Bao, J. Zhou, "The fairness of perfect concurrent signature", in *ICICS '06*, LNCS 4307, Springer-Verlag, 2006, pp. 435–451.
[12]S. Chow, W. Susilo, "Generic construction of (identity-based) perfect concurrent signature", in *ICICS '05*, LNCS 3783, Springer-Verlag, 2005, pp. 194-206.
[13]Z.J. Huang, K.F. Chen, Y.M. Wang, "Analysis and improvements of two identity-based perfect concurrent signature schemes", *Informatica*, vol. 18, no. 3, pp. 375-394, 2007.
[14]Z.J. Huang, J.N. Chen, "Certificate-based perfect concurrent signature", in *Proc. of the 2010 International Conference on Multimedia Information Networking and Security*, ACM, 2010, pp. 526-530.
[15]H.P. Li, W.D. Kou, X.Z. Du, "Fair e-commerce protocols without a third party", in *ISCC 2006*, Cagliari, Italy: 2006, pp. 324-327.
[16]G.H. Chen, S.H. Qing, Z.F. Qi, et a1., "Novel fair exchange protocol based on concurrent signature", *Journal on Communications*, vol. 29, no. 7, pp. 39-43, 2008.
[17]Y.F. Li, D. He, X.H. Lu, "Abuse-free fair exchange protocol without TTP", *Application Research of Computers*, vol. 26, no. 8, pp. 3053-3055, 2009.
[18]Y. Zheng, "Digital signcryption or how to achieve cost (signature & encryption)<<cost (signature) + cost (encryption)", in *CRYPTO'97*. Springer-Verlag, Berlin: 1997, pp.165-179.
[19]W. Yuan, L. Hu, H.T. Li, et al., "Cryptanalysis and improvement of an ID-based threshold signcryption scheme", *Journal of Computers*, vol. 7, no. 6, pp. 1345-1352, 2012.
[20]W. Yuan, L. Hu, H.T. Li, et al., "Analysis and enhancement of three identity-based signcryption protocols", *Journal of Computers*, vol. 7, no. 4, pp. 1006-1013, 2012.
[21]M. Luo, C.H. Zou, J. Hu, et al., "Signcryption- based fair exchange protocol", *Journal on Communications*, vol. 31, no. 8, pp. 87-93, 2010.
[22]Y.B. Shun, Y. Shun, C. Zhao, et al., "Security analysis on the improvement of a signcryption-based fair exchange protocol", *Journal of Beijing University of Posts and Telecommunications*, vol. 34, no. 6, pp. 38-41, 2011.

**Qing Ye** was born in Liaoning Province, China in 1981. She received the B.S. degree from Northeastern University (China) in 2003 and M.S. degree from Kunming University of Science and Technology (China) in 2008. She is currently pursuing the Ph.D. degree at Beijing University of Posts and Telecommunications (China). Her research is focused on information security and cryptography.

**Hongfu Guo** was born in Gansu Province, China in 1984. He received the B.S. and M.S. degree from Beijing Institute of Technology (China) in 2007, 2012 respectively. He is currently a lecturer in Henan Polytechnic University (China). His research is focused on fluid mechanics.