Generic Construction of (Identity-Based) Perfect Concurrent Signatures

Sherman S.M. Chow[1] and Willy Susilo[2]

[1] Department of Computer Science,
Courant Institute of Mathematical Sciences,
New York University, NY 10012, USA
schow@cs.nyu.edu

[2] Center for Information Security Research,
School of Information Technology and Computer Science,
University of Wollongong, Wollongong 2522, Australia
wsusilo@uow.edu.au

Abstract. The notion of concurrent signatures was recently introduced by Chen, Kudla and Paterson. In concurrent signature schemes, two entities can produce two signatures that are *not* binding, until an extra piece of information (namely the *keystone*) is released by one of the parties. Subsequently, it was noted that the concurrent signature scheme proposed in the seminal paper cannot provide perfect ambiguity. Then, the notion of *perfect* concurrent signatures was introduced. In this paper, we define the notion of *identity-based (or ID-based) perfect concurrent signature schemes*. We provide the *first* generic construction of (ID-based) perfect concurrent signature schemes from ring signature schemes. Using the proposed framework, we give two concrete ID-based perfect concurrent signature schemes based on two major paradigms of ID-based ring signature schemes. Security proofs are based on the random oracle model.

Keywords: Concurrent Signatures, Perfect Ambiguity, Fair-Exchange, Ring Signatures, Identity-based Signatures, Bilinear Pairing.

1 Introduction

Consider the situation where a customer Alice would like to make a purchase request of a physical item from a shop owner Bob. One of the ways to do the transaction is asking Alice to firstly sign a payment instruction to pay Bob the price of the item. Then, Bob agrees by signing a statement that he authorizes Alice to pick the item up from the store, which will be sent via an email or other means upon receiving Alice's signature. We would like to make sure that both parties (the customer and the shop owner in our case) get the other party's item, or no party gets the other party's item at the end of a transaction, that is, the principle of fair exchange. For purchase occurred in a face-to-face manner, people have a higher confidence in getting back the other party's item shortly after giving out his or her item to be exchanged. However, to achieve fair exchange over Internet, in which two parties are mutually distrustful, is not a trivial task.

Concurrent signature can help when the full power of fair exchange is not necessary [6]. A pair of concurrent signatures can be made binding at the same time, i.e. when Alice picks up the item from Bob's store. At this time, Alice's signature (i.e. payment instruction) will be binding and Bob's signature (to allow Alice to pick up the item) will also be binding concurrently.

Subsequently, [13] noted that the concurrent signature scheme proposed in [6] cannot provide perfect ambiguity if both signers are known to be trustworthy. With the aim to further anonymize the signatures before the signatures are made binding, the notion of *perfect* concurrent signatures was introduced.

1.1 Related Work

Fair exchange of signature is a fundamental research problem in cryptography. Fairness in exchanging signatures is normally achieved with the help of a trusted third party (TTP) (which is often offline [2]). There were some attempts where a fair exchange of signatures can be achieved with a "semi-trusted" TTP who can be called upon to handle disputes between signers [1, 9]. This type of fair exchange is also referred to as an optimistic fair exchange. The well-known open problem in fair exchange is the requirement of a dispute resolving TTP whose role cannot be replaced by a normal certification authority (CA).

In [12], the notion of *ring signatures* was formalized and an efficient scheme based on RSA was proposed. A ring signature scheme allows a signer who knows at least one piece of secret information (or a trapdoor) to produce a sequence of n random permutations and form them into a ring. This ambiguous signature can be used to convince any third party that one of the people in the group (who knows the trapdoor information) has authenticated the message on behalf of the group. The authentication provides *signer ambiguity*, in the sense that no one can identify who has actually signed the message. The ID-based version of ring signature schemes was introduced in [14]. After that, a number of ID-based ring signature schemes were proposed. A recent study [7] showed that these schemes can be classified into two major paradigms, namely, the conversation from non-ID-based ring signature and the extension from ID-based signature. Please refer to [7] for a more detailed review of ID-based ring signature schemes.

1.2 Our Contributions

We define the notion of ID-based perfect concurrent signatures, which is the strongest notion (in terms of privacy) of concurrent signature currently available. We provide a generic construction of both non-ID-based and ID-based perfect concurrent signature schemes from ring signatures, which is the first discussion in the literature. We illustrate our idea by two schemes from each of two major paradigms of existing ID-based ring signature schemes. Both of them enjoy short signature length which is only one group element on elliptic curve larger than most existing ID-based signature schemes, our second scheme is also efficient in the sense that no pairing operation is required for the generation of signature.

1.3 Paper Organization

The rest of this paper is organized as follows. The next section reviews some notions that will be used throughout this paper. Section 3 provides a model of ID-based perfect concurrent signature schemes together with its security requirements. We also present a generic construction of (ID-based) perfect concurrent signature protocol in this section. In Section 4 and Section 5, we provide two concrete ID-based perfect concurrent signature schemes. Section 6 concludes the paper and discusses future research direction.

2 Preliminaries

2.1 Basic Concepts on Bilinear Pairings

Let $\mathbb{G}_1, \mathbb{G}_2$ be cyclic additive groups generated by P_1, P_2, respectively, whose order are a prime q. Let $\mathbb{G}_M$ be a cyclic multiplicative group with the same order q. We assume there is an isomorphism $\psi : \mathbb{G}_2 \to \mathbb{G}_1$ such that $\psi(P_2) = P_1$. Let $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_2 \to \mathbb{G}_M$ be a bilinear mapping with the following properties:

1. *Bilinearity*: $\hat{e}(aP, bQ) = \hat{e}(P, Q)^{ab}$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2, a, b, \in \mathbb{Z}_q$.
2. *Non-degeneracy*: There exists $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$ such that $\hat{e}(P, Q) \neq 1$.
3. *Computability*: There exists an efficient algorithm to compute $\hat{e}(P, Q)$ for all $P \in \mathbb{G}_1, Q \in \mathbb{G}_2$.

For simplicity, hereafter, we set $\mathbb{G}_1 = \mathbb{G}_2$ and $P_1 = P_2$. We note that our scheme can be easily modified for a general case, when $\mathbb{G}_1 \neq \mathbb{G}_2$.

A bilinear pairing instance generator is defined as a probabilistic polynomial time algorithm $\mathcal{IG}$ that takes as input a security parameter ℓ and returns a uniformly random tuple $param = (p, \mathbb{G}_1, \mathbb{G}_M, \hat{e}, P)$ of bilinear parameters, including a prime number p of size ℓ, a cyclic additive group $\mathbb{G}_1$ of order q, a multiplicative group $\mathbb{G}_M$ of order q, a bilinear map $\hat{e} : \mathbb{G}_1 \times \mathbb{G}_1 \to \mathbb{G}_M$ and a generator P of $\mathbb{G}_1$. For a group $\mathbb{G}$ of prime order, we denote the set $\mathbb{G}^* = \mathbb{G} \setminus \{\mathcal{O}\}$ where $\mathcal{O}$ is the identity element of the group.

2.2 Complexity Assumption

Definition 1. Computational Co-Diffie-Hellman (Co-CDH) Problem.
Given a randomly chosen (P_1, P_2, aP_1, bP_2)*, where* $P_1, P_2 \in \mathbb{G}_1$*,* $a, b \in \mathbb{Z}_q^*$*, and* a, b *are unknown, compute* $abP_2 \in \mathbb{G}_M$*.*

Definition 2. Co-CDH Assumption.
If $\mathcal{IG}$ *is a Co-CDH parameter generator, the advantage* $\texttt{Adv}_{\mathcal{IG}}(\mathcal{A})$ *that an algorithm* $\mathcal{A}$ *has in solving the Co-CDH problem is defined to be the probability that the algorithm* $\mathcal{A}$ *outputs* abP_2 *on inputs* $\mathbb{G}_1, \mathbb{G}_M, P_1, P_2, aP_1, bP_2$*, where* $(\mathbb{G}_1, \mathbb{G}_M)$ *is the output of* $\mathcal{IG}$ *for sufficiently large security parameter* ℓ*,* P_1, P_2 *are random generators of* $\mathbb{G}_1$ *and* a, b *are random elements of* $\mathbb{Z}_q^*$*. The Co-CDH assumption is that* $\texttt{Adv}_{\mathcal{IG}}(\mathcal{A})$ *is negligible for all efficient algorithms* $\mathcal{A}$*.*

2.3 Review on Concurrent Signatures

In concurrent signatures, there are two parties involved in the protocol, namely A and B (or Alice and Bob, respectively). At first, both parties' signatures are ambiguous from any third party's point of view, but they will be simultaneously binding *after* an additional information, called a *"keystone"* is released by one of the participants. Since one party is required to create a keystone and send the first message to the other party, we call this party the *initial signer*. A party who responds to the initial signature by creating another signature is called a *matching signer*. We note that if Alice does not release the keystone, then the transaction cannot be completed, although Bob would like to do so. Nevertheless, there are many scenarios where this type of signature schemes is applicable [6].

Similar to the definition in [6], concurrent signatures are digital signature schemes that consist of the following algorithms:

- SETUP: A probabilistic algorithm that accepts a security parameter ℓ, outputs the descriptions of the message space $\mathcal{M}$, the signature space $\mathcal{S}$, the keystone space $\mathcal{K}$, the keystone fix space $\mathcal{F}$, a function $KSGEN : \mathcal{K} \to \mathcal{F}$ and any other parameters π.
- KEYGEN: A probabilistic algorithm that accepts a security parameter ℓ, outputs the public key y_i; together with the corresponding private key x_i to be kept secretly.
- ASIGN: A probabilistic algorithm that accepts $(y_i, y_j, x_i, h_1, h_2, m)$, where $h_1, h_2 \in \mathcal{F}$, y_i, $y_j \neq y_i$ are public keys, x_i is the private key corresponding to y_i, and $m \in \mathcal{M}$, outputs a *signer-ambiguous* signature $\sigma = (s, h_1, h_2)$ where $s \in \mathcal{S}$, and h_1, h_2 are the sources of randomness used in generating s.
- $\mathsf{AVERIFY}$: An algorithm that accepts $S = (\sigma, y_i, y_j, m)$, where $\sigma = (s, h_1, h_2)$, $s \in \mathcal{S}$, $h_1, h_2 \in \mathcal{F}$, y_i and y_j are public keys, and $m \in \mathcal{M}$, outputs `accept` or `reject`. The symmetric property of $\mathsf{AVERIFY}$ requires $\mathsf{AVERIFY}(\sigma', y_j, y_i, m) = \mathsf{AVERIFY}(\sigma, y_i, y_j, m)$ for $\sigma' = (s, h_2, h_1)$.
- VERIFY: An algorithm that accepts (k, S) where $k \in \mathcal{K}$ is a keystone and S is of the form $S = (\sigma, y_i, y_j, m)$, where $\sigma = (s, h_1, h_2)$ with $s \in \mathcal{S}$, $h_1, h_2 \in \mathcal{F}$, y_i and y_j are public keys, and $m \in \mathcal{M}$. The algorithm verifies whether $KSGEN(k) \stackrel{?}{=} h_2$ holds. If it does not hold, then it terminates with output `reject`. Otherwise, it runs $\mathsf{AVERIFY}(S)$.

As discussed in the introduction, the concrete construction of concurrent signature schemes in [6] cannot provide *perfect* ambiguity in certain situations. In their scheme, the two signatures have an explicit relationship which can be easily observable by any third party. As a consequence, when the two signers are well known to be honest that will always conform to the protocol, then any third party would trust that the signatures are valid. Since the signatures can be identified even *before* the keystone is released, it contradicts with the requirement of concurrent signatures. Concurrent signature schemes with perfect ambiguity was considered in [13]. They presented two schemes based on the discrete logarithm problem and bilinear pairings. Their constructions are based on the framework proposed by [6], and they have not considered the generic construction of perfect concurrent signature schemes.

3 Generic Framework and Security Notions

We note that the algorithms listed out by [6] may not be enough to cater for the need of perfect ambiguity. In view of this, we provide a new generic framework.

3.1 Building Blocks

Firstly, we provide a formal definition of the algorithm used in our generic construction of perfect concurrent signature schemes, by incorporating some elements from the notion introduced in [6]. Notice that to achieve the perfect ambiguity, we no longer require the matching signer to use the same keystone fix as the initial signer. Beside, a pair of keystones is used instead of a single one. We also describe the essential properties of these algorithms for the construction of perfect concurrent signature schemes.

Definition 3. *A perfect concurrent signature scheme is a digital signature scheme that consists of the following algorithms:*

- SETUP*: A probabilistic algorithm that on input a security parameter ℓ, outputs the system parameters params which is the descriptions of the the message space $\mathcal{M}$, the signature space $\mathcal{S}$, the keystone-pair space $\mathcal{K}_I \times K_M$, the keystone fix space $\mathcal{F}$ and the encrypted keystone space $\mathcal{K}'$. Note that we do not include params explicitly as the input in the following descriptions.*
- KEYGEN*: A probabilistic algorithm that is invoked by a participant ID. The algorithm outputs a public key Q_{ID} and the corresponding the secret key $\mathcal{S}_{\mathsf{ID}}$.*
- $\mathsf{FIX\text{-}INITIAL\text{-}KEYSTONE}$*: A deterministic algorithm that on input a initial-keystone $k_I \in \mathcal{K}_I$, it outputs the corresponding keystone fix $f_I \in \mathcal{F}$.*
- ASIGN*: A probabilistic algorithm that on inputs $(\mathsf{ID}_i, \mathsf{ID}_j, \mathcal{S}_{\mathsf{ID}_i}, \alpha, f, m)$, where $\alpha, f \in \mathcal{F}$, $\mathsf{ID}_i, \mathsf{ID}_j$ are the identities of the participants, $\mathcal{S}_{\mathsf{ID}_i}$ is the secret key associated with ID_i, and $m \in \mathcal{M}$, outputs an ambiguous signature $\sigma = \{U_i, U_j, V\}$ on m.*
- $\mathsf{ENC\text{-}MATCHING\text{-}KEYSTONE}$*: A deterministic algorithm that on input a matching-keystone $k_M \in \mathcal{K}_M$, it outputs the encrypted matching-keystone $K_M \in \mathcal{K}'$.*
- $\mathsf{FIX\text{-}SECRET\text{-}KEYSTONE}$*: A deterministic algorithm that on inputs an encrypted matching-keystone $K_M \in \mathcal{K}'$ and a secret key $\mathcal{S}_{\mathsf{ID}_i}$, outputs a secret keystone fix $f_S \in \mathcal{F}$.*
- $\mathsf{AVERIFY}$*: A deterministic algorithm that takes as input $S = (\sigma, \mathsf{ID}_i, \mathsf{ID}_j, m)$ and outputs* `accept` *or* `reject`*. Again we require a symmetric property that $\mathsf{AVERIFY}(\sigma, \mathsf{ID}_i, \mathsf{ID}_j, m) = \mathsf{AVERIFY}(\sigma', \mathsf{ID}_j, \mathsf{ID}_i, m)$ for $\sigma' = \{U_j, U_i, V\}$.*
- $\mathsf{VERIFY\text{-}INITIAL\text{-}KEYSTONE}$*: A deterministic algorithm that on input an initial-keystone $k_I \in \mathcal{K}_I$ and its corresponding fix $k_I \in \mathcal{F}$, it outputs* `accept` *or* `reject` *by checking $f_I \stackrel{?}{=} \mathsf{FIX-INITIAL-KEYSTONE}(k_I)$.*
- $\mathsf{VERIFY\text{-}SECRET\text{-}KEYSTONE}$*: A deterministic algorithm that on input a matching-keystone $k_M \in \mathcal{K}_M$, a secret keystone fix $f_S \in \mathcal{F}$ and an identity ID_i, it outputs* `accept` *or* `reject` *depending whether $f_S = \mathsf{FIX\text{-}SECRET\text{-}KEYSTONE}(\mathsf{ENC\text{-}MATCHING\text{-}KEYSTONE}(k_M), \mathcal{S}_{\mathsf{ID}_i})$.*

- $\mathsf{VERIFY\text{-}CONNECTION}$*: A deterministic algorithm that on input a pair of signatures* $\sigma_i = \{U_i, U_j, V\}$ *and* $\sigma_j = \{U_i', U_j', V'\}$ *and a pair of keystone fix* f_I *and* f_S*, it outputs* `accept` *or* `reject` *depending whether* $U_j = f_I$ *and* $U_i' = U_j \otimes f_S$*, where* $\otimes$ *is the operator of the group* $\mathcal{F}$*.*
- VERIFY*: A deterministic algorithm that takes as input* (k_I, k_M, S')*, where* $(k_I, k_M) \in \mathcal{K}_I \times \mathcal{K}_M$*,* $S' = (\sigma_i, \sigma_j, \mathsf{ID}_i, \mathsf{ID}_j, m)$*. The algorithm verifies if all of* VERIFY-INITIAL-STONE, VERIFY-SECRET-KEYSTONE *and* VERIFY-CONNECTION *are true. If not, it terminates with output* `reject`*. Otherwise, it runs* $\mathsf{AVERIFY}(S)$ *and the output of this algorithm is the output of the* AVERIFY *algorithm.*

3.2 ID-Based Scenario

For ID-based perfect concurrent signature, we need to modify the SETUP algorithm described and replace KEYGEN algorithm by a new EXTRACT algorithm in the above definition.

Definition 4. *An ID-based perfect concurrent signature scheme requires the following algorithms:*

- SETUP*: A probabilistic algorithm that on input a security parameter* ℓ*, outputs descriptions of the set of participants* $\mathcal{U}$*, the message space* $\mathcal{M}$*, the signature space* $\mathcal{S}$*, the keystone-pair space* $\mathcal{K}_I \times K_M$*, the keystone fix space* $\mathcal{F}$*, and the encrypted keystone space* $\mathcal{K}'$*. The algorithm also outputs the public key of the private key generator (PKG) and the master secret key of the PKG for the extraction of user's private key.*
- EXTRACT*: A deterministic algorithm that is invoked by a participant and the PKG. On input an* ID *of a participant, the algorithm outputs a participant's secret key* $\mathcal{S}_{\mathsf{ID}}$*.*

3.3 Generic Construction

In this section, we describe a generic construction of (ID-based) concurrent signature protocol. We highlight the properties of the algorithm involved. There are two parties, namely A (Alice) and B (Bob) that are involved in the protocol. Without losing generality, we assume that A is the initial signer and B is the matching signer. The protocol works as follows.

Firstly, CA/PKG runs the SETUP algorithm to determine the public parameters of the scheme. Then, depending on whether the scheme is ID-based, user invokes the corresponding algorithm to get the public-private key pair.

More specifically, for non-ID-based scenario, both A and B run KEYGEN to generate a public-private key pair (denoted by $(\mathsf{Q}_{\mathsf{ID}_A}, \mathcal{S}_{\mathsf{ID}_A})$ and $(\mathsf{Q}_{\mathsf{ID}_B}, \mathcal{S}_{\mathsf{ID}_B})$ respectively), register the public key and the identity with the CA, and possibly provides a proof-of-knowledge of the private key to the CA as well. After authentication (and the checking of the proof-of-knowledge) the CA issues a digital certificate binding the relation of the purported identity to the user.

For the ID-based scenario, both A and B visit the PKG and engage in the $\mathsf{EXTRACT}$ algorithm to obtain their secret key $\mathcal{S}_{\mathsf{ID}_A}$ and $\mathcal{S}_{\mathsf{ID}_B}$, respectively. The identities of A and B are available publicly as ID_A and ID_B, together with public hash functions $H_0 : \{0,1\}^* \to \mathbb{G}_1$. Hence, the public key $\mathsf{Q}_{\mathsf{ID}_i}$ can be computed by anyone (for instance, by computing $\mathsf{Q}_{\mathsf{ID}_i} = H_0(\mathsf{ID}_i)$).

After both users got their corresponding key pair, the protocol is as follows.

1. A picks a random keystone $(k_I, k_M) \in K_I \times K_M$ and executes the $\mathsf{SET\text{-}INITIAL\text{-}KEYSTONE}$ algorithm using k_I as the input to obtain $f_I \in \mathcal{F}$.
   A good candidate for $\mathsf{SET\text{-}INITIAL\text{-}KEYSTONE}$ is a cryptographic hash function, since it is hard to invert, i.e. given y from the range of the function, it is hard to find the pre-image x.
2. A selects a message $m_A \in \mathcal{M}$, together with her identity ID_A and B's identity ID_B, and computes her ambiguous signature as $\sigma_A = \{U_A, U_B, V\} \leftarrow \mathsf{ASIGN}(\mathsf{ID}_A, \mathsf{ID}_B, \mathcal{S}_{\mathsf{ID}_A}, \mathcal{O}_{\mathcal{F}}, f_I, m_A)$ where $\mathcal{O}_{\mathcal{F}}$ denotes the identity element of the group $\mathcal{F}$. ($\mathcal{O}_F$ is used to unify the list of input parameters used by ID_A and ID_B for the ASIGN algorithm, which merely means that A can skip a certain group operation inside the ASIGN algorithm that is used to connect B's signature with A's.)
   We require that the ASIGN algorithm to be able to produce ambiguous signature σ such that any one can get convinced that either $\mathcal{S}_{\mathsf{ID}_A}$ or $\mathcal{S}_{\mathsf{ID}_B}$ is used as the input but does not know exactly which one with probability greater than 1/2. Moreover, there are two parts (which can be implicitly) involved with the signature such that the first part can be chosen arbitrary while the value of another part must be depending on the first part. Most of existing ring signature schemes satisfy these properties.
3. A hides the second keystone k_M by executing the $\mathsf{ENC\text{-}MATCHING\text{-}KEYSTONE}$ algorithm using k_M as the input to obtain $K_M \in \mathcal{K}'$, A then sends K_M and σ_A to B. We require that k_M cannot be effectively computable from K_M. The choice of implementation for $\mathsf{ENC\text{-}MATCHING\text{-}KEYSTONE}$ will be discussed shortly afterward.
4. Upon receiving A's ambiguous signature σ_A, B verifies the signature by testing whether $\mathsf{AVERIFY}(\sigma_A, \mathsf{ID}_A, \mathsf{ID}_B, m_A) \overset{?}{=} \mathtt{accept}$ holds. Obviously, the $\mathsf{AVERIFY}$ algorithm is simply the one matching with the ASIGN algorithm.
5. B aborts if the above equation does not hold. Otherwise, B picks a message $m_B \in \mathcal{M}$ to sign. B firstly executes $\mathsf{FIX\text{-}SECRET\text{-}KEYSTONE}$ with the encrypted matching keystone K_M and his secret key $\mathcal{S}_{\mathsf{ID}_B}$ as input to obtain a secret matching keystone fix f_S, then computes his ambiguous message $\sigma_B = \{U'_A, U'_B, V'\} \leftarrow \mathsf{ASIGN}(\mathsf{ID}_B, \mathsf{ID}_A, \mathcal{S}_{\mathsf{ID}_B}, U_B, f_S, m_B)$ and sends this value to A. We require that the value of f_S is uniquely determined by k_M and $\mathcal{S}_{\mathsf{ID}_B}$ and cannot be computed without the knowledge of $\mathcal{S}_{\mathsf{ID}_B}$ or k_M (that is why the keystone-fix is called a secret), yet its correctness can be checked by only knowing k_M. All these properties can be achieved by probabilistic public key encryption, such that k_M is the randomness used in encryption, K_M is part of the ciphertext, which can be viewed as a kind of

session key employed by probabilistic public key encryption. The value of f_S can be verified by using the knowledge of k_M and the recipient's public key $\mathsf{Q}_{\mathsf{ID}_B}$.

6. Upon receiving B's ambiguous signature σ_B, A verifies it by testing whether
   - $\mathsf{VERIFY\text{-}SECRET\text{-}KEYSTONE}(k_M, f_S, \mathsf{Q}_{\mathsf{ID}_B}) \overset{?}{=} \texttt{accept}$,
   - $\mathsf{VERIFY\text{-}CONNECTION}(f_I, f_S, \sigma_A, \sigma_B) \overset{?}{=} \texttt{accept}$ and
   - $\mathsf{AVERIFY}(\sigma_B, \mathsf{ID}_B, \mathsf{ID}_A, m_B) \overset{?}{=} \texttt{accept}$

   all hold. If not, then A aborts. Otherwise, A releases the keystone (k_I, k_M) to B, and both signatures are binding concurrently.

3.4 Security Notions

As the original model of concurrent signatures in [6], we require a perfect concurrent signatures (either ID-based or not) to satisfy *correctness*, *ambiguity*, *unforgeability* and *fairness*. Intuitively, these notions are described as follows. Note that we follow the definition of ambiguity in [13] instead of the one in [6].

- *Correctness*: If a signature σ has been generated *correctly* by invoking ASIGN algorithm on a message $m \in \mathcal{M}$, $\mathsf{AVERIFY}$ algorithm will return "`accept`" with an overwhelming probability, given a signature σ on m and a security parameter ℓ. Moreover, after the keystone-pair $(k_I, k_M) \in \mathcal{K}_I \times \mathcal{K}_M$, is released, then the output of VERIFY algorithm will be "`accept`" with an overwhelming probability.
- *Ambiguity*: We require that given the two ambiguous signatures (σ_1, σ_2), any adversary will not be able to distinguish who was the actual signer of the signatures *before* the keystone is released. Any adversary can only conclude that one of the following events has occurred:
  1. Both σ_1 and σ_2 were generated by the initial signer.
  2. Both σ_1 and σ_2 were generated by the matching signer.
  3. The initial signer generated σ_1 while the matching signer generated σ_2.
  4. The matching signer generated σ_1 while the initial signer generated σ_2.

  All these cases are equally probable from the adversary's view.
- *Unforgeability*: There are two levels of unforgeability to be considered.
  - *Level 1*: When an adversary $\mathcal{A}$ does not have any knowledge of the respective secret key $\mathcal{S}_{\mathsf{ID}}$, then no valid signature that will pass the $\mathsf{AVERIFY}$ algorithm can be produced. Otherwise, one of the underlying hard problems can be solved by using this adversary's capability. This requirement is for the matching signer to get convinced that the signature presented by the initial signer is indeed originated from her.
  - *Level 2*: Any party cannot *frame* the other party that he or she has indeed signed a message. We require that although both signatures are ambiguous, any party who would like to frame (or cheat) the others will not be able to produce a valid keystone with an overwhelming probability. This means that the first signature can only be generated by the initial signer and it is unforgeable by anyone else, including the matching

signer. At the same time, the second signature can only originate from the matching signer, which is unforgeable by any person other than him, including the initial signer.

– *Fairness*: We require that any valid ambiguous signatures generated using the same keystone will all become binding *after* the keystone is released. Hence, a matching signer cannot be left in a position where a keystone binds his signature to him whilst the initial signer's signature is not binding to her. This requirement is important for the case like the initial signer try to present a signature of another message after the matching signer has verified the validity of the original message and complete his part of protocol. However, we do not require that the matching signer will definitely receive the necessary keystone.

Definition 5. *An ID-based perfect concurrent signature scheme is* secure *if it is existentially unforgeable under a chosen message attack, ambiguous and fair.*

4 A Concrete Instantiation

We present a concrete ID-based perfect concurrent signature scheme using the above general construction, with the ID-based ring signature scheme proposed by Zhang and Kim [14] and the basic version of ID-based encryption scheme with semantic security proposed by Boneh and Franklin [4]. Using our generic construction in Section 3, we define the required eleven algorithms.

– SETUP: The PKG selects a random number $s \in \mathbb{Z}_q^*$ and sets $P_{pub} = sP$. It selects three cryptographic hash functions $H_0 : \{0,1\}^* \to \mathbb{G}_1$ and $H_1 : \{0,1\}^* \to \mathbb{Z}_q$. and $H_2 : \{0,1\}^* \to \mathbb{G}_1$. It publishes system parameters $params = \{\mathbb{G}_1, \mathbb{G}_M, \hat{e}, q, P, P_{pub}, H_0, H_1, H_2\}$, and keeps s as the *master secret key*. The algorithm also sets $\mathcal{M} = \mathcal{K}_I = \mathcal{K}_M = \mathcal{F} = \mathbb{Z}_q$ and $\mathcal{K}' = \mathbb{G}_1$.

– EXTRACT: The EXTRACT algorithm is defined as follows.
  1. A user $\mathcal{U}_i$ submits his or her identity ID_i to the PKG.
  2. After a successful identification, PKG generates $\mathcal{U}_i$ secret key as follows.
     - Compute $\mathsf{Q}_{\mathsf{ID}_i} = H_0(\mathsf{ID}_i)$.
     - Compute $\mathcal{U}_i$'s secret key as $\mathcal{S}_{\mathsf{ID}_i} = s\mathsf{Q}_{\mathsf{ID}_i}$.
     - Deliver $\mathcal{S}_{\mathsf{ID}_i}$ as user $\mathcal{U}_i$'s secret key through a private and authenticated channel.

– FIX-INITIAL-KEYSTONE: Assuming a keystone $k_I \in \mathbb{Z}_q$ is randomly selected, this algorithm outputs $f_I = H_1(k_I)$ as the keystone-fix.

– ASIGN: The ASIGN algorithm accepts the following parameters $(\mathsf{ID}_i, \mathsf{ID}_j, \mathcal{S}_{\mathsf{ID}_i}, \alpha, f, m)$, where $\mathcal{S}_{\mathsf{ID}_i}$ is the secret key associated with $\mathsf{Q}_{\mathsf{ID}_i}$, $f \in \mathcal{F}$ and $m \in \mathcal{M}$ is the message. The algorithm will perform the following.
  1. Select a random point $Z \in \mathbb{G}_1^*$.
  2. Set $u_j \leftarrow \alpha \cdot f$.
  3. Compute $u_0 = H_1\big(H_2(m)||(\mathsf{ID}_i \oplus \mathsf{ID}_j)||\hat{e}(Z,P)\hat{e}(u_j\mathsf{Q}_{\mathsf{ID}_j}, P_{pub})\big)$.
  4. Compute $V = u_0^{-1}(Z - (u_0 - u_j)\mathcal{S}_{\mathsf{ID}_i})$.
  5. Output $\sigma = (u_i = u_0 - u_j, u_j, V)$ as the signature on message m.

- ENC-MATCHING-KEYSTONE: Assuming a keystone $k_M \in \mathbb{Z}_q$ is randomly selected, this algorithm outputs $K_M = k_M P$ as the encrypted keystone.
- FIX-SECRET-KEYSTONE: This algorithm returns $H_1(\hat{e}(K_M, \mathcal{S}_{\mathsf{ID}_j}))$.
- AVERIFY. The AVERIFY algorithm accepts $(\sigma, \mathsf{ID}_i, \mathsf{ID}_j, m)$, where $\sigma = (u_i, u_j, V)$. The algorithm verifies whether

$$u_i + u_j \stackrel{?}{=} H_1\left(H_2(m)||(\mathsf{ID}_i \oplus \mathsf{ID}_j)||\hat{e}(V,P)^{(u_i+u_j)}\hat{e}(u_i\mathsf{Q}_{\mathsf{ID}_i}, P_{pub})\hat{e}(u_j\mathsf{Q}_{\mathsf{ID}_j}, P_{pub})\right)$$

  holds with equality. If so, then output `accept`. Otherwise, output `reject`.
- VERIFY-INITIAL-KEYSTONE: This algorithm outputs `accept` if $f_I = H_1(k_I)$, `reject` otherwise.
- VERIFY-SECRET-KEYSTONE: It outputs `accept` if $f_S = H_1(\hat{e}(P_{pub}, \mathsf{Q}_{\mathsf{ID}_j})^{k_M})$, `reject` otherwise.
- VERIFY-CONNECTION: This algorithm outputs `accept` if $U_i' = U_j \cdot f_S$, `reject` otherwise.
- VERIFY. The algorithm accepts (k_I, k_M, S'), where $k_I \in \mathcal{K}_I$ and $k_M \in \mathcal{K}_M$ are the keystones and $S' = (m, \sigma_i, \sigma_j, \mathsf{ID}_i, \mathsf{ID}_j)$, where $\sigma = (u_i, u_j, V)$. The algorithm verifies whether (k_I, k_M) is valid, by using the above two algorithm. If it does not hold, then output `reject`. Otherwise, run AVERIFY(S). The output of VERIFY is the output of AVERIFY algorithm.

Correctness.
The correctness of the above proposed scheme is justified as follows.

$$\begin{aligned}
u_i + u_j &= H_1\left(H_2(m)||(\mathsf{ID}_i \oplus \mathsf{ID}_j)||\hat{e}(V,P)^{(u_i+u_j)}\hat{e}(u_i\mathsf{Q}_{\mathsf{ID}_i}, P_{pub})\hat{e}(u_j\mathsf{Q}_{\mathsf{ID}_j}, P_{pub})\right)\\
u_0 &= H_1\left(H_2(m)||(\mathsf{ID}_i \oplus \mathsf{ID}_j)||\hat{e}((u_i+u_j)V + u_i\mathcal{S}_{\mathsf{ID}_i}, P)\hat{e}(u_j\mathsf{Q}_{\mathsf{ID}_j}, P_{pub})\right)\\
&= H_1\left(H_2(m)||(\mathsf{ID}_i \oplus \mathsf{ID}_j)||\hat{e}(u_0V + (u_0-u_j)\mathcal{S}_{\mathsf{ID}_i}, P)\hat{e}(u_j\mathsf{Q}_{\mathsf{ID}_j}, P_{pub})\right)\\
&= H_1\left(H_2(m)||(\mathsf{ID}_i \oplus \mathsf{ID}_j)||\hat{e}(Z,P)\hat{e}(u_j\mathsf{Q}_{\mathsf{ID}_j}, P_{pub})\right) \qquad \blacksquare
\end{aligned}$$

4.1 Security Consideration

The security proofs are omitted due to space limitation. We refer the reader to the full version of this paper for a more complex account.

Theorem 1. (Ambiguity) *Before the keystone k is released, both signatures are ambiguous.*

Lemma 1. *When the output of* VERIFY *is* `accept`*, then any third party can be sure who has generated the signature. Any party cannot* frame *that the other party has signed a message without his or her consent assuming the one-way property of the hash function. This guarantees that the signature is unforgeable.*

Theorem 2. (Unforgeability) *The scheme presented in this section is existentially unforgeable under a chosen message attack in the random oracle model, assuming the one-way property of the hash function, the hardness of the discrete logarithm problem and the Co-CDH assumption.*

Theorem 3. (Fairness) *For all signatures that are generated with the same keystone will be binding concurrently when the keystone is released.*

Theorem 4. *Our ID-based perfect concurrent signature scheme presented in this scheme is* secure *in the random oracle model, assuming the hardness of the discrete logarithm problem.*

4.2 Signature Length

In the above scheme, each signature is a three-tuple $\sigma_i = (u_1, u_2, V)$, where $u_1, u_2 \in \mathbb{Z}_q$ and $V \in \mathbb{G}_1$. Using any of the families of curves described in [5], one can take q to be a 170-bit prime and use a group $\mathbb{G}_1$ where each element is 171 bits. For example, $\mathbb{G}_1$ is derived from the curve $E/GF(3^{97})$ defined by $y^2 = x^3 - x + 1$, which has 923-bit discrete-logarithm security. With these choices, the total signature length for a pair of signature is 1,022 bits or 128 bytes.

5 A More Efficient Construction

Now we present a more efficient variant of ID-based perfect concurrent signature which requires no pairing operation in signing without sacrificing the computational efficiency of verification or other steps. Again, the construction follows our idea of generic construction in Section 3. We utilize the ID-based ring signature scheme proposed by Chow *et al.* [8] and also the basic version of ID-based encryption scheme with semantic security proposed in [4].

- SETUP: Basically it is the same as our first scheme, but the description of spaces become $\mathcal{M} = \mathcal{K}_I = \mathcal{K}_M = \mathbb{Z}_q$, $\mathcal{F} = \mathcal{K}' = \mathbb{G}_1$.
- $\mathsf{EXTRACT}$: The same as our first scheme.
- $\mathsf{FIX\text{-}INITIAL\text{-}KEYSTONE}$: Assuming a keystone $k_I \in \mathbb{G}_2$ is randomly selected, this algorithm outputs $f_I = H_2(k_I)$ as the keystone-fix.
- ASIGN: The input of this algorithm includes two identities ID_i and ID_j, a private key $\mathcal{S}_{\mathsf{ID}_i}$, a message m, a $\mathbb{G}_1$ element α, and a $\mathbb{G}_1$ element f.
  1. Compute $U_j = \alpha + f$ and $h_j = H_1(m||(\mathsf{ID}_i \oplus \mathsf{ID}_j)||U_j)$.
  2. Choose $r'_i \in_R \mathbb{Z}_q^*$, compute $U_i = r'_i \mathsf{Q}_{\mathsf{ID}_i} - U_j - h_j \mathsf{Q}_{\mathsf{ID}_j}$.
  3. Compute $h_i = H_1(m||(\mathsf{ID}_i \oplus \mathsf{ID}_j)||U_i)$ and $V = (h_i + r'_i)\mathcal{S}_{\mathsf{ID}_i}$.
  4. Output the signature $\sigma = \{U_i, U_j, V\}$.
- $\mathsf{ENC\text{-}MATCHING\text{-}KEYSTONE}$: The same as our first scheme.
- $\mathsf{FIX\text{-}SECRET\text{-}KEYSTONE}$: This algorithm returns $f_S = H_2(\hat{e}(K_M, \mathcal{S}_{\mathsf{ID}_j}))$.
- $\mathsf{AVERIFY}$: The input of this algorithm includes two identities ID_i and ID_j, a message m, and a ring signature $\sigma = \{U_i, U_j, V\}$.
  1. Compute $h_i = H_1(m||(\mathsf{ID}_i \oplus \mathsf{ID}_j)||U_i)$ and $h_j = H_1(m||(\mathsf{ID}_i \oplus \mathsf{ID}_j)||U_j)$.
  2. Return `accept` if $\hat{e}(P_{pub}, U_i + h_i \mathsf{Q}_{\mathsf{ID}_i} + U_j + h_j \mathsf{Q}_{\mathsf{ID}_j}) = \hat{e}(P, V)$, `reject` otherwise.
- $\mathsf{VERIFY\text{-}INITIAL\text{-}KEYSTONE}$: This algorithm outputs `accept` if $f_I = H_2(k_I)$, `reject` otherwise.

– VERIFY-SECRET-KEYSTONE: It outputs `accept` if $f_S = H_2(\hat{e}(P_{pub}, \mathsf{Q}_{\mathsf{ID}_j})^{k_M})$, `reject` otherwise.
– VERIFY-CONNECTION: This algorithm outputs `accept` if $U'_i = U_j + f_S$, `reject` otherwise.
– VERIFY. The algorithm accepts (k_I, m_I, S'), where $k_I \in \mathcal{K}_I$ and $k_M \in \mathcal{K}_M$ are the keystones and $S' = (m, \sigma_i, \sigma_j, \mathsf{ID}_i, \mathsf{ID}_j)$, where $\sigma = (U_i, U_j, V)$. The algorithm verifies whether (k_I, k_M) is valid and the connection between σ_i and σ_j is valid by using the above three algorithm. If it does not hold, then output `reject`. Otherwise, run AVERIFY(S). The output of VERIFY is the output of AVERIFY algorithm.

Correctness.
The correctness of our second scheme is justified as follows.

$$\begin{aligned}&\hat{e}(P_{pub}, U_i + h_i\mathsf{Q}_{\mathsf{ID}_i} + U_j + h_j\mathsf{Q}_{\mathsf{ID}_j})\\ &= \hat{e}(P_{pub}, r'_i\mathsf{Q}_{\mathsf{ID}_i} - U_j - h_j\mathsf{Q}_{\mathsf{ID}_j} + h_i\mathsf{Q}_{\mathsf{ID}_i} + U_j + h_j\mathsf{Q}_{\mathsf{ID}_j})\\ &= \hat{e}(sP, (h_i + r'_i)\mathsf{Q}_{\mathsf{ID}_i}) = \hat{e}(P, (h_i + r'_i)S_{\mathsf{ID}_i})\end{aligned} \quad \blacksquare$$

5.1 Security Consideration

Theorem 5. (Ambiguity) *Before the keystone k is released, both signatures are ambiguous.*

Lemma 2. *When the output of* VERIFY *is* `accept`*, then any third party can be sure who has generated the signature. Any party cannot* frame *that the other party has signed a message without his or her consent assuming the one-way property of the hash function. This guarantees that the signature is unforgeable.*

Theorem 6. (Unforgeability) *The scheme presented in this section is existentially unforgeable under a chosen message attack in the random oracle model, assuming the one-way property of the hash function, the hardness of the discrete logarithm problem and the Co-CDH assumption.*

Theorem 7. (Fairness) *For all signatures that are generated with the same keystone will be binding concurrently when the keystone is released.*

Theorem 8. *Our ID-based perfect concurrent signature scheme presented in this scheme is* secure *in the random oracle model, assuming the hardness of the discrete logarithm problem.*

5.2 Signature Length and Efficiency

In this scheme, each signature is a three-tuple (U_1, U_2, V), where $U_1, U_2, V \in \mathbb{G}_1$. With the same setting as our first scheme, our second scheme only requires 1,026 bits or 129 bytes for a pair of signatures. Hence, the signature is nearly as short as that of the first one. This signature length is only one group element on elliptic curve larger than most existing ID-based signature schemes (for example, see the review in [3]). Our second scheme inherits the efficiency of the underlying scheme by Chow *et al.* [8], such that no pairing operation is needed for signing, with a normal computational cost for other algorithms of the protocol.

6 Conclusion and Future Research Direction

We introduced the notion of ID-based perfect concurrent signatures, which is an extension of the notion of concurrent signatures proposed in [6]. We provided the first generic construction of (ID-based) perfect concurrent signature protocol in the literature. We presented two concrete constructions of ID-based perfect concurrent signature schemes based on our generic framework. Our second scheme requires no pairing operation in signing. We also provided a complete security analysis for our schemes on their ambiguity, fairness and unforgeability.

Recently, a new ID-based ring signature scheme was proposed [10]. Instead of following the existing paradigms of ID-based ring signature constructions, the scheme is constructed using a cryptographic primitive known as accumulator (e.g. see [10]). It would be interesting to see if concurrent signature could be realized from cryptographic accumulator.

References

1. N. Asokan, Victor Shoup, and Michael Waidner. Optimistic Fair Exchange of Digital Signatures. *IEEE Journal on Selected Areas in Communications*, 18, 2000.
2. Feng Bao, Robert H. Deng, and Wenbo Mao. Efficient and Practical Fair Exchange Protocols. In *IEEE Symposium on Security and Privacy 1998*, pp. 77–85, 1998.
3. Mihir Bellare, Chanathip Namprempre, and Gregory Neven. Security Proofs for Identity-Based Identification and Signature Schemes. In *Adv in Cryptology - Eurocrypt 2004, LNCS 3027*, pp. 268–286, 2004.
4. Dan Boneh and Matt Franklin. Identity-based Encryption from the Weil Pairing. In *Adv in Cryptology - Crypto 01, LNCS 2139*, pp. 213–229, 2001.
5. Dan Boneh, Ben Lynn, and Hovav Shacham. Short Signatures from the Weil Pairing. *Adv in Cryptology - Asiacrypt 2001, LNCS 2248*, pp. 514–532, 2001.
6. Liqun Chen, Caroline Kudla, and Kenneth G. Paterson. Concurrent Signatures. In *Adv in Cryptology - Eurocrypt 2004, LNCS 3027*, pp. 287–305, 2004.
7. Sherman S.M. Chow, Richard W.C. Lui, Lucas C.K. Hui, and Siu Ming Yiu. Identity Based Ring Signature: Why, How and What Next. *EuroPKI 2005, LNCS 3545*, pp. 144-161, 2005.
8. Sherman S.M. Chow, Siu Ming Yiu, and Lucas C.K. Hui. Efficient Identity Based Ring Signature. *Applied Crypto and Network Security - ACNS 2005, LNCS 3531*, pp. 499-512, 2005.
9. Yevgeniy Dodis and Leonid Reyzin. Breaking and Repairing Optimistic Fair Exchange from PODC 2003. *ACM Workshop on Digital Rights Management*, 2003.
10. Lan Nguyen. Accumulators from Bilinear Pairings and Applications. *Topics in Cryptology - CT-RSA 2005, LNCS 3376*, pp. 275–292, 2005.
11. David Pointcheval and Jacques Stern. Security Proofs for Signature Schemes. *Adv in Cryptology - Eurocrypt 1996, LNCS 1070*, pp. 387 – 398, 1996.
12. Ronald L. Rivest, Adi Shamir, and Yael Tauman: How to Leak a Secret. *Adv in Cryptology - Asiacrypt 2001, LNCS 2248*, pp. 552 – 565, 2001.
13. Willy Susilo, Yi Mu and Fangguo Zhang. Perfect Concurrent Signature Schemes. *Inf and Comm Security - ICICS 2004, LNCS 3269*, pp. 14–26, 2004.
14. Fangguo Zhang and Kwangjo Kim. ID-based Blind Signature and Ring Signature from Pairings. *Adv in Cryptology - Asiacrypt 2002, LNCS 2501*, pp. 533 – 547, 2002.