Asymmetric Concurrent Signatures

Khanh Nguyen

Gemplus R&D,
12 Ayer Rajah Crescent,
Singapore 139941
kenny.nguyen-qk@gemplus.com

Abstract. The concept of concurrent signatures allows two entities to produce two signatures in such a way that, the signer of each signature is ambiguous from a third party's point of view until the release of a secret, known as the *keystone*. Once the keystone is released, both signatures become binding to their respective signers concurrently. Previous concurrent signature schemes use the concept of ring signatures in their construction. Ring signatures identify the ring and thus concurrent signatures constructed from ring signature are related and linkable. We propose a new concurrent signature scheme which is independent of the ring signature concept. Our concurrent signatures are anonymous. The ordinary signatures obtained from our concurrent signature protocol are unlinkable and do not reveal which concurrent signature transaction has occurred. The price we pay is our concurrent signatures are *asymmetric* in the sense that the initial signature and subsequent signatures are not of the same construction.

1 Introduction

1.1 Concurrent Signatures

The concept of concurrent signatures was introduced in [3]. In a concurrent signature scheme, two parties A and B interact without the help of any third party to sign messages M_A and M_B in such a way that both signatures are ambiguous without an extra piece of information, known as the *keystone*. Without the keystone, from a third party's point of view, both signatures are ambiguous as they could have been generated by either of the parties and thus do not represent any entity's commitments to the messages. With the keystone, the signer for each signature is identified and both signatures become instantly binding to their respective signers. In the original proposal [3], the keystone is a randomly chosen piece of information and possessed by the protocol's initiator. During the signature generation phase, the keystone is not known to other parties. When the validation of concurrent signatures is required, the initiator reveals the keystone which validates all the signatures *concurrently*.

Concurrent signatures find applications in fair exchange of digital signatures and fair tendering of contracts. To exchange digital signatures, two parties A and B engage in a concurrent signature protocol by which each party receives the other's concurrent signature on the desired message. The exchange is fair in the sense that without the knowledge of the keystone, both of the concurrent signatures are ambiguous and non-binding to the signers. To prove the validity of the concurrent signature generated by party B, party A would need to reveal the keystone. The revelation of this knowledge in turn would bind the concurrent signature generated by party A to its signer, i.e., to party A, hence the fairness is achieved. This notion of fairness is weak as it allows party A to reveal the keystone in private to another party. In this scenario, the party B who is denied of the knowledge of the keystone, can not prove the validity of the concurrent signature generated by party A. However, this weak notion of fairness is acceptable in many instances, such as contracts for party A to buy physical goods from party B. The advantage of fair exchange using concurrent signatures is that it eliminates the requirement of a trusted third party from the protocol and it is not overly interactive. Previous solutions to the problem of fair exchange of digital signatures (see [1,2] and references therein for a detailed survey) are either highly interactive with multiple rounds of exchange or require the existence of a third party trusted by both parties A and B. Multiple rounds of exchange are inefficient while the existence of a trusted third party is not always warranted.

Previous concurrent signature proposals [3,10] are based on the concept of ring signatures. In those schemes, each concurrent signature is a ring signature [8] generated from the ring consisting of all involved parties. Each ring signature while identifies that the signer is a member of the ring, does not reveal the actual signer. The ambiguity of each concurrent signature comes from this anonymous attribute of the underlying ring signature scheme. Ring signatures while hide the actual signer identity in the ring, do identify the ring. Hence it does leak information about the transaction.

Furthermore, in those proposals [3,10], once the keystone is revealed, its value must be included in each signature in order to bind the signature to its actual signer. This means all the signatures obtained from a concurrent signature protocol are linked together. This is not desirable when the anonymity of the transaction is required.

1.2 Our Contributions

In this paper, we propose a concurrent signature scheme which offers a stronger notion of anonymity than previous schemes. Under the Decisional Diffie-Hellman assumption, each of our concurrent signatures could be created by anyone (not just by a member of the ring as done previously) without the knowledge of the keystone. Due to this, a set of k concurrent signatures in our scheme is *anonymous* and releases no information about whether the transaction is real or faked. Once the keystone is revealed, our concurrent signatures could be converted to ordinary Schnorr-like signatures. Except for one signature issued by the protocol initiator, our converted signatures do not contain the keystone and thus remain *unlinkable*. This is in total contrast to all previous proposals [3,10] in which once the keystone is revealed, all the signatures obtained from a concurrent signature transaction are linked by the formation of the ring.

The cost we pay is that our concurrent signatures are *asymmetric*. The signature signed by the party who processes the keystone is different from the other signatures. However, this is not a real issue in most applications where concurrent signatures find their uses.

1.3 Technical Approach

Each concurrent signature could be viewed as a *promise of signature* which could be converted into an ordinary signature by revealing a secret information – the *keystone*. In CKP concurrent signatures, a promise of signature is a ring signature issued by a member of the ring. Upon the release of the keystone information, it converts the ring signature to a signature that could only be issued by the identified member. While ring signatures could be used to construct promises of signatures, it is not a necessary condition.

In [1], Asokan, Shoup and Waidner (ASW) proposed a technique which reduces a promise of a signature to a promise of a homomorphic pre-image. Without the homomorphic pre-image, the promise of signature looks indistinguishable from random elements of the signature space as it could have been created by anyone using solely public information. With the homomorphic pre-image, the promise of signature could be converted into an ordinary signature. This technique is applicable to most well-known signature schemes (e.g., Schnorr [9], RSA [7], DSS [5] and GQ [4] signature schemes). If we view the homomorphic pre-image as the keystone, then an ASW promise of signature is indeed a concurrent signature. The advantage of using the ASW construction is that a promise of signature while contains public information about a single entity, could be created by anyone; hence without the keystone, it leaks no information about the signature. This is a genuine advantage from ring signatures where signatures are linked together by the nature of the ring.

An ASW promise of a signature could only be constructed with the knowledge of homomorphic pre-image, i.e., the keystone. In the standard model of the concurrent signatures, the keystone should only be known to one party and not to be shared with other parties. Thus except for one party in the protocol, the ASW promise of signature construction could not be used for others. In those cases, we need a promise of signature construction that does not require the knowledge of the keystone. Fortunately, a such new construction is possible for a variant of the Schnorr signature scheme. This construction is described in section 2.

1.4 Organization

The remaining of the paper is organized as follows. Section 2 describes the concept of promises of signatures. This concept forms the main basic building block for our concurrent signature scheme. Section 3 discusses our formal definitions which include both the definitions for concurrent signature scheme and protocol. Section 4 gives the security model for our concurrent signature scheme. Here, we modify the security model from [3] in order to capture the additional security properties of *anonymity* and *unlinkability*. Section 5 proposes a concrete concurrent signature scheme. It also discusses the security of the construction and extensions of this proposal to multi-party case.

2 Promises of Signatures

Definition 1. *Let f be some cryptographic function. The value $\sigma = \langle s, u \rangle$ is said to be a valid promise of signature $\rho = \langle k, u \rangle$ on some message m if the following conditions hold:*

- **Publicly Verifiable:** *given σ, everyone is convinced that if there exists $k = f^{-1}(s)$ then $\rho = \langle k, u \rangle$ is a valid ordinary signature.*
- **Anonymity:** *without the knowledge of $k = f^{-1}(s)$, σ is indistinguishable from random elements of the signature space.*

To convert a promise of signature to an ordinary signature, one only needs to reveal the value k such that $f(k)$ gives s. The verification $s = f(k)$ is adequate to verify the signature $\rho = \langle k, u \rangle$ provided that $\sigma = \langle s, u \rangle$ is a valid promise of signature.

2.1 Promises of Schnorr Signatures

The Schnorr signature scheme [9] is constructed as follows:

- **Setup:** the public parameters are the primes p and q of appropriate size such that $q|(p-1)$, and a generator g for the subgroup in $\mathbb{Z}_p^*$ of order q.
- **Key Generation:** select a random $x \in \mathbb{Z}_q$ and compute $h = g^x \bmod p$. The public key is $\{p, q, g, h\}$ and its corresponding private key is x.
- **Sign:** the signer chooses a random $r \in Z_q$ and computes $k = cx + r \bmod q$ where $c = H(g^r \bmod p, m)$ and H is a hash function. The signature is $\rho = (k, c)$.
- **Verify:** to verify the signature $\rho = (k, c)$, one checks $c = H(g^k h^{-c} \bmod p, m)$.

For the remaining of this paper, we omit the modular reduction $\bmod p$ from the modular exponentiation $a^r \bmod p$ for any given $a \in \mathbb{Z}_p^*$ and any arbitrary r. Instead, we use a^r to denote $a^r \bmod p$ whenever the context is clear.

The promise of the Schnorr signature $\rho = \langle k, c \rangle$ is $\sigma = \langle s, c \rangle$, where $s = f(k)$ for $f(x) = g^x$. To verify the promise of signature, one verifies $c = H(sh^{-c}, m)$. To convert the promise of signature to an ordinary signature, the signer reveals k. The verification is $g^k = s$ which implies $c = H(g^k h^{-c}, m)$, and thus $\rho = \langle k, c \rangle$ is a valid Schnorr signature.

Lemma 1. *The value $\sigma = \langle s, c \rangle$ where $c = H(sh^{-c}, m)$ is a promise of signature $\rho = \langle k, c \rangle$, where $s = f(k)$ for $f(x) = g^x$.*

2.2 Promises of Schnorr-Like Signatures

Apart from the above promise of signature. We also need a promise of signature construction in which the promise of signature could be constructed without the knowledge of the promise, i.e., without the knowledge of the homomorphic inverse. We design such a promise of signature construction for a variant of Schnorr signature scheme. This variant of Schnorr signature scheme is constructed as follows:

- **Setup:** the public parameters are the primes p and q of appropriate size such that $q|(p-1)$, and a generator g for the subgroup in $\mathbb{Z}_p^*$ of order q.
- **Key Generation:** select a random $x \in \mathbb{Z}_q$ and compute $h = g^x$. The public key is $\{p, q, g, h\}$ and the corresponding private key is x.
- **Sign:** the signer chooses two random κ and $r \in Z_q$ and computes $\kappa = (r-c)/x \bmod q$ where $c = H(g^r, m)$ and H is a hash function. The signature is $\rho = (\kappa, c)$.
- **Verify:** to verify the signature $\rho = (\kappa, c)$, one checks $c = H(g^c h^\kappa, m)$.

This is a straightforward variant of the Schnorr signature scheme. The difference here is that the equation $r = \kappa x + c \bmod q$ is used instead of the standard equation $r = k - cx \bmod q$. Using essentially the same security proof for the original Schnorr signature scheme [6], we have the following security result:

Lemma 2. *In the random oracle model, this Schnorr-like signature scheme is existentially unforgeable against adaptive chosen message attacks, assuming the hardness of the discrete logarithm problem.*

The promise of the Schnorr-like signature $\rho = \langle \kappa, c \rangle$ is $\omega = \langle s, \kappa_1, c \rangle$ where $s = f(\kappa - \kappa_1)$ for $f(x) = h^x$. To verify this promise of signature, one verifies $c = H(g^c h^{\kappa_1} s, m)$. To convert the promise of signature to an ordinary signature, the signer reveals $\kappa_2 = \kappa - \kappa_1 \bmod q$. The verification is $h^{\kappa_2} = s$ which implies $c = H(g^c h^{\kappa_1 + \kappa_2}, m)$; and hence $\omega = \langle \kappa, c \rangle = \langle \kappa_1 + \kappa_2 \bmod q, c \rangle$ is a valid signature. Note that the knowledge of κ_2 is not *required* in order to create the promise of signature ω.

Lemma 3. *The value $\omega = \langle s, \kappa_1, c \rangle$ where $c = H(g^c h^{\kappa_1} s, m)$ is a valid promise of signature $\rho = \langle \kappa, c \rangle$, where $\kappa = \kappa_1 + \kappa_2 \bmod q$ and $s = f(\kappa_2)$ for $f(x) = h^x$.*

Proof. The verification $c = H(g^c h^{\kappa_1} s, m)$ requires only public information. If there exists κ_2 such that $h^{\kappa_2} = s$, we have $c = H(g^c h^{\kappa_1 + \kappa_2}, m)$ and thus $\rho = \langle \kappa, c \rangle = \langle \kappa_1 + \kappa_2, c \rangle$ is a valid ordinary signature. Hence, the publicly verifiable condition is satisfied.

The anonymity property of this promise of signature follows from the fact that one could generate a valid promise of signature from public information: the simulator chooses two random r and $\kappa_1 \in \mathbb{Z}_q$, computes $c = H(g^r h^{\kappa_1}, m)$ and sets $s = g^{r-c}$. We have $\omega = \langle s, \kappa_1, c \rangle$ satisfies $c = H(g^c h^{\kappa_1} s, m)$ and thus is a valid promise of signature constructed solely from public information.

3 Formal Definitions

3.1 Asymmetric Concurrent Signatures

In this section, we give a formal definition for concurrent signature schemes. Our definition is an adaption of the CKP definition with some modification accommodating the asymmetric property of signatures.

Definition 2. *An asymmetric concurrent signature scheme is a digital signature scheme consisted of the following algorithms:*

SETUP: A probabilistic algorithm that on input of a security parameter l, outputs: the set of participants $\mathcal{U}$ each equipped with a private key x_i and the corresponding public key X_i. The algorithm also outputs the message space $\mathcal{M}$, the keystone space $\mathcal{K}$, the keystone fix space $\mathcal{F}$ and some additional system parameters π. The algorithm also defines a function $KGEN : \mathcal{K} \rightarrow \mathcal{F}$, a set of functions $\{KGEN_j : \mathcal{K} \rightarrow F\}$ and a keystone transformation function $KTRAN : \mathcal{F} \times \{x_i\} \rightarrow \mathcal{F}$.

ISIGN: A probabilistic algorithm that on inputs $\langle X_i, x_i, M_i \rangle$, where X_i is a public key, x_i is the corresponding private key and $M_i \in \mathcal{M}$ is the message, outputs a promise of signature $\sigma_i = \langle s, u_i \rangle$ and the relevant keystone $k \in \mathcal{K}$, where $s = KGEN(k)$.

SSIGN: A probabilistic algorithm that on inputs $\langle X_j, x_j, M_j \rangle$ and s, where X_j is a public key, x_j is the corresponding private key , $M_j \in \mathcal{M}$ is the message and s a keystone fix, outputs a promise of signature $\omega_j = \langle s', v_j \rangle$ where $s' = KTRAN(s, x_j)$.

IVERIFY: An algorithm which takes as input a promise of signature $\sigma_i = \langle s, u_i \rangle$ and a message M_i, and outputs accept or reject. The algorithm outputs accept if and only if σ_i is a valid promise of the signature $\langle k, \sigma_i \rangle$ on message M_i with $f(x) = KGEN(x)$.

SVERIFY: An algorithm which takes as input a promise of signature $\omega_j = \langle s', v_j \rangle$, a message M_j, and outputs accept or reject. The algorithm outputs accept if and only if $omega_j$ is a valid promise of the signature $\langle k, \omega_j \rangle$ on message M_j with $f(x) = KGEN_j(x)$.

VERIFY: An algorithm which takes as input, a promise of signature $\sigma_i = \langle s, u_i \rangle$ (or $\omega_j = \langle s', v_j \rangle$), a message M and a keystone $k \in K$, and outputs accept or reject. The algorithm will output accept if and only if $s = KGEN(k)$ and $\langle k, \sigma_i \rangle$ (or $s' = KGEN_j(k)$ and $\langle k, \omega_j \rangle$) forms a valid signature on message M.

We note that our concurrent signature protocol differs from the protocol of [3] in that the keystone fixes used for A and B are different. This crucial property allows us to unlink signatures obtained from the same signature protocol. For this to work, we require the keystone fix transform function $KTRAN$ to be isomorphic and that s' leaks no more information about k rather than s. This security notion is discussed in the following section.

3.2 Concurrent Signature Protocol

Using the above definition, our two-party concurrent signature protocol is as follows. Let A be the initial signer who initiates the protocol and B be the matching signer who responds to the initial signer. A and B first run SETUP algorithm to generate the public key parameters for the system. We assume A's public and private keys are X_A and x_A respectively. Likewise, B's public and private keys are X_B and x_B respectively. Then A and B engage in the following protocol:

1: To start of the protocol, A runs ISIGN algorithm to generate a promise of signature $\sigma_A = \langle s, u_A \rangle$ and the relevant keystone k for the message M_A. The values of σ_A and M_A are sent to B.
2: Upon receiving σ_A and M_A, B verifies the validity of σ_A using IVERIFY. If σ_A is a valid promise of signature on message M_A, B uses the keystone fix s to run SSIGN algorithm to generate a promise of signature $\omega_B = \langle s', v_B \rangle$ for the message M_B. The values of ω_B and M_B are sent to A.
3: Upon receiving ω_B and M_B, A runs SVERIFY to verify the validity of ω_B. If so, A uses the keystone k to verify the keystone fix s'. If this keystone fix is valid, A forwards the keystone k to B.

4 Security Model

The original security model of [3] addresses four basic security properties, namely completeness, fairness, ambiguity and unforgeability. The unforgeability property is somewhat redundant as it has already been captured with the fairness property. If concurrent signatures are forgeable, fairness would not be achieved.

We require our concurrent signature protocol to achieve four security requirements, namely completeness, fairness, anonymity and unlinkability. The completeness and fairness properties are the standard security requirements in the original model of [3] while the anonymity and unlinkability properties are new. Our anonymity property replaces the ambiguity property in the original model. It provides a stronger anonymity attribute for concurrent signatures than what was originally allowed with the ambiguity property. The unlinkability property addresses the anonymity of ordinary signatures obtained from the concurrent signature protocol. This property is new and was not addressed in the original model.

We note that our anonymity and unlinkability properties are not overlapped. The anonymity property deals with the concurrent signatures while the unlinkability property deals with the ordinary signatures obtained from the concurrent signature protocol (upon the revelation of the keystone). It is quite possible that once the keystone is revealed, it would link the anonymous concurrent signatures together and thus the obtained (ordinary) signatures are not unlinkable.

4.1 Completeness

Definition 3. *A concurrent signature protocol is complete if the following conditions hold:*

- *If* $\sigma_i = \langle s, u_i\rangle = ISIGN(X_i, x_i, M_i)$*, then* $IVERIFY(s, u_i) = accept$*. Moreover, if* $KGEN(k) = s$ *for some* $k \in \mathcal{K}$*,* $VERIFY(k, s, u_i) = accept$*.*
- *If* $\omega_j = \langle s', v_j\rangle = SSIGN(X_j, x_j, M_j)$*, then* $SVERIFY(k, s', v_j) = accept$*. Moreover, if* $KGEN_j(k) = s'$ *for some* $k \in \mathcal{K}$*,* $VERIFY(k, s', v_j) = accept$*.*
- *If* $KGEN(k) = s$ *and* $KGEN_j(k') = s'$*, then* $k = k'$*.*

4.2 Fairness

The fairness property is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

GAME 1*:*

Setup: as SETUP in section 3.1. The challenger $\mathcal{C}$ is given the public key X and its corresponding secret key x. The adversary is given the public key X' and its corresponding secret key x'.

Queries: the adversary $\mathcal{A}$ is allowed to make a sequence of the following queries to the challenger $\mathcal{C}$:

- *ISIGN* query, $\mathcal{A}$ will supply the message m_i and $\mathcal{C}$ will output the promise of signature $\sigma_i = \langle s_i, u_i\rangle = ISIGN(X, x, m_i)$.
- *SSIGN* query, $\mathcal{A}$ will supply his promise of signature $\hat{\sigma_j} = \langle \hat{s_j}, \hat{u_j}\rangle$ with the message m_j. If $IVERIFY(\hat{s_j}, \hat{u_j}) = accept$, $\mathcal{A}$ obtains from $\mathcal{C}$: $\omega_j = \langle s_j, v_j\rangle = SSIGN(X, x, m_j, \hat{s_j})$; otherwise $\mathcal{A}$ obtains nothing.
- *KRELEASE* query, $\mathcal{A}$ requests $\mathcal{C}$ to reveal the keystone $k \in \mathcal{K}$ used to produce the keystone fix $s \in \mathcal{F}$ in a previous *ISIGN* query.

Output: Finally $\mathcal{A}$ outputs a string ϕ and $\mathcal{C}$ outputs a string $\bar{\phi}$. The adversary wins if either of the following conditions hold:

- $\phi = (k, \sigma) = (k, s, u)$, $VERIFY(k, s, u) = accept$ and $\mathcal{A}$ has not made *KRELEASE* query for s or,
- $\phi = (k, \omega) = (k, s, v)$, $VERIFY(k, s, v) = accept$, and $s = KTRAN(\hat{s}, x)$; $\bar{\phi} = (k, \sigma) = (k, \hat{s}, \hat{u})$ and $VERIFY(k, \hat{s}, \hat{u}) \neq accept$.

Definition 4. *A concurrent signature protocol is fair if the advantage of an polynomial-bounded adversary in the above game is negligible.*

4.3 Anonymity

Definition 5. *A concurrent signature is anonymous if the following conditions hold:*

- *The value* $\sigma_i = \langle s, u_i\rangle = ISIGN(X_i, x_i, M_i)$ *is indistinguishable from random elements of the signature space.*
- *The tuple* (s, ω_j) *where* s *is the keystone fix input to* $SSIGN$ *and* $\omega_j = \langle s', v_j\rangle = SSIGN(X_j, x_j, M_j, s)$ *is indistinguishable from random elements of the signature space.*

4.4 Unlinkability

The unlinkability property is defined by the following game between an adversary $\mathcal{A}$ and a challenger $\mathcal{C}$.

GAME 2*:*

Setup: as SETUP in section 3.1. The challenger $\mathcal{C}$ is also given two pairs of public and private keys (X, x) and (X', x').

Challenge: The challenger simulates by himself two instances of the concurrent signature protocol (using his two different pairs of public and secret keys (X, x) and (X', x') to obtain two different pairs of converted signatures (ρ_0, ρ_0') and (ρ_1, ρ_1'). Here ρ_0 and ρ_1 are signatures of the same type (Schnorr or Schnorr-like) issued with the secret key x; and ρ_0' and ρ_1' are signatures of the same type (Schnorr-like or Schnorr respectively) issued with the secret key x'. The challenger then chooses a random bit $b \in \{0, 1\}$ and set $\rho' = \rho_b'$. The signatures ρ_0, ρ_1 and ρ' are then given to the adversary along with the two private keys x and x'.

Output: Finally $\mathcal{A}$ outputs the bit b'. The adversary wins the game if $b = b'$.

Definition 6. *A concurrent signature scheme is unlinkable if the probability of an adversary to win the above game is not better than* $1/2$*.*

This definition captures the intuition that if signatures are unlinkable, the probability that the adversary to identify the correct pair (ρ_b, ρ_b') must not be non-negligibly better than a random toss of the coin. By letting the adversary to have the secret keys after the challenge is created, we essentially allow the adversary to have the complete view of all protocol runs thereafter. Here we made an implicit assumption that the messages signed are not linked.

5 An Asymmetric Concurrent Signature Scheme

We now proceed to give a concrete asymmetric concurrent signature scheme.

Setup: the public parameters are the primes p and q of appropriate size such that $q|(p-1)$, and a generator g for the subgroup in $\mathbb{Z}_p^*$ of order q. The space $\mathcal{M}$ is the set of all binary strings $\{0,1\}^*$, $K = \mathbb{Z}_q$ and F is the subgroup of $\mathbb{Z}_p^*$ generated by g. Private keys x_i's are chosen randomly from $\mathbb{Z}_q$. Their corresponding public keys are X_i's, each satisfies $X_i = g^{x_i}$. $KGEN$ is defined as $KGEN(x) = g^x$, each $KGEN_j$ is defined as $KGEN_j(k) = X_j^k$, and $KTRAN$ is defined as $KTRAN(s, x_j) = s^{x_j}$.

ISIGN: on input of $\langle X_i, x_i, M_i \rangle$, the algorithm generates a random $r \in \mathbb{Z}_q$ and returns the Schnorr promise of signature $\sigma_i = \langle s, c \rangle$ where $c = H(g^r, M_i)$, $s = g^{r+cx_i}$.

SSIGN: on input of $\langle X_j, x_j, M_j \rangle$ and s, the algorithm generates a random $r' \in \mathbb{Z}_q$ and returns the promise of Schnorr-like signature $\omega_j = \langle s', \kappa_1, c' \rangle$ where $s' = s^{x_j}$, $c' = H(g^{r'} s', M_j)$ and $\kappa_1 = (r' - c')/x_j \bmod q$.

IVERIFY: on input of $\sigma_i = \langle s, c\rangle$, outputs *accept* if $c = H(sX_i^{-c}, M_i)$ and *reject* otherwise.

SVERIFY: on input of $\omega_j = \langle s', \kappa_1, c'\rangle$, outputs *accept* if $c' = H(g^{c'} X_i^{\kappa_1} s', M_j)$ and *reject* otherwise.

VERIFY: on input of the keystone k and the promise of signature $\sigma_i = \langle s, c\rangle$ (or k and the promise of signature $\omega_j = \langle s', \kappa_1, c'\rangle$), the algorithm outputs *accept* if $KGEN(k) = s$ and $IVERIFY(\sigma_i) = accept$ (or respectively $KGEN_j(k) = s'$ and $SVERIFY(\omega_j) = accept$).

5.1 Security

The completeness of the above concurrent signature scheme is by inspection. It remains to show that our concurrent signature protocol is fair and anonymous and the converted signatures are unlinkable.

Theorem 1. *The concurrent signature protocol constructed from the above concurrent signature scheme is* fair*, provided that the Schnorr and Schnorr-like signature schemes are unforgeable.*

Proof. The proof is by contradiction. If the concurrent signature protocol is not fair, by definition it must violate one of these two conditions with non-negligible probability:

Case 1: the party B can obtain a signature $(k, \sigma) = (k, s, c)$ such that (k, s, c) is accepted by $VERIFY$ without getting the keystone k from A.

Case 2: the party A can obtain a signature $(k, \omega) = (k, s', \kappa_1, c')$ such that (k, s', κ_1, c') is accepted by $VERIFY$ while the output of party B $(k, \sigma) = (k, s, c)$ is not accepted by $VERIFY$, i.e., $VERIFY(k, s, c) \neq accept$.

We consider each of the two cases.

Case 1. B is able to obtain a valid signature (k, σ) such that (k, s, c) is accepted by $VERIFY$ without getting k from A. This is equivalent to B getting a valid signature (k, σ) from the promise of signature σ. This implies a Schnorr signature forgery.

Case 2. A is able to obtain a valid signature (k, ω). A could either receive ω from B or generate ω by herself. If A does not receive ω from B, A must generate the whole tuple (k, ω) by herself. This means that A could forge the Schnorr-like signature (k, ω) which contradicts the unforgeability property of the basic signature scheme.

If A receives the promise of signature $\omega = \langle s', \kappa_1, c'\rangle$, it means B must have obtained a promise of signature $\sigma = \langle s, c\rangle$ such that $s' = s^{x_B}$. Since $\langle k, \omega\rangle$ is a valid signature, we have $s' = X_B^k$. This means $s = s'^{1/X_B} = g^k$; and thus (k, σ) is a valid signature which contradicts the original assumption.

Theorem 2. *In the random oracle model, the concurrent signature protocol constructed from the above concurrent signature scheme is* anonymous *under the Decisional Diffie-Hellman assumption.*

Proof. By definition, our signature protocol is anonymous if these conditions are both satisfied:

Case 1: the promise of signature $\sigma_i = \langle s, c \rangle$ outputted by the algorithm *ISSIGN* on input (X_i, x_i, M_i) is indistinguishable from random elements of the signature space and,

Case 2: the tuple (s, ω_j) where $\omega_j = \langle s', \kappa_1, c' \rangle$ is the output of algorithm *SSIGN* on input (X_j, x_j, M_j, s) is indistinguishable from random elements of the signature space.

Case 1 comes straight from the anonymity property of the Schnorr promise of signature. For Case 2, we show that if there is an oracle which distinguishes (s, ω_j) from random elements of the signature space, there is a machine which could use the oracle to solve the Decisional Diffie-Hellman problem in the random oracle model.

Our machine interacts with the oracle as follows:

Input: a tuple of $\langle g, g^x, g^y, g^r \rangle$.

Operation 1: The machine serves as the hash function for the oracle. The machine maintains a list of previous hash query definitions. If a hash query is defined, the machine returns the previous output. Otherwise, the machine returns a random output and adds the entry pair to the definition list.

Operation 2: The machine chooses random c and κ_1 and sets $s = g^y, X_j = g^x, s' = g^z$, and $\omega_j = \langle s', \kappa, c \rangle$. The machine adds the pair of $(c, g^c X_j^{\kappa_1} s)$ to the list of hash function definition. Finally the machine sends the pair (s, ω_j) to the oracle. This operation could be initiated by the machine or by the oracle. We place no restriction on how this operation could be initiated.

Output: If the oracle outputs that one of $\{(s, \omega_j)\}$ is a valid promise of signature, the machine outputs $\langle g, g^x, g^y, g^r \rangle$ is a valid Diffie-Hellman tuple, i.e., $r = xy$.

If $\langle g, g^x, g^y, g^r \rangle$ is a Diffie-Hellman tuple then $r = xy$ and thus (s', ω_j) is a valid pair (with $k = x$). If $\langle g, g^x, g^y, g^r \rangle$ is not a Diffie-Hellman tuple then $log_s s' \neq x$ and thus (s', ω_j) is not a valid pair. Thus our machine will give a correct answer to the Decisional Diffie-Hellman problem if the machine would terminate and the oracle could distinguish a valid (s, ω_j) from an invalid one. The latter is by the assumption of the oracle. For the former, it remains to show that the machine terminates in a polynomially-bounded number of operations.

It is clear that Operation 1 always terminates. Operation 2 would not terminate only if the hash query is already defined for challenge $g^c X_j^{\kappa_1} s$, i.e., there is another value $c' \neq c$ such that $c' = H(g^c X_j^{\kappa_1} s)$ is already defined by the machine. With random c and κ_1, $g^c X_j^{\kappa_1} s$ is uniformly distributed and the probability that it appears in a polynomial-bounded list is negligible.

Theorem 3. *Assuming the messages are unrelated, when converted to become ordinary signatures, our concurrent signatures are unlinkable under the random oracle model.*

Proof. According to definition 4, to prove that our concurrent signatures are unlinkable, we need to show that the advantage of an adversary to identify the bit $b \in \{0,1\}$ in GAME 2 (section 4.4) over the random toss of the coin is negligible. Since the messages are irrelevant to our discussion, we ignore the messages in the proof whenever possible.

We shall provide the proof for the case where ρ_0 and ρ_1 are Schnorr signatures and ρ'_b is a Schnorr-like signature. The case where ρ_0 and ρ_1 are Schnorr-like signatures and ρ'_b is a Schnorr signature is similar and omitted.

Let $\rho_0 = (k_0, c_0)$ and $\rho_1 = (k_1, c_1)$ where $c_i = H(X^{-c_i} g^{k_i})$ $(i = 1, 2)$, and let $\rho'_b = (k'_b, c'_b)$ where $c'_b = H(X'^{k'_b} g^{c'_b})$.

Let $r_0 = k_0 - c_0 x \bmod q$ and $r'_0 = (k'_b - k_0)x_j + c'_b \bmod q$. Since c_0 and c'_b are outputs of the random oracle $H()$, the pair $\langle r_0, r'_0 \rangle$ is uniformly distributed in $\mathbb{Z}_q \times \mathbb{Z}_q$. It is easy to verify that the concurrent signature protocol in which r_0 is the random input to the algorithm $ISIGN(X, x, M_0)$ and r'_0 is the random input to the algorithm $SSIGN(X', x', M'_b)$ is legitimate and returns the signatures ρ_0 and ρ'_b.

Similarly, let $r_1 = k_1 - c_1 x \bmod q$ and $r'_1 = (k'_b - k_1)x_j + c'_b \bmod q$. Since c_1 and c'_b are outputs of the random oracle $H()$, the pair $\langle r_1, r'_1 \rangle$ is uniformly distributed in $\mathbb{Z}_q \times \mathbb{Z}_q$. It is likewise easy to verify that the concurrent signature protocol in which r_1 is the random input to the algorithm $ISIGN(X, x, M_1)$ and r'_1 is the random input to the algorithm $SSIGN(X', x', M'_b)$ is legitimate and returns the signatures ρ_1 and ρ'_b.

Then the probability for the adversary to identify the bit b is equal to the probability for the adversary to identify whether the random pair $\langle r_0, r'_0 \rangle$ or $\langle r_1, r'_1 \rangle$ is the actual pair $\langle r_b, r'_b \rangle$ used in the protocol which generates ρ_b and ρ'_b.

Since r_b and r'_b are uniformly and independently chosen random values from $\mathbb{Z}_q$, the pair $\langle r_b, r'_b \rangle$ is uniformly distributed in $\mathbb{Z}_q \times \mathbb{Z}_q$. This means the probability that $\langle r_b, r'_b \rangle = \langle r_0, r'_0 \rangle$ is equal to the probability that $\langle r_b, r'_b \rangle = \langle r_1, r'_1 \rangle$, i.e., the probability for the adversary to win GAME 2 is no better than a random toss of the coin.

5.2 Discussion

The benefit of anonymity that our concurrent signature scheme offers might turn out to be a disadvantage to the matching signer in certain cases. Due to the anonymity property, the matching signer might not be able to distinguish faked queries from genuine ones and hence is vulnerable against denial-of-service attacks. This problem could be overcome by forcing the initial signer to prove in zero-knowledge the knowledge of the homomorphic inverse. This is easily accomplished using the standard Schnorr identification protocol [9]. The trade-off is a reduction in the protocol performance. We note that this problem is not applicable to the original CKP scheme as due to the nature of ring signatures, the matching signer can always determine if a signature sent from the initial signer is genuine or faked.

The protocol could be extended to work with multiple matching signers. In this model, one initial signer is used as the hub connecting with all multiple matching signers. Each matching signer is assumed to behave like the the matching signer in the two-party case. It is easy to see that in this scenario, the anonymity of each and every signer is protected. Each concurrent signature in this case is still of the same size. When the keystone is revealed, all obtained signatures still remain unlinkable. We note that the CKP construction would require extra information proportional to the size of the group embedded in each signature. This results in concurrent signatures of the size proportional to the size of the group.

References

1. N. Asokan, V. Shoup, and M. Waidner, *Optimistic Fair Exchange of Digital Signatures* (Extended Abstract), In K. Nyberg (ed.), EUROCRYPT '98, Lecture Notes in Computer Science vol. 1403, Springer-Verlag, Berlin, 1998, pp. 591-606.
2. 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.
3. L. Chen, C.J. Kudla and K.G. Paterson, *Concurrent Signatures.* In C. Cachin and J. Camenisch (eds.), EUROCRYPT 2004, Lecture Notes in Computer Science vol. 3027, Springer-Verlag, 2004, pp. 287-305.
4. L.C. Guillou and J,-J. Quisquater, *A Practical Zero-Knowledge Protocol Fitted to Security Microprocessor Minimizing Both Transmission and Memory*, EUROCRYPT 1988, Lecture Notes in Computer Science, Springer-Verlag, 1989, pp. 123-128.
5. National Institute of Standards and Technology, NST FIPS PUB 186, *Digital Signature Standard*, U.S. Department of Commerce, May, 1994.
6. D. Pointcheval and J. Stern, *Security Arguments for Digital Signatures and Blind Signatures*, Journal of Cryptology, Volume 13 - Number 3, Springer-Verlag, 2000, pp. 361-396.
7. R. Rivest, A. Shamir, and L.M. Adleman, *A Method for Obtaining Digital Signatures and Public-Key Cryptosystems*, Communicaions of the ACM, vol. 21, no. 2, Feb 1978, pp. 120-126.
8. R. L. Rivest, A. Shamir, and Y. Tauman, *How to Leak a Secret*, C. Boyd (Ed.), ASIACRYPT 2001, Lecture Notes in Computer Science vol. 2248, Springer-Verlag, 2001, pp. 552-565.
9. C.P. Schnorr, *Efficient signature generation by smart cards*, In Journal of Cryptology, vol. 4, no. 3, 1991, pp. 161-174.
10. W. Susilo, Y. Mu and F. Zhang, *Perfect Concurrent Signature Schemes*, ICICS 2004, Lecture Notes in Computer Science vol. 3269, Springer-Verlag, 2004, pp. 14-27.