Fair Exchange Signature Schemes

Jingwei Liu[1], Rong Sun[2], Weidong Kou[2], and Xinmei Wang[2]

1. The Key Laboratory of Computer Network and Information Security, Ministry of Education, Xidian University, P.O. Box 119, 710071 Xi'an, China.
2. State Key Laboratory of Integrated Service Networks, Xidian University, P.O. Box 119, 710071 Xi'an, China.
jwliu@mail.xidian.edu.cn, rong_sun@hotmail.com, kou_weidong@yahoo.com.cn, xmwang@mail.xidian.edu.cn

Abstract. In this paper we propose a new class of Fair Exchange Signature Scheme(FESS) that allows two players to exchange digital signatures in a fair way. Our signature scheme is a general idea and has various implementations on most of the existing signature schemes, thus it may also be considered as an interesting extension of concurrent signature presented in EUROCRYPT 2004 that is constructed from ring signatures. In our scheme, two unwakened signatures signed separately by two participants can be verified easily by the other player, but it would not go into effect until an extra piece of commitment *keystone* is released by one of the players. Once the *keystone* revealed, two signatures are both aroused and become effective. A key feature of the proposed scheme is that two players can exchange digital signatures simultaneously through a secret commitment *keystone* without involvement of any Trusted Third Party. Moreover, the efficiency of our signature scheme is higher than that of concurrent signature.

Keywords: FESS, Concurrent Signature, Schnorr Signature, Fair Exchange, Electronic Commerce

1 Introduction

The widely use of open networks such as the Internet provides a stability foundation for electronic commerce, which usually involves two distrusted parties exchanging their items from each other, for instance, e-commerce payment protocols, electronic contract signing, and certified e-mail delivery. Due to the rapid growth of electronic commerce nowadays, fair exchange turns out to be an increasingly important topic. A digital exchange problem is deemed to be fair if at the end of exchange, either each party receives the expected item or neither party receives it. In general scenarios, digital items' exchanges have to be carried over open networks and both participants may not trust each other. There could be subsequent disputes about what was exchanged during a transaction even if the exchange itself was completed fairly. In this case, evidence should be accumulated during the exchange to enable the settlement of any future disputes.

In the recent years various schemes on the fair exchange problem have been proposed and reported in the literature. These schemes often fall into three categories:

Some solutions on fair exchange problem is often gradual exchange protocols [1–7] where two parties have to be too interactive and cumbersome for exchanging digital items by many steps. Nevertheless, these methods cannot provide fairness fully, because at the end of protocols, one player often has an advantage of one more bit than the other player does. In [3], the authors introduce timed commitments. A timed commitment is a commitment scheme in which there is an optional forced opening phase enabling the receiver to recover (with effort) the committed value without the help of the committer. But this method is only considered for Rabin and RSA signatures of a special kind. In [7], the authors show how to achieve timed fair exchange of digital signatures of standard type. Their construction follows the gradual release paradigm, and works on a new "time" structure that is called a mirrored time-line. But the length of it lead to another apparent problem, which is making sure that the underlying sequence has a period large enough so that cycling is not observed.

Recently, researches on fair exchange protocols mainly exploit an on-line or off-line Trusted Third Party (TTP) [8–21], which is involved in protocols run (on-line) or Account Opening and Disputes (off-line). So either on-line or off-line TTP may cause the bottleneck problem and at least inefficiencies in the operation though its involvement is further reduced in [8, 12, 20]. In [8], the authors introduce a new protocol that allows two players to exchange digital signatures over the Internet in a fair way. The protocol relies on a trusted third party, but is "optimistic," in that the third party is only needed in cases where one player attempts to cheat or simply crashes. The key feature of the protocol is that a player can always force a timely and fair termination, without the cooperation of the other player.

The latest direction of fair exchange is supposed to overleap TTP in the protocols. Two participants carry out digital items' exchanges using special signatures. By this means, some new fair exchange protocols are designed, which become more efficient than prior art does. The concept of concurrent signatures was introduced by Chen, Kudla and Paterson in Eurocrypt 2004 [22]. Such signature schemes allow two parties to produce and exchange two ambiguous signatures until an extra piece of information (called keystone) is released by one of the parties, which exploiting the ambiguity property enjoyed by the ring signatures [23, 24]. More specifically, before the keystone is released, those two signatures are ambiguous with respect to the identity of the signing party, i.e., they may be issued either by two parties together or just by one party alone; after the keystone is publicly known, however, both signatures are bound to their true signers concurrently, i.e., any third party can validate who signed which signature. Concurrent signature allows to build a fair exchange protocol which allow two parties to interact to exchange digital items without the involvement of the trusted third party. But it is at the sacrifice that the initial party controls the keystone and therefore he has an extra power to decide when the keystone is

released or whether it is at all. In ICICS 2004, Susilo, Mu and Zhang [29] further proposed perfect concurrent signatures to strengthen the ambiguity of concurrent signatures. That is, even if the both signers are known having issued one of the two ambiguous signatures, any third party is still unable to deduce who signed which signature, different from Chen et al.s scheme. But in [30], Wang et al. point out that Susilo et al.s two perfect concurrent signature schemes are actually not concurrent signatures and present an effective way to avoid this attack.

1.1 Our Contributions

Lots of the previous works for fair exchange are not fully suitable for the applications on open networks, because most fair exchange applications have to be provided not only security but also efficiency. In our opinion, an ideal solution for fair exchange should be both secure and efficient. With the opinion, we propose a new class of Fair Exchange Signature Schemes (FESS) in this paper that allows two players to exchange digital signatures over open computer and communications networks (such as the Internet) in a fair way, so that either each player gets the other's signature at the same time, or neither player does. In our scheme, each unwakened signature signed separately by two participants can be verified easily by the other player, but it does not go into effect until an extra piece of information *keystone* is released by one of the players. Once the *keystone* revealed, two signatures are both aroused and become effective. This new signature scheme can be applied to different application environments through various implementations. We will also introduce how to construct an implementation of FESS without a TTP. Thus it provides a valid primitive that is of interest in designing fair exchange schemes. Of course there might be other applications to consider. A key feature of FESS is that two players can exchange digital signatures simultaneously without involvement of any TTP. Although FESS do not overcome the weakness of concurrent signature, which is the initial party controls the keystone, but it really has higher efficiency than concurrent signature does.

The main contributions in this paper are listed as follows:

1. Propose a generic definition of FESS.
2. Demonstrate how to construct FESS without TTP.
3. Provide the security proof of FESS in the random oracle model.

The rest of the paper is organized as follows. We first introduce the basic definitions of FESS in the following section. The basic models for FESS are described in detail in Section 3. In Section 4, we construct an implementation of the FESS. In Section 5, we show the properties discussion and security analysis of our scheme and provide efficiency comparison between our scheme and concurrent signature. Finally, the concluding remarks and future researches are given in Section 6.

2 Basic Definitions

In this section, we introduce some basic definitions of the FESS. The parameters involved in our schemes are depicted in the following.

- a plaintext message space M: a set of strings over some alphabet.
- a keystone message space K_{ks}: a set of strings over some alphabet.
- a keystone fix space K: a set of possible keystone fix volume.
- a signature space S: a set of possible signatures.
- a signing key space X: a set of possible keys for signature creation.
- a verification key space Y: a set of possible keys for signature verification.

Definition 1. *A full FESS consists of five procedures (Parameter Setup, KGen, Sign, SVerify, KVerify)*:

- an efficient probabilistic algorithm **Parameter Setup**:

$$k \rightarrow \langle \{x_i\}, \{y_i\}, \texttt{description of}\{M, K_{ks}, K, S\}\rangle, \tag{1}$$

where k is a security parameter, $x_i \in X$ and $y_i \in Y$.

- an efficient one-way function **KGen**: $K_{ks} \rightarrow K$, which generates a keystone fix $k \in K$ with the input of a secret $keystone \in K_{ks}$. Secure hash functions can be used as **KGen** .
- an efficient probabilistic signing algorithm **Sign**: $M \times K \times X \rightarrow S$, for any message $m \in M$, keystone fix $k \in K$ and private key $x \in X$, we denote by $s \leftarrow \mathbf{Sign}_x(m, k)$ where $s \in S$.
- an efficient signature verification algorithm **SVerify**: $M \times K \times S \times Y \rightarrow \{\text{True, False}\}$, for any $m \in M$, $k \in K$, and $y \in Y$, it is necessary

$$\mathbf{SVerify}_y(m, k, s) = \begin{cases} \text{True, if } s = \mathbf{Sign}_x(m, k) \\ \text{False, if } s \neq \mathbf{Sign}_x(m, k) \end{cases} \tag{2}$$

- an efficient keystone verification algorithm **KVerify**: $M \times K \times S \times Y \times K_{ks} \rightarrow \{\text{True, False}\}$, for any $m \in M$, $k \in K$, $y \in Y$ and $keystone \in K_{ks}$, it is necessary

$$\mathbf{KVerify}_y(m, k, s, keystone) = \begin{cases} \text{True, if } \mathbf{SVerify}_y(m, k, s) = \text{True} \\ \quad \text{and } k = \mathbf{KGen}(keystone) \\ \text{False, if } \mathbf{SVerify}_y(m, k, s) = \text{False} \\ \quad \text{or } k \neq \mathbf{KGen}(keystone) \end{cases} \tag{3}$$

3 Basic Models

3.1 Fair Exchange Signature Protocols

In the normal case, most of fair exchange schemes often involve an on-line or off-line third party, but, for an embedded commitment, it is achieved in our scheme only between two participants, without loss of generality, Alice (initial signer) and Bob (respond signer). Alice who initiates the protocol first generates a piece of secret information – *keystone* randomly, signs a message with her private key

and a keystone fix built by a one-way function with input of *keystone* and sends the signature message to Bob. Bob responds this message by signing another message with his private key and the same keystone fix.

Following *Definition 1*, the detailed implement process of FESS is depicted as follows.

Alice and Bob first choose an efficient signature scheme and the relevant parameters. Let $x_A, x_B \in X$ denote Alice and Bob's private key separately and $y_A, y_B \in Y$ is the public key corresponding to the private key of two participants.

1. Alice chooses a $keystone \in K_{ks}$ randomly and computes $k = \mathbf{KGen}(keystone)$, where $k \in K$. And she takes k and her private key x_A to sign a signature $s_A = \mathbf{Sign}_{x_A}(m_A, k)$ on a message m_A agreed with Bob. The verifiable signature message is $\sigma_A = \langle m_A, k, s_A \rangle$ that should be sent to Bob.

2. After receiving Alice's verifiable signature message σ_A, Bob verifies the message σ_A using algorithm **SVerify** described in section 2. If $\mathbf{SVerify}_{y_A}(\sigma_A) =$ True, Bob chooses a message m_B agreed with Alice and takes k and his private key x_B to sign a signature $s_B = \mathbf{Sign}_{x_B}(m_B, k)$. Bob sends the verifiable signature message $\sigma_B = \langle m_B, k, s_B \rangle$ back to Alice. Otherwise, if $\mathbf{SVerify}_{y_A}(\sigma_A) =$ False, Bob aborts. Note that Bob uses the same value k as Alice does.

3. After receiving Bob's verifiable signature message σ_B, Alice verifies the message σ_B also using algorithm **SVerify**. If $\mathbf{SVerify}_{y_B}(\sigma_B) =$ True, Alice release *keystone* to arouse not only σ_B but also σ_A, thus two verifiable signatures go into effect at the same moment. If $\mathbf{SVerify}_{y_B}(\sigma_B) =$ False, Alice aborts.

4. Everyone can verify whether$\mathbf{KVerify}_{y_A}(\sigma_A, keystone) =$ True or $\mathbf{KVerify}_{y_B}(\sigma_B, keystone) =$ True.

Here we need to point out that the FESS provides fairness through the dormancy property, which is different from the ambiguous property of concurrent signature [22]. As a useful cryptographic tool, our scheme provides a primitive to build efficient fair exchange and contract signing protocols. In the next section, we will give an example of it.

3.2 Attack Model for FESS

For a secure signature scheme, the property of secure against existential forgery on adaptively chosen message attack is necessary. In this model [27, 28], an adversary wins the game if he outputs a valid pair of a message and a signature, where he is allowed to ask the signer to sign any message except the output. Here we will introduce an attack model for FESS, similarly to [28]. We say that a FESS, which consists of five algorithms: *Parameter Setup, KGen, Sign, SVerify, KVerify* , is secure against existential forgery on adaptively chosen message if no polynomial time algorithm $\mathcal{A}$ has a non-negligible advantage against a challenger $\mathcal{S}$ in the following game:

1. $\mathcal{S}$ runs Parameter Setup algorithm firstly and gives the public system parameters to $\mathcal{A}$.

2. $\mathcal{A}$ can require the following queries:

(a) Hash function query. $\mathcal{S}$ computes the value of the hash function for the requested input and sends the value to $\mathcal{A}$.

(b) *KGen* query. $\mathcal{A}$ can request that $\mathcal{S}$ selects a $keystone \in K_{ks}$ and returns fix $k = \mathbf{KGen}(keystone)$.

(c) *KReveal* query. $\mathcal{A}$ can request keystone of any keystone fix $k \in K$ produced by a previous KGen query.

(d) *Sign* query. Given a message $m \in M$ and a $k \in K$, $\mathcal{S}$ returns a signature s which is obtained by running Sign algorithm.

3. $\mathcal{A}$ outputs $\langle m, k, s\rangle$, where m is a message, k is a keystone fix and s is a signature, such that $\langle m, k\rangle$ are not equal to the inputs of any query to Sign and k is a previous output of KGen query and a previous input of KReveal query. $\mathcal{A}$ wins the game if s is a valid signature of $\langle m, k\rangle$.

Using this attack model, we can reduce the security of keystone signature scheme to the hardness of discrete logarithm problem in section 5.

4 An Implementation of FESS

In this section, we give an implementation of FESS actually by Schnorr signature. We give the system parameters firstly.

Parameter Settings:

- System parameters: Let p and q be two large primes and $q\,|p-1$. The notation g denotes an element of order q of Z_p^*.
- Alice: Alice has a pair of keys (x_A, y_A) for Schnorr signature where x_A is Alice's private key , y_A is her public key and $y_A = g^{x_A} \bmod p$.
- Bob: Bob has a pair of keys (x_B, y_B) for Schnorr signature where x_B is Bob's private key , y_B is his public key and $y_B = g^{x_B} \bmod p$.

1. Alice chooses $keystone = \langle ID_{AB}\rangle$ and computes $k = G(ID_{AB})$, where ID_{AB} is some random information about Alice and Bob's identity and G is a hash function. Alice generates her signatures$s_A = \mathbf{Sign}_{x_A}(m_A, k) = \langle r_A, e_A, c_A\rangle$, where $r_A = g^{k_A} \bmod p$, $e_A = H(m_A, k, r_A)$, $c_A = k_A + e_A x_A \bmod q$, and H is a hash function. The verifiable signature message is $\sigma_A = \langle m_A, k, s_A\rangle$ that is sent to Bob.

2. Bob verifies σ_A using **SVerify** algorithm. If $e_A = H(m_A, k, g^{c_A} y_A^{e_A} \bmod p)$, Bob signs $s_B = \mathbf{Sign}_{x_B}(m_B, k) = \langle r_B, e_B, c_B\rangle$ and sends $\sigma_B = \langle m_B, k, s_B\rangle$ to Alice, otherwise does nothing.

3. Alice also verifies σ_B. If $e_B = H(m_B, k, g^{c_B} y_B^{e_B} \bmod p)$, Alice releases *keystone*, otherwise aborts.

4. Each participant can prove σ_A (or σ_B) valid by revealing $k = G(ID_{AB})$, $\mathbf{SVerify}_{y_A}(\sigma_A) = \text{True}$ (or $\mathbf{SVerify}_{y_B}(\sigma_B) = \text{True}$).

From the above example, we conclude that FESS is a general idea and has various implementations on most of the existing signature schemes, thus it may also be considered as an interesting extension of concurrent signature [22] presented in EUROCRYPT 2004 that can be constructed from ring signatures. Because it can be implemented from more simple and efficient signature schemes, FESS has higher efficiency than concurrent signature does. We will show executive efficiency comparison between FESS and concurrent signature in next section.

5 Security and Efficiency Analysis of FESS

5.1 Security

In this section, we will discuss the security of FESS in the random oracle model [26].

LEMMA 5.1. *(**Correctness**) All parties' signatures can ensure the right parties send or receive the right messages.*

proof: If $s = \mathbf{Sign}_x(m,k) = \langle r, e, c\rangle$, $r = g^{k'} \bmod p$, $e = H(m,k,r)$ and $c = k' + ex \bmod q$ then $e = H(m,k,g^c y^e \bmod p) \Leftrightarrow \mathbf{SVerify}_y(m,k,s) =$ True. Moreover, if $\mathbf{SVerify}_y(m,k,s) =$ True and $k = G(keystone)$ then $\mathbf{KVerify}_y(m,k,s,keystone) =$ True. □

To prove the *Unforgeability* of FESS, we introduce an important conclusion – *Forking Lemma* [28]. It gives a reductionist security proof for triplet ElGamal-family signature schemes which produce a signature $(Gen, Sign, Verify)$ on a input message m.

LEMMA 5.2. ***(Forking Lemma)*** *Let $\mathcal{A}$ be a probabilistic polynomial time Turing machine whose input only consists of public data. We denote respectively by Q and R the number of queries that $\mathcal{A}$ can ask to the random oracle and the number of queries that $\mathcal{A}$ can ask to the signer. Assume that, within time bound K, $\mathcal{A}$ produces, with probability $\varepsilon \geq 10(R+1)(R+Q)/2^k$ (where k is a security parameter), a valid signature $(m, \sigma_1, h, \sigma_2)$. If the triples (σ_1, h, σ_2) can be simulated without knowing the secret key, with an indistinguishable distribution probability, then there is another machine which has control over the machine obtained from $\mathcal{A}$ replacing interaction with the signer by simulation and produces two valid signatures $(m, \sigma_1, h, \sigma_2)$ and $(m, \sigma_1, h', \sigma_2')$ such that $h \neq h'$ in expected time $T' \leq 120686QT/\varepsilon$.*

LEMMA 5.3. ***(Unforgeability)*** *The FESS is unforgeable under a chosen message attack in the random oracle model.*

proof: The proof is referred to the proof of unforgeability of the signature scheme by Pointcheval and Stern [27], and makes use of the forking lemma [27, 28]. We suppose that G and H are random oracles, and there exists a probabilistic polynomial time Turing machine $\mathcal{A}$ whose input only consists of public data. We assume that $\mathcal{A}$ can make Q_G queries to the random oracle G, Q_H queries to the random oracle H and R queries to the signing oracle $Sign$. Within time bound T, $\mathcal{A}$ produces, with probability $\varepsilon \geq 10(R+1)(R+Q_H)/2^q$ (where q is a security parameter), a valid signature $\langle m, k, \langle r, e, c\rangle\rangle$.

simulation: $\mathcal{S}$ gives the parameters $\langle g, p, q\rangle$ and $y = g^x \bmod p$ to $\mathcal{A}$. $\mathcal{S}$ tries to simulate the challenger by simulating all the oracles to gain the secret key x. $\mathcal{A}$ can query as follows:

G-Queries: $\mathcal{A}$ can query the random oracle G at any time. $\mathcal{S}$ simulates the random oracle by keeping list of tuple $\langle m_i, k_i\rangle$ which is called the G-List. When the oracle is queried with an input $m \in \{0,1\}^*$, $\mathcal{S}$ responds as follows:

1. If the query m is already on the G-List in the tuple $\langle m, k_i\rangle$, then $\mathcal{S}$ outputs k_i.

2. Otherwise $\mathcal{S}$ selects a random $k \in K$, outputs k and adds $\langle m, k\rangle$ to the G-List.

H-Queries: $\mathcal{A}$ can query the random oracle H at any time. $\mathcal{S}$ simulates the random oracle by keeping list of tuple $\langle \sum_i, e_i\rangle$ which is called the H-List, where $\sum_i$ is a triple of $\langle m_i, k_i, r_i\rangle$. When the oracle is queried with an input $\sum$, $\mathcal{S}$ responds as follows:

1. If the query $\sum$ is already on the H-List in the tuple $\langle \sum, e_i\rangle$, then $\mathcal{S}$ outputs e_i.

2. Otherwise $\mathcal{S}$ selects a random $e \in \mathbb{Z}_q$, outputs e and adds $\langle \sum, e\rangle$ to the H-List.

KGen-Queries: $\mathcal{S}$ maintain a K-List of tuples $\langle keystone, k\rangle$. $\mathcal{A}$ can request that $\mathcal{S}$ selects a $keystone \in K_{ks}$ and returns fix $k = G(keystone)$. $\mathcal{S}$ chooses a random $keystone \in K_{ks}$ and computes $k = G(keystone)$. $\mathcal{S}$ outputs k and adds $\langle keystone, k\rangle$ to the K-List. In fact, K-List is a sublist of G-List, but is required to answer $KReveal$ queries.

KReveal-Queries: $\mathcal{A}$ can request keystone of any keystone fix $k \in K$ produced by a previous $KGen$ query. If there exists a tuple $\langle keystone, k\rangle$ on the K-List, then $\mathcal{S}$ returns $keystone$, otherwise it outputs invalid.

Sign-Queries: $\mathcal{S}$ simulates the signature oracle by accepting signature queries of the form $\langle m, k\rangle$ where $m \in M$ is the message to be signed and $k \in K$ is a keystone fix. $\mathcal{S}$ answers the query as follows:

1. $\mathcal{S}$ picks a random c `and` $e \in \mathbb{Z}_q$ which e isn't equal to some previous output for the H oracle.
2. $\mathcal{S}$ computes $r = g^c y^e \bmod p$. If $\sum = \langle m, k, r\rangle$ is some previous input for the H oracle, then return to step 1.
3. $\mathcal{S}$ adds a tuple $\langle \sum, e\rangle$ to H-List.
4. $\mathcal{S}$ outputs $s = \langle r, e, c\rangle$ as the signature for message m.

NOTE: Here we must check whether the distributions of real signature δ and forged signature δ' are same.

$$\begin{cases} \delta = \{(r, e, c) | k \in \mathbb{Z}_q, k \neq 0, e \in \mathbb{Z}_q, r = g^k \bmod p, c = k + xe \bmod q\} \\ \delta' = \{(r, e, c) | e \in \mathbb{Z}_q, c \in \mathbb{Z}_q, r = g^c y^e \neq 1 \bmod p\} \end{cases}$$

First we compute the probability of a real signature signed using secret key,

$$\Pr_{\delta}[(r, e, c) = (\varepsilon, \beta, \gamma)] = \Pr_{k \neq 0, e}[r = g^k = \varepsilon, e = \beta, c = k + xe = \gamma] = \frac{1}{q(q-1)}.$$

The probability of a forged signature is

$$\Pr_{\delta'}[(r, e, c) = (\varepsilon, \beta, \gamma)] = \Pr_{e,c}[e = \beta, c = \gamma, r = g^c y^e = \varepsilon \neq 1 \bmod p] = \frac{1}{q(q-1)}.$$

So the triple $\langle r, e, c\rangle$ can be simulated without knowing the secret key, with an indistinguishable distribution probability. Thus, the signing oracle simulated by

$\mathcal{S}$ is high quality, and thereby $\mathcal{A}$ is very satisfied with the Sign-Queries' answer. He can fully exert his forgery ability.

Output: Finally, with non-negligible probability, $\mathcal{A}$ output a signature $s = \langle r, e, c\rangle$ with a message $m \in M$ and $k \in K$, where $\mathbf{SVerify}_y(m, k, s) = True$, in the case that $\mathcal{A}$ produces $k = G(keystone)$ through $KGen$ queries and $KReveal$ query with input k and no Sign query with input $\langle m, k\rangle$ were made by $\mathcal{A}$.

Now $\mathcal{S}$ can play the simulation twice so that $\mathcal{A}$ should produce two valid signature $s = \langle r, e, c\rangle$ and $s' = \langle r, e', c'\rangle$ with $e \neq e'$. Then we have the following equations.

$$r = g^c y^e = g^{c-xe} = g^{c'-xe'} = g^{c'} y^{e'} \bmod p \tag{4}$$

From above equations $\mathcal{S}$ can solve the hard discrete logarithm:

$$\log_g y = -x = \frac{c - c'}{e' - e} \bmod q \tag{5}$$

within expected time less than $\frac{120686 \times 2^q \times Q_H T}{10 \times (R+1) \times (R+Q_H)}$. This contradicts the hardness of the discrete logarithm problem. □

In [22], to provide two participants' signatures are concurrent, Chen et al make full use of the ambiguous property of ring signatures. Everyone except initial signer cannot confirm who is the signer within two participants until initial signer releases the keystone. But in FESS, to ensure simultaneity, we introduce the property of *Dormancy*. Two signatures that have been exchanged would not go to valid until the secret information *keystone* is released to arouse them.

LEMMA 5.4. ***(Dormancy)*** *The FESS is dormant before the secret information keystone is released in the random oracle model.*

proof: The random oracle assumption is same as before. We suppose there exists a probabilistic polynomial time Turing machine $\mathcal{A}$ whose input only consists of public data. We assume that $\mathcal{A}$ can make Q_G queries to the random oracle G, Q_K queries to the random oracle $KGen$ and R queries to the signing oracle $Sign$.

simulation: $\mathcal{S}$ gives the parameters $\langle g, p, q\rangle$ and $y = g^x \bmod p$ to $\mathcal{A}$. $\mathcal{S}$ tries to simulate the challenger by simulating all the oracles to reveal a *keystone* with $k = G(keystone)$. $\mathcal{A}$ can query as in LEMMA 5.3.

Output: Finally, with non-negligible probability, $\mathcal{A}$ outputs a *keystone* and a signature $s = \langle r, e, c\rangle$ with a message $m \in M$ and $k \in K$, where $\mathbf{KVerify}_y(m, k, s, keystone) = True$, in the case that $\mathcal{A}$ produces $k = G(keystone)$ through $KGen$ queries and no $KReveal$ query with input k was made by $\mathcal{A}$.

In this model, it is an easy job for $\mathcal{A}$ to obtain a valid signature $s = \langle r, e, c\rangle$ with $\mathbf{SVerify}_y(m, k, s) = True$ through **Sign-Queries**. But $\mathcal{A}$ cannot make query to **KReveal-Queries**, so he has a probability $\frac{Q_G Q_K}{q}$, which is a negligible probability, to reveal *keystone*. This contradicts the assumption that, with non-negligible probability, $\mathcal{A}$ outputs a *keystone* and a signature $s = \langle r, e, c\rangle$ with a message $m \in M$ and $k \in K$, where $\mathbf{KVerify}_y(m, k, s, keystone) = True$. □

LEMMA 5.5. ***(Fairness)*** *The FESS is fair in the random oracle model.*

proof: The random oracle assumption is same as in LEMMA 5.3. We suppose there exists a probabilistic polynomial time Turing machine $\mathcal{A}$ whose input only consists of public data. We assume that $\mathcal{A}$ can make Q_G queries to the random oracle G, Q_H queries to the random oracle H, Q_K queries to the random oracle $KGen$ and R queries to the signing oracle $Sign$.

simulation: $\mathcal{S}$ gives the parameters $\langle g, p, q\rangle$ and $y = g^x \bmod p$ to $\mathcal{A}$. $\mathcal{S}$ tries to simulate the challenger by simulating all the oracles to gain the secret key x or reveal a $keystone$ with $k = G(keystone)$. $\mathcal{A}$ can query as before.

Output: Finally, with non-negligible probability, $\mathcal{A}$ output a $keystone$ and a signature $s = \langle r, e, c\rangle$ with a message $m \in M$ and $k \in K$, where $\mathbf{KVerify}_y$ $(m, k, s, keystone) = True$, one of the following two cases holds:

1. $\mathcal{A}$ produces $k = G(keystone)$ through $KGen$ queries and $KReveal$ query with input k and no Sign query with input $\langle m, k\rangle$ were made by $\mathcal{A}$.
2. $\mathcal{A}$ produces $k = G(keystone)$ through $KGen$ queries and no $KReveal$ query with input k was made by $\mathcal{A}$.

In the case 1, it is easy to educe a contradiction from LEMMA 5.3. In the case 2, the output conditions can occur only with a negligible probability. This follows LEMMA 5.4. □

THEOREM 5.6. *The FESS are secure in the random oracle model, assuming the hardness of the discrete logarithm problem.*

proof: The proof follows directly from correctness, unforgeability, dormancy and fairness. □

5.2 Efficiency

Because FESS can be implemented from more simple and efficient signature schemes, it has higher efficiency than concurrent signature does. Executive efficiency comparison between FESS and concurrent signature is given in Table 1. In the table 1, " E " denotes the number of exponentiation in $\mathbb{Z}_p$, " M_p " denotes the number of multiplication in $\mathbb{Z}_p$, " M_q " denotes the number of multiplication in $\mathbb{Z}_q$, " A " denotes the number of addition in $\mathbb{Z}_q$, " H " denotes the number of hash operation.

Table 1. Efficiency Comparison

Algorithm	FESS	Concurrent Signature
Initial Sign	$1E + 1M_q + 1A + 2H$	$2E + 1M_q + 1M_p + 2A + 2H$
Respond Sign	$1E + 1M_q + 1A + 1H$	$2E + 1M_q + 1M_p + 2A + 1H$
SVerify	$2E + 1M_p + 1H$	$3E + 2M_p + 1A + 1H$
KVerify	$2E + 1M_p + 2H$	$3E + 2M_p + 1A + 2H$

6 Conclusions

In this paper we propose a secure and efficient signature scheme — *FESS* that allows two players to exchange digital signatures in a fair way. It is a general idea and can be implemented from most of the existing signatures. In this scheme, each unwakened signature can be verified easily by the other player, but it doesn't go into effect until an extra piece of information *keystone* is released by one of the players. Once the *keystone* released, two signatures are both aroused and become effective. A key feature of the proposed scheme is that two players can exchange digital signatures simultaneously through a secret commitment without involvement of any Trusted Third Party. Although FESS does not overcome the weakness of concurrent signature, which is the initial party controls the keystone, but, from the comparison, we can point out that the executive efficiency of FESS is higher than that of concurrent signatures. For having variety implementations from most of existing signature schemes, FESS can be applied to different environment. As a useful cryptographic tool, FESS provides a primitive to build efficient fair exchange and contract signing protocols.

Our scheme can also be extended to the multi-party case easily. In this case, the security assumption could also be proved in the same way. We have taken the directions for future research to reduce the initial signer's advantage of revelation of *keystone*.

References

1. E. F. Brickell, D. Chaum, I. B. Damgard and J. van de Graaf, *Gradual and verifiable release of a secret*, Advances in Cryptology: Proceedings of Crypto'87, LNCS vol. 293, Santa Barbara, California, August, 1987, pp. 156-166.
2. M. Ben-Or, O. Goldreich, S. Micali and R. Rivest, *A fair protocol for signing contracts*, IEEE Transactions on Information Theory, IT-36(1), January 1990, pp.40-46.
3. D. Boneh, and M. Naor, *Timed commitments (extended abstract)*, In Advances in Cryptology - CRYPTO 2000, LNCS vol. 1880, Springer-Verlag, 2000, pp. 236-254.
4. R. Cleve, *Controlled gradual disclosure schemes for random bits and their applications*, Advances in Cryptology: Proceedings of Crypto'89, LNCS vol. 435, Santa Barbara, California, August 1989, pp. 573-588.
5. I. B. Damgard, *Practical and provably secure release of a secret and exchange of signatures*, Advances in Cryptology: Proceedings of Eurocrypt'93, LNCS vol. 765, Lofthus, Norway, May 1993, pp. 200-217.
6. O. Goldreich, *A simple protocol for signing contracts*, In Advances in Cryptology - CRYPTO 1983, Plenum Press, New York, 1984, pp. 133-136.
7. J. Garay, and C. Pomerance, *Timed fair exchange of standard signatures*, In Proc. Financial Cryptography 2003, LNCS vol. 2742, Springer-Verlag, 2003, pp. 190-207.
8. N. Asokan, V. Shoup, and M. Waidner, *Optimistic fair exchange of digital signatures*, Advances in Cryptology - EUROCRYPT'98. LNCS, Vol. 1403, Springer-Verlag, 1998, pp. 591-606.

9. N. Asokan, V. Shoup, and M. Waidner, *Optimistic fair exchange of signatures*, In IEEE Journal on Selected Areas in Communication vol. 18(4), 2000, pp. 593-610.
10. C. Boyd and E. Foo, *Off-line fair payment protocols using convertible signature*, Proceedings of Asiacrypt'98, LNCS vol. 1514, Springer-Verlag, 1998, pp. 271-285.
11. F. Bao, *Colluding attacks to a payment protocol and two signature exchange schemes*, Proceedings of Asiacrypt 2004, LNCS vol. 3329, Springer-Verlag, 2004, pp. 417-429.
12. F. Bao, R. H. Deng and W. Mao, *Efficient and practical fair exchange protocols with off-line TTP*, Proceedings of 1998 IEEE Symposium on Security and Privacy, Oakland, California, May 1998, pp. 77-85.
13. D. Boneh, C. Gentry, B. Lynn and H. Shacham, *Aggregrate and verifiably encrypted signatures from bilinear maps*, In Advances in Cryptology -EUROCRYPT 2003, LNCS vol. 2656, Springer-Verlag, 2003, pp. 416-432.
14. R. H. Deng, L. Gong, A. A. Lazar and W. Wang, *Practical protocols for certified electronic mail*, Journal of Network and Systems Management, 4(3), 1996, pp. 279-297.
15. Y. Dodis, and L. Reyzin, *Breaking and repairing optimistic fair exchange from PODC 2003*, In ACM Workshop on Digital Rights Management (DRM), October 2003, pp. 47-54.
16. M. Franklin and M. Reiter, *Fair exchange with a semi-trusted third party*, Proceedings of 4th ACM Conference on Computer and Communications Security, Zurich, Switzerland, April 1997, pp. 1-6.
17. J. Garay, M. Jakobsson and P. MacKenzie, *Abuse-free optimistic contract signing*, In Advances in Cryptology - CRYPTO 1999, LNCS vol. 1666, Springer-Verlag, 1999, pp. 449-466.
18. J. M. Park, E. Chong, H. Siegel, I. Ray, *Constructing Fair-Exchange Protocols for E-Commerce Via Distributed Computation of RSA Signatures*, Proceedings of the Twenty-Second ACM Symposium on Principles of Distributed Computing (PODC 2003), Boston, Massachusetts, USA, July 13-16, 2003, pp. 172-181.
19. J. Zhou and D. Gollmann, *A fair non-repudiation protocol*, Proceedings of 1996 IEEE Symposium on Security and Privacy, Oakland, California, May 1996, pp. 55-61.
20. J. Zhou and D. Gollmann, *An efficient non-repudiation protocol*, Proceedings of 10th IEEE Computer Security Foundations Workshop, Rockport, Massachusetts, June 1997, pp. 126-132.
21. Jianying Zhou, Robert Deng, and F. Bao, *Some remarks on a fair exchange protocol*, Third International Workshop on Practice and Theory in Public Key Cryptosystems, PKC 2000. LNCS, Vol. 1751, Springer-Verlag, Australia, 2000, pp. 46-57.
22. Liqun Chen, Caroline Kudla and Kenneth G.Paterson, *Concurrent Signature*, Advances in Cryptology - EUROCRYPT 2004. LNCS, Vol. 3027, Springer-Verlag , 2004, pp. 287-305.
23. R. Rivest, A. Shamir and Y. Tauman, *How to leak a secret*, In Advances in Cryptology - ASIACRYPT 2001, LNCS vol. 2248, Springer-Verlag, 2001, pp. 552-565.
24. M. Abe, M. Ohkubo, and K. Suzuki, *1-out-of-n signatures from a variety of keys*, In Advances in Cryptology - ASIACRYPT 2002, LNCS vol. 2501, Springer-Verlag, 2002, pp. 415-432.
25. G. Ateniese, *Efficient verifiable encryption and fair exchange of digital signatures*, Proceedings of the 6th ACM Conference on Computer and Communications Security (CCS), 1999, pp. 138-146.

26. M. Bellare, and P. Rogaway, *Random oracles are practical: a paradigm for designing efficient protocols*, In Proc. of the 1st CCCS, ACM press, 1993, pp. 62-73.
27. D. Pointcheval and J. Stern, *Security proofs for signature schemes*, In Advances in Cryptology - EUROCRYPT 1996, LNCS vol. 1070, Springer-Verlag, 1996, pp. 387-398.
28. D. Pointcheval and J. Stern, *Security arguments for digital signatures and blind signatures*, In Journal of Cryptology, vol. 13(2000), pp. 361-396.
29. W. Susilo, Y. Mu, and F. Zhang, *Perfect concurrent signature schemes*, In: Information and Communications Security (ICICS 04), LNCS 3269, Spriger- Verlag, 2004, pp. 14-26.
30. Guilin Wang, Feng Bao, and Jianying Zhou, *The Fairness of Perfect Concurrent Signatures*, In: Information and Communications Security (ICICS 06), LNCS 4307, Spriger- Verlag, 2006, pp. 435-451.