Formal Definition and Construction of Nominative Signature

Dennis Y.W. Liu[1], Duncan S. Wong[1], Xinyi Huang[2], Guilin Wang[3], Qiong Huang[1], Yi Mu[2], and Willy Susilo[2]

[1] Department of Computer Science
City University of Hong Kong
Hong Kong, China
{dliu,duncan,csqhuang}@cs.cityu.edu.hk*
[2] Centre for Computer and Information Security Research
School of Computer Science and Software Engineering
University of Wollongong
Wollongong, NSW 2522, Australia
{xh068,ymu,wsusilo}@uow.edu.au
[3] Institute for Infocomm Research, Singapore and
School of Computer Science University of Birmingham, UK
G.Wang@cs.bham.ac.uk

Abstract. Since the introduction of nominative signature in 1996, there are three problems that have still not been solved. First, there is no convincing application proposed; second, there is no formal security model available; and third, there is no proven secure scheme constructed, given that all the previous schemes have already been found flawed. In this paper, we give positive answers to these problems. First, we illustrate that nominative signature is a better tool for building user certification systems which were originally implemented using universal designated-verifier signature. Second, we propose a formal definition and adversarial model for nominative signature. Third, we show that Chaum's undeniable signature can be transformed to an efficient nominative signature by simply using a standard signature. The security of our transformation can be proven under the standard number-theoretic assumption.

1 Introduction

A nominative signature (NS) involves three parties: *nominator* A, *nominee* B and *verifier* C. The nominator A arbitrarily chooses a message m and works jointly with the nominee B to produce a signature σ called nominative signature. The validity of σ can only be verified by B and if σ is valid, B can convince the verifier C the validity of σ using a *confirmation protocol*; otherwise, B can convince C the invalidity of σ using a *disavowal protocol*. Based on the previous

* The authors were supported by CityU grants (Project nos. 7001844, 7001959, 7002001).

S. Qing, H. Imai, and G. Wang (Eds.): ICICS 2007, LNCS 4861, pp. 57–68, 2007.

literature [13,11,17,9], we consolidate the security requirements for a nominative signature as follows.

1. (*Joint Work of Nominator and Nominee*) A or B alone is not able to produce a valid σ;
2. (*Only Nominee Can Determine the Validity of Signature*) Only B can verify σ;
3. (*Can Only be Verified with Nominee's Consent*) The validity of σ is only verifiable with the aid of B, by running a confirmation/disavowal protocol with B;
4. (*Nominee Cannot Repudiate*) If σ is valid, B cannot mislead C to believe that σ is invalid using the disavowal protocol. If σ is invalid, B cannot mislead C to believe that σ is valid using the confirmation protocol;

Since the introduction of nominative signature (NS) [13], it has been considered as a dual scheme of undeniable signature (US) [4,2,5]. For an undeniable signature, its validity can only be verified with the aid of the signer, while for a nominative signature, its validity can only be verified with the aid of the nominee, rather than the nominator. Nominative signature is also related to designated verifier signature (DVS) [12], designated confirmer signature (DCS) [3] and universal designated-verifier signature (UDVS) [15]. We illustrate their similarities and differences below.

	Parties Involved	**Creator(s) of Signature**	**Playing the Role of Prover**		
			A	B	C
US	A, C	A	$\surd$	NA	$\times$
DCS	A, B, C	A	$\surd$	$\surd$	$\times$
DVS	A, C	A	$\surd$	NA	$\times$
UDVS	A, B, C	A and B[1]	$\surd$	$\surd$	$\times$
NS	A, B, C	A and B	$\times$	$\surd$	$\times$

Legend: A – Signer or Nominator (for NS); B – Confirmer (for DCS) or Signature Holder (for UDVS) or Nominee (for NS); C – Verifier or Designated Verifier (for DCS or UDVS); NA – not applicable.

As we can see, only NS does not allow the signer to prove the validity of a signature to a third party.

1.1 User Certification Systems

Since the introduction of NS in 1996 [13], there are only a few schemes [13,11] proposed. Unfortunately, all of them have already been found flawed [17,9]. Even worse, there is no convincing application described and NS still remains as of theoretical interest only. In the following, we show that NS is actually a much better tool for building *user certification systems* than UDVS [15] which was originally believed to be one of the most suitable ways of implementing this type of systems.

[1] A first creates a standard publicly verifiable signature and sends it securely to B; B then generates a UDVS signature based on the received standard signature.

UDVS, introduced by Steinfeld et al. [15] in 2003, allows a signature holder B to convince a designated verifier C that B holds a signer A's signature s on some message m, while C cannot further convince anybody of this fact. As illustrated in [15], UDVS is useful for constructing user certification systems, which concern about showing the validity of users' birth certificates, driving licences and academic transcripts, issued by an authority A. In such a system, a user B does not want a designated verifier C to disseminate B's certificate s (issued by A), while B needs to convince C that the certificate s is authentic, that is, signed by A.

NS can also be used for this purpose, but in a more natural way. For UDVS, A (the signer *or* the authority) should be trusted by B (the signature holder *or* the user of a certificate) in a very strong sense. If A is malicious, there are two attacks which will compromise B's interest on protecting his certificates. First, A may maliciously reveal the pair (s, m) to the public, and since s is a standard publicly verifiable signature, once s becomes public, everyone can verify its validity. B cannot show whether s is released by A because B himself can also make s public. Second, A can generate a UDVS signature all by himself because the UDVS signature can readily be generated from the public keys of A and C in addition to the pair (s, m). Hence, A can impersonate B arbitrarily. In contrast, NS does not have these weaknesses.

For NS, A cannot confirm or disavow a nominative signature σ (which is a user certificate in this type of applications) and σ is not publicly verifiable. Also, B does not have a publicly verifiable signature issued by A. Note that A can still issue standard signature on m or NS on m jointly with other nominees. But these events will just show that A is dishonest.

1.2 Related Work

The notion and construction of nominative signature (NS) were first proposed by Kim, Park and Won [13]. However, their construction was later found flawed [11] as the nominator in their construction can always determine the validity of a nominative signature, that is, violating Property 2 of NS described at the beginning of Sec. 1. In [11], Huang and Wang proposed the notion of convertible nominative signature, which allows the nominee to convert a nominative signature to a publicly verifiable one. They also proposed a new scheme. However, in [17,9], it was found that the nominator in their scheme can generate valid signatures on his own and show the validity of the signature to anyone without the consent of the nominee. That is, their scheme does not satisfy Properties 1 to 3.

In [11], a definition and some requirements for nominative signature were specified. However, their definition does not match with the scheme they proposed and the set of security requirements is incomplete and does not seem to be formal enough for provable security.

Our Results. We propose a formal definition and a rigorous set of adversarial models for nominative signature. We also propose a provably secure construction, which is based on Chaum's undeniable signature [2] and a strongly unforgeable signature scheme.

Paper Organization. The definition of nominative signature and its security models are specified in Sec. 2. The description and security analysis of our construction are given in Sec. 3. The paper is concluded in Sec. 4.

2 Definitions and Security Models

A nominative signature (NS) consists of three probabilistic polynomial-time (PPT) algorithms (SystemSetup, KeyGen, $\mathsf{Ver}^{\mathsf{nominee}}$) and three protocols (SigGen, Confirmation, Disavowal).

1. SystemSetup (System Setup): On input 1^k where $k \in \mathbb{N}$ is a security parameter, it generates a list of system parameters denoted by param.
2. KeyGen (User Key Generation): On input param, it generates a public/private key pair (pk, sk).
3. $\mathsf{Ver}^{\mathsf{nominee}}$ (Nominee-only Verification): On input a message m, a nominative signature σ, a public key pk_A and a private key sk_B, it returns valid or invalid.

An NS proceeds as follows. Given a security parameter $k \in \mathbb{N}$, SystemSetup is invoked and param is generated. KeyGen is then executed to initialize each party that is to be involved in the subsequent part of the scheme. One party called nominator is denoted by A. Let (pk_A, sk_A) be the public/private key pair of A. Let B be the nominee that A nominates, and (pk_B, sk_B) be B's public/private key pair. In the rest of the paper, we assume that entities can be uniquely identified from their public keys. To generate a nominative signature σ, A chooses a message $m \in \{0,1\}^*$, and carries out SigGen protocol with B. The protocol is defined as follows.

SigGen Protocol: Common inputs of A and B are param and m. A's additional input is pk_B, indicating that A nominates B as the nominee; and B's additional input is pk_A indicating that A is the nominator. At the end, either A or B outputs σ. The party who outputs σ should be explicitly indicated in the actual scheme specification.

The validity of a nominative signature σ on message m (with respect to pk_A and pk_B) can be determined by B as $\mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B)$. To convince a third party C on the validity or invalidity of (m, σ, pk_A, pk_B), B as a prover and C as a verifier carry out the Confirmation or Disavowal protocol as follows.

Confirmation/Disavowal Protocol: On input (m, σ, pk_A, pk_B), B sets μ to 1 if valid $\leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B)$; otherwise, μ is set to 0. B first sends μ to C. If $\mu = 1$, Confirmation protocol is carried out; otherwise, Disavowal protocol is carried out. At the end of the protocol, C outputs either accept or reject while B has no output.

Correctness. Suppose that all the algorithms and protocols of a nominative signature scheme are carried out accordingly by honest entities A, B and C, the scheme is said to satisfy the correctness requirement if

1. valid $\leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B)$; and
2. C outputs accept at the end of the Confirmation protocol.

Validity of a Nominative Signature. A nominative signature σ on message m with respect to nominator A and nominee B is *valid* if $\mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B) = \mathsf{valid}$. In this case, we say that quadruple (m, σ, pk_A, pk_B) is *valid*. Note that only B can determine the validity of σ.

In the following, we propose and formalize a set of security notions for nominative signature. They are (1) unforgeability, (2) invisibility, (3) security against impersonation, and (4) non-repudiation.

2.1 Unforgeability

Intuitively, an adversary should not able to forge a valid nominative signature if any of the private keys of A and B is not known. Our game below is based on the notion of existential unforgeability against chosen message attack [8] with the extension of allowing access to confirmation/disavowal oracle based on passive attack or active/concurrent attack introduced by Kurosawa and Heng [14] in the undeniable signature setting.

We also allow the adversary to access an oracle called SignTranscript which simulates various interactions between the adversary and other honest entities. In addition, the adversary may collude with other parties or claim that some particular party is his nominee without the party's consent. Hence we also allow the adversary to adaptively access CreateUser oracle and Corrupt oracle as defined below.

Game Unforgeability: Let $\mathcal{S}$ be the simulator and $\mathcal{F}$ be a forger.

1. (*Initialization*) Let $k \in \mathbb{N}$ be a security parameter. First, $\mathsf{param} \leftarrow \mathsf{SystemSetup}(1^k)$ is executed and key pairs (pk_A, sk_A) and (pk_B, sk_B) for nominator A and nominee B, respectively, are generated using KeyGen. Then $\mathcal{F}$ is invoked with inputs 1^k, pk_A and pk_B.
2. (*Attacking Phase*) $\mathcal{F}$ can make queries to the following oracles:
   - CreateUser: On input an identity, say I, it generates a key pair (pk_I, sk_I) using KeyGen and returns pk_I.
   - Corrupt: On input a public key pk, if pk is generated by CreateUser or in $\{pk_A, pk_B\}$, the corresponding private key is returned; otherwise, $\perp$ is returned. pk is said to be *corrupted*.
   - SignTranscript: On input a message m, two distinct public keys, pk_1 (the nominator) and pk_2 (the nominee) such that at least one of them is uncorrupted, and one parameter called $role \in \{\mathsf{nil}, \mathsf{nominator}, \mathsf{nominee}\}$,
     - if $role = \mathsf{nil}$, $\mathcal{S}$ simulates a run of SigGen and returns a valid quadruple (m, σ, pk_1, pk_2) and $trans_\sigma$ which is the transcript of the execution of SigGen;
     - if $role = \mathsf{nominator}$, $\mathcal{S}$ (as nominee with public key pk_2) simulates a run of SigGen with $\mathcal{F}$ (as nominator with pk_1);
     - if $role = \mathsf{nominee}$, $\mathcal{S}$ (as nominator with pk_1) simulates a run of SigGen with $\mathcal{F}$ (as nominee with public key pk_2).

- Confirmation/disavowal: On input a message m, a nominative signature σ and two public keys pk_1 (the nominator), pk_2 (the nominee), let sk_2 be the corresponding private key of pk_2, the oracle responds based on whether a passive attack or an active/concurrent attack is mounted.
  - In a passive attack, if $\mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_1, sk_2) = \mathsf{valid}$, the oracle returns a bit $\mu = 1$ and a transcript of the Confirmation protocol. Otherwise, $\mu = 0$ and a transcript of the Disavowal protocol are returned.
  - In an active/concurrent attack, if $\mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_1, sk_2) = \mathsf{valid}$, the oracle returns $\mu = 1$ and then proceeds to execute the Confirmation protocol with $\mathcal{F}$ (acting as a verifier). Otherwise, the oracle returns $\mu = 0$ and executes the Disavowal protocol with $\mathcal{F}$. The difference between active and concurrent attack is that $\mathcal{F}$ interacts serially with the oracle in the active attack while $\mathcal{F}$ interacts with different instances of the oracle concurrently in the concurrent attack.

3. (*Output Phase*) $\mathcal{F}$ outputs a pair (m^*, σ^*) as a forgery of A's nominative signature on message m^* with B as the nominee.

The forger $\mathcal{F}$ *wins* the game if $\mathsf{Ver}^{\mathsf{nominee}}(m^*, \sigma^*, pk_A, sk_B) = \mathsf{valid}$ and (1) $\mathcal{F}$ does not corrupt both sk_A and sk_B using oracle Corrupt; (2) $(m^*, pk_A, pk_B, role)$ has never been queried to SignTranscript for any valid value of $role$; (3) $(m^*, \sigma', pk_A, pk_B)$ has never been queried to Confirmation/disavowal for any nominative signature σ' with respect to pk_A and pk_B. $\mathcal{F}$'s advantage is defined to be the probability that $\mathcal{F}$ wins.

Definition 1. *A nominative signature scheme is said to be unforgeable if no PPT forger $\mathcal{F}$ has a non-negligible advantage in Game Unforgeability.*

Note that he second restriction above does not disallow $\mathcal{F}$ to query SignTranscript with $(m^*, pk_A, pk', role)$ provided that any $pk' \neq pk_B$.

2.2 Invisibility

We now formalize the requirement that only nominee B can determine whether a nominative signature is valid. We adopt the formalization idea given by Galbraith and Mao [7]. The formalization is indistinguishability based and is defined to distinguish between a valid signature σ on message m or just some value chosen uniformly at random from the corresponding signature space. Note that if the scheme is unforgeable in the sense of Def. 1, then it is negligible that a uniformly chosen value from the signature space is a valid signature on m.

Game Invisibility: The initialization phase is the same as that of Game Unforgeability and the distinguisher $\mathcal{D}$ is permitted to issue queries to all the oracles described in the attacking phase of Game Unforgeability.

1. At some point in the attacking phase, $\mathcal{D}$ outputs a message m^* and requests a challenge nominative signature σ^* on m^*. The challenge σ^* is generated based on the outcome of a hidden coin toss b.

- If $b = 1$, σ^* is generated by running SigGen.
- If $b = 0$, σ^* is chosen randomly from the signature space of the nominative signature scheme with respect to pk_A and pk_B.

2. At the end of the game, $\mathcal{D}$ outputs a guess b'.

$\mathcal{D}$ *wins* the game if $b' = b$ and (1) $\mathcal{D}$ does not corrupt sk_B; (2) the quadruple $(m^*, pk_A, pk_B, role)$, for any valid value of $role$, has never been queried to SignTranscript; (3) $(m^*, \sigma^*, pk_A, pk_B)$ has never been queried to Confirmation/disavowal.

$\mathcal{D}$'s advantage in this game is defined as $|\Pr[b' = b] - \frac{1}{2}|$.

Definition 2. *A nominative signature scheme is said to have the property of invisibility if no PPT distinguisher $\mathcal{D}$ has a non-negligible advantage in Game Invisibility.*

2.3 Security Against Impersonation

The notion of impersonation was first proposed by Kurosawa and Heng [14] in the context of undeniable signature. Instead of achieving zero-knowledgeness, it is noticed that the actual security requirement is to prevent the proving capability of the validity of a signature from being given away to any illegitimate party. This requirement is also commonly referred to as non-transferability. We consider the following game against an impersonator $\mathcal{I}$.

Game Impersonation: The initialization phase is the same as that of Game Unforgeability. The game has two phases as follows.

- (*Preparation Phase*) Impersonator $\mathcal{I}$ is invoked on input 1^k, pk_A, pk_B, sk_A. In this phase, $\mathcal{I}$ may query any of the oracles defined in Game Unforgeability. $\mathcal{I}$ prepares a triple (m^*, σ^*, μ) where m^* is some message, σ^* is a nominative signature (i.e. σ^* is in the signature space with respect to pk_A and pk_B) and μ is a bit.
- (*Impersonation Phase*) If $\mu = 1$, $\mathcal{I}$ (as nominee) executes Confirmation protocol with the simulator (as a verifier) on common inputs $(m^*, \sigma^*, pk_A, pk_B)$. If $\mu = 0$, $\mathcal{I}$ executes Disavowal protocol with the same set of inputs.

$\mathcal{I}$ *wins* if the simulator outputs accept at the Impersonation Phase while $\mathcal{I}$ has never corrupted sk_B in the game. $\mathcal{I}$'s advantage is defined to be the probability that $\mathcal{I}$ wins.

Definition 3. *A nominative signature scheme is said to be secure against impersonation if no PPT impersonator $\mathcal{I}$ has a non-negligible advantage in Game Impersonation.*

2.4 Non-repudiation

Due to the property of invisibility, no one except the nominee can determine the validity of a signature. In addition, even the nominator A and the nominee

B jointly generate a valid quadruple (m, σ, pk_A, pk_B), this only indicates that $\mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B)$ outputs valid. It does not imply that nominee B cannot cheat by executing Disavowal protocol successfully on (m, σ, pk_A, pk_B) with a verifier. Therefore, for ensuring that B cannot repudiate, we require this security notion. We consider the game below against a cheating nominee $\mathcal{B}$.

Game Non-repudiation: The initialization phase is the same as that of Game Unforgeability and the cheating nominee $\mathcal{B}$ can query any of the oracles defined in Game Unforgeability. sk_B is also given to $\mathcal{B}$.

- (*Preparation Phase*) $\mathcal{B}$ prepares (m^*, σ^*, μ) where m^* is some message and σ^* is a nominative signature. $\mu = 1$ if $\mathsf{Ver}^{\mathsf{nominee}}(m^*, \sigma^*, pk_A, sk_B) = \mathsf{valid}$; otherwise, $\mu = 0$.
- (*Repudiation Phase*) If $\mu = 1$, $\mathcal{B}$ executes Disavowal protocol with the simulator (acting as a verifier) on $(m^*, \sigma^*, pk_A, pk_B)$ but the first bit sent to the simulator is 0. If $\mu = 0$, $\mathcal{B}$ executes Confirmation protocol but the first bit sent to the simulator is 1.

$\mathcal{B}$ *wins* the game if the simulator acting as the verifier outputs accept. $\mathcal{B}$'s advantage is defined to be the probability that $\mathcal{B}$ wins.

Definition 4. *A nominative signature scheme is said to be secure against repudiation by nominee if no PPT cheating nominee $\mathcal{B}$ has a non-negligible advantage in Game Non-repudiation.*

3 Our Construction

In this section, we propose an efficient and provably secure construction of nominative signature. Our construction is based on Chaum's undeniable signature [2,14] and a strongly unforgeable signature scheme [1,16,10]. One desirable property of our construction is that one may generalize it to a generic scheme or instantiate it with some other undeniable signature schemes. We leave this as our further investigation. In the following, let σ^{undeni} be Chaum's undeniable signature and $\sigma^{standard}$ a strongly unforgeable standard signature. Also let $k \in \mathbb{N}$ be a system parameter.

SystemSetup: The algorithm generates a cyclic group G of prime order $q \geq 2^k$, a generator g, and a hash function $H : \{0,1\}^* \rightarrow G$. Let $\mathsf{param} = (k, G, q, g, H)$. We say that (g, g^u, g^v, g^w) is a DH-tuple [14] if $w = uv \bmod q$; otherwise, it is a non-DH-tuple.

KeyGen: On input param, (pk, sk) is generated where $sk = (x, Sig)$ for some random $x \in_R \mathbb{Z}_q$ and standard signature generation algorithm Sig, and $pk = (y, Ver)$ for $y = g^x$ and standard signature verification algorithm Ver. We use $pk_A = (y_A, Ver_A)$ and $sk_A = (x_A, Sig_A)$ to denote nominator A's public and private key, respectively. Similarly, let (pk_B, sk_B) be nominee B's public/private key pair.

SigGen Protocol: Let $m \in \{0,1\}^*$ be a message. On input param and m, and specific input pk_B for A and pk_A for B, the protocol is carried out as follows.

1. B sends $\sigma^{undeni} = H(m\|pk_A)^{x_B}$ to A.
2. B then proves to A that $(g, y_B, H(m\|pk_A), \sigma^{undeni})$ is a DH-tuple using a Witness Indistinguishable (WI) protocol [6,14][2].
3. If A accepts, A outputs $\sigma = (\sigma^{undeni}, \sigma^{standard})$ where $\sigma^{standard} = Sig_A(\sigma^{undeni})$ which is A's standard signature on σ^{undeni}.

We say that $\sigma = (\sigma_1, \sigma_2)$ is a nominative signature (i.e. σ is in the signature space with respect to pk_A and pk_B) if $\sigma_1 \in G$ and σ_2 is in the set of A's signature on "message" σ_1, that is, $Ver_A(\sigma_1, \sigma_2) = 1$ meaning that σ_2 is a valid standard signature of "message" σ_1.

$\mathsf{Ver}^{\mathsf{nominee}}$: On input (m, σ, pk_A, sk_B), where $\sigma = (\sigma^{undeni}, \sigma^{standard})$ is a nominative signature (i.e. σ is in the signature space defined as above), if $\sigma^{undeni} = H(m\|pk_A)^{x_B}$, output valid; otherwise, output invalid.

Confirmation/Disavowal Protocol: On input (m, σ, pk_A, pk_B) where σ is a nominative signature, if $\mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B) = \mathsf{valid}$, B sends $\mu = 1$ to C; otherwise, $\mu = 0$ is sent to C. B then proves/disproves to C the DH-tuple/non-DH-tuple $(g, y_B, H(m\|pk_A), \sigma^{undeni})$ using WI protocols [6,14].

3.1 Discussions

Although each party's public or private key has two components, for nominator, only the component of standard signature (i.e. Sig_A, Ver_A) is used; while for nominee, only the component of undeniable signature (i.e. x_B, y_B) is used. In practice, the nominee of one message can be the nominator of another message. So we make the description above general enough for this practical scenario. Also, and more important, it abides by the definition (Sec. 2). In some settings, the two components of each key can be combined. For example, if both A and B are using discrete-log based keys for generating standard signatures, then one private key x is enough for each of them. Namely, each user can use the same private key for generating both standard signatures (e.g. Schnorr's signature scheme) and Chaum's undeniable signatures.

The standard signature $\sigma^{standard}$ generated by A only authenticates the "message" σ^{undeni} rather than the actual message m. There is still no proof on whether $(\sigma^{undeni}, \sigma^{standard})$ corresponds to m. Someone can replace m with another message, say m', and claim that $(\sigma^{undeni}, \sigma^{standard})$ corresponds to m'. No one can prove this claim, only nominee can.

Different from Chaum's original scheme [2] (precisely, we use the hash variant of Chaum's scheme [14]), the undeniable signature σ^{undeni} is computed as $H(m\|pk_A)^{x_B}$ rather than $H(m)^{x_B}$ as in the original scheme. It is important to

[2] First observed by Kurosawa and Heng [14], Chaum's undeniable signature (i.e. σ^{undeni}) can be confirmed/disavowed if the prover knows one of the two witnesses, that is, x_B or discrete logarithm of $H(m\|pk_A)$. This allows us to use the WI protocol.

include A's public key. Otherwise, the scheme will be insecure against unforgeability (Sec. 2.1) and invisibility (Sec. 2.2) due to the capture of multi-party environment in our security models. For example, under the model of unforgeability (Sec. 2.1), suppose pk_A is not included, forger $\mathcal{F}$ in the model can corrupt A's private key sk_A, then query SignTranscript on $(m, pk_I, pk_B, \mathsf{nil})$ where pk_I is some public key returned by CreateUser. As defined, the game simulator will return a valid quadruple (m, σ, pk_I, pk_B) where pk_B indicates the nominee. Note that $\sigma = (H(m)^{x_B}, Sig_I(H(m)^{x_B}))$. Finally, $\mathcal{F}$ outputs $(m^*, \sigma^* = (\sigma^{undeni*}, \sigma^{standard*}), pk_A, pk_B)$ where $m^* = m$, $\sigma^{undeni*} = H(m)^{x_B}$ and $\sigma^{standard*} = Sign_A(H(m)^{x_B})$. This attack shows that a malicious party A can set a party B up and claim that B is A's nominee even B is not.

3.2 Security Analysis

We now analyze the security of the construction proposed above with respect to the security notions formalized in Sec. 2.

Lemma 1. *Let $k \in \mathbb{N}$ be a security parameter. For the nominative signature scheme proposed above, suppose a (t, ϵ, Q)-forger has obtained the nominee B's private key sk_B and is able to forge a valid nominative signature with probability at least ϵ, there exists a (t', ϵ')-adversary which can existentially forge a standard signature under the model of chosen message attack [8] with probability at least $\epsilon' = (1 - 2^{-k}Q)\epsilon$ after running at most time $t' = t + Qt_q + c$ where t_q is the maximum time for simulating one oracle query and c is some constant.*

Lemma 2. *Let $k \in \mathbb{N}$ be a security parameter. For the nominative signature scheme proposed above, suppose a (t, ϵ, Q)-forger has obtained the nominator A's private key sk_A and is able to forge a valid nominative signature, there exists a (t', ϵ')-adversary which can solve a CDH (Computational Diffie-Hellman) problem instance with probability at least $\epsilon' = (1 - 2^{-k})(1 - 2^{-k}Q)Q^{-1}\epsilon$ after running at most time $t' = t + Qt_q + c$ where t_q is the maximum time for simulating one oracle query and c is some constant.*

Theorem 1 (Unforgeability). *The nominative signature scheme proposed above is unforgeable (Def. 1) if there exists a standard signature scheme which is existentially unforgeable against chosen message attack [8] and CDH problem in G is hard.*

The theorem follows directly from Lemma 1 and 2.

Theorem 2 (Invisibility). *The nominative signature scheme proposed above has the property of invisibility (Def. 2) under the Decisional Diffie-Hellman (DDH) assumption, if the underlying standard signature scheme is strongly existentially unforgeable against chosen message attack (strong euf-cma [1,16,10]).*

Due to page limitation, we leave all the security proofs in the full version of this paper. We remark that our proof requires a stronger sense of secure signature

scheme (namely, strong euf-cma secure) for invisibility, rather than a conventional euf-cma secure signature scheme as required for achieving unforgeability. It prevents the distinguisher in Game Invisibility from querying the Confirmation/disavowal oracle on an existentially forged value of the challenge signature σ^*. In practice, strong euf-cma secure signature schemes can be constructed efficiently. We refer readers to [1,16,10] for examples of efficient generic constructions of strong euf-cma secure signature schemes. Other methods in place of a strong euf-cma secure signature scheme may be feasible. For example, we may define an equivalence class of all valid signatures of σ^* and restrict the Confirmation/disavowal oracle from responding to any of the values in the class. We leave this as our further investigation.

Theorem 3 (Security Against Impersonation). *The nominative signature scheme proposed above is secure against impersonation (Def. 3) under the discrete logarithm (DLOG) assumption.*

Both confirmation and disavowal protocols use the WI protocols of [14], that have been proven to satisfy the requirement of security against impersonation in a similar model (Theorem 3 of [14]).

Theorem 4 (Non-repudiation). *The nominative signature scheme proposed above is secure against repudiation by nominee (Def. 4).*

This follows directly the soundness property of the WI proofs in [14].

4 Conclusion

In this paper, we proposed a rigorous set of security models for capturing the security notions of nominative signature. We also proposed a provably secure construction which efficiently converts Chaum's undeniable signature to a nominative signature using a strongly unforgeable signature scheme. We hope that with this formal security model, more provably secure nominative signature schemes can be proposed in the near future. We also believe that the security model is of independent interest and further enhancement of the security model is feasible. We consider this to be our future work.

References

1. Boneh, D., Shen, E., Waters, B.: Strongly unforgeable signatures based on computational Diffie-Hellman. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 229–240. Springer, Heidelberg (2006)
2. Chaum, D.: Zero-knowledge undeniable signatures. In: Damgård, I.B. (ed.) EUROCRYPT 1990. LNCS, vol. 473, pp. 458–464. Springer, Heidelberg (1991)
3. Chaum, D.: Designated confirmer signatures. In: De Santis, A. (ed.) EUROCRYPT 1994. LNCS, vol. 950, pp. 86–91. Springer, Heidelberg (1995)
4. Chaum, D., van Antwerpen, H.: Undeniable signatures. In: Brassard, G. (ed.) CRYPTO 1989. LNCS, vol. 435, pp. 212–216. Springer, Heidelberg (1990)

5. Chaum, D., van Antwerpen, H.: Cryptographically strong undeniable signatures, unconditionally secure for the signer. In: Feigenbaum, J. (ed.) CRYPTO 1991. LNCS, vol. 576, pp. 470–484. Springer, Heidelberg (1992)
6. Feige, U., Shamir, A.: Witness indistinguishable and witness hiding protocols. In: Proc. 22nd ACM Symp. on Theory of Computing, pp. 416–426. ACM Press, New York (May 1990)
7. Galbraith, S., Mao, W.: Invisibility and anonymity of undeniable and confirmer signatures. In: Joye, M. (ed.) CT-RSA 2003. LNCS, vol. 2612, pp. 80–97. Springer, Heidelberg (2003)
8. Goldwasser, S., Micali, S., Rivest, R.: A digital signature scheme secure against adaptive chosen-message attack. SIAM J. Computing 17(2), 281–308 (1988)
9. Guo, L., Wang, G., Wong, D.: Further discussions on the security of a nominative signature scheme. Cryptology ePrint Archive, Report 2006/007 (2006)
10. Huang, Q., Wong, D.S., Zhao, Y.: Generic transformation to strongly unforgeable signatures. In: ACNS 2007, pp. 1–17. Springer, Heidelberg (2007), also available at: `http://eprint.iacr.org/2006/346`
11. Huang, Z., Wang, Y.: Convertible nominative signatures. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 348–357. Springer, Heidelberg (2004)
12. Jakobsson, M., Sako, K., Impagliazzo, R.: Designated verifier proofs and their applications. In: Maurer, U.M. (ed.) EUROCRYPT 1996. LNCS, vol. 1070, pp. 143–154. Springer, Heidelberg (1996)
13. Kim, S.J., Park, S.J., Won, D.H.: Zero-knowledge nominative signatures. In: PragoCrypt 1996, International Conference on the Theory and Applications of Cryptology, pp. 380–392 (1996)
14. Kurosawa, K., Heng, S.: 3-move undeniable signature scheme. In: Cramer, R.J.F. (ed.) EUROCRYPT 2005. LNCS, vol. 3494, pp. 181–197. Springer, Heidelberg (2005)
15. Steinfeld, R., Bull, L., Wang, H., Pieprzyk, J.: Universal designated-verifier signatures. In: Laih, C.-S. (ed.) ASIACRYPT 2003. LNCS, vol. 2894, pp. 523–542. Springer, Heidelberg (2003)
16. Steinfeld, R., Pieprzyk, J., Wang, H.: How to strengthen any weakly unforgeable signature into a strongly unforgeable signature. In: Abe, M. (ed.) CT-RSA 2007. LNCS, vol. 4377, pp. 357–371. Springer, Heidelberg (2006)
17. Susilo, W., Mu, Y.: On the security of nominative signatures. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 329–335. Springer, Heidelberg (2005)