Nominative Signature: Application, Security Model and Construction

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 Information Security Research
School of Information Technology and Computer Science
University of Wollongong
Wollongong 2522, Australia
{xh068,ymu,wsusilo}@uow.edu.au

[3] Infocommm Security Department
Institute for Infocomm Research (I^{2}R)
Singapore
glwang@i2r.a-star.edu.sg

Abstract. Since the introduction of nominative signature in 1996, there have been only a few schemes proposed and all of them have already been found flawed. In addition, there is no formal security model defined. Even more problematic, there is no convincing application proposed. Due to these problems, the research of nominative signature has almost stalled and it is unknown if a secure nominative signature scheme can be built or there exists an application for it. 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 are originally believed to be best implemented using a universal designated-verifier signature. Second, we propose a formal definition and a rigorous set of adversarial models for nominative signature. Third, we show that Chaum's undeniable signature can be transformed efficiently to a nominative signature and prove its security.

Keywords: Digital Signature, Nominative Signature, Undeniable Signature

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*. Below are the properties of a nominative signature [14,12,18,10].

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;
5. (*Nominator Chooses Message*) Message m is chosen by A;
6. (*Only Nominator Can Nominate*) B is chosen/nominated by A.

Since the introduction of nominative signature (NS) [14], it has been considered as a dual scheme of undeniable signature (US) [5,3,6]. 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 (albeit it is the nominator who chooses the message). Nominative signature is also related to designated verifier signature (DVS) [13], designated confirmer signature (DCS) [4] and universal designated-verifier signature (UDVS) [16]. 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	√	NA	×
DCS	A, B, C	A	√	√	×
DVS	A, C	A	√	NA	×
UDVS	A, B, C	A and B[4]	√	√	×
NS	A, B, C	A and B	×	√	×

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 has the ability of proving the validity of a signature been dethroned from the nominator (or signer who chooses the message). None of the other signature types has this property.

1.1 User Certification Systems

Since the introduction of NS in 1996 [14], there have been only a few schemes [14,12] proposed and all of them have already been found flawed [18,10]. Even worse, there is no convincing application ever proposed and NS still remains as of theoretical interest only. In the following, we show that NS is actually a much better tool for

[4] 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.

building *user certification systems* than UDVS [16] which is originally believed to be one of the most suitable ways of implementing this type of systems.

UDVS, introduced by Steinfeld et al. [16] 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 [16], 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 [14]. However, their construction was later found flawed [12] 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 [12], 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 [18,10], 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 [12], 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 [3] 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 algorithms ($\mathsf{SystemSetup}$, KeyGen, $\mathsf{Ver}^{\mathsf{nominee}}$) and three protocols ($\mathsf{SigGen}$, $\mathsf{Confirmation}$, $\mathsf{Disavowal}$).

1. $\mathsf{SystemSetup}$ (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 $\mathsf{invalid}$.

An NS proceeds as follows. Given a security parameter $k \in \mathbb{N}$, $\mathsf{SystemSetup}$ 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.

Signature Space: A value σ is a nominative signature with respect to pk_A and pk_B if it is in the *signature space* of the NS with respect to pk_A and pk_B. We emphasize that the signature space has to be specified explicitly in each actual NS scheme.

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 $\mathsf{Confirmation}$ or $\mathsf{Disavowal}$ protocol as follows.

Confirmation/Disavowal Protocol: On input (m, σ, pk_A, pk_B), B sets μ to 1 if $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B)$; 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. $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B)$; 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 (Property 2).

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

According to Property 1, an adversary should not able to forge a valid message-signature pair if the adversary does not know the private keys of both A and B. A straightforward approach is to apply the notion of existential unforgeability against chosen message attack [9] using signing oracle with the extension of allowing access to confirmation/disavowal oracle based on passive attack or active/concurrent attack introduced by Kurosawa and Heng [15] in the context of undeniable signature to a game for nominative signature. However, a nominative signature scheme has two additional properties which are related to unforgeability but cannot be captured in this way. These are Properties 5 and 6 described in Sec. 1. To capture these properties, the adversary is also allowed 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:
   – $\mathsf{CreateUser}$: On input an identity, say I, it generates a key pair (pk_I, sk_I) using KeyGen and returns pk_I.
   – $\mathsf{Corrupt}$: On input a public key pk, if pk is generated by $\mathsf{CreateUser}$ or in $\{pk_A, pk_B\}$, the corresponding private key is returned; otherwise, $\perp$ is returned. pk is said to be *corrupted*.
   – $\mathsf{SignTranscript}$: 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).
   – $\mathsf{Confirmation/disavowal}$: 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, the oracle runs $\mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_1, sk_2)$. If the output is valid (that is, quadruple (m, σ, pk_1, pk_2) is valid), the oracle returns a bit $\mu = 1$ and a transcript of the $\mathsf{Confirmation}$ protocol. Otherwise, $\mu = 0$ and a transcript of the $\mathsf{Disavowal}$ protocol are returned.
     - In an active/concurrent attack, the oracle checks if quadruple (m, σ, pk_1, pk_2) is valid. If so, the oracle returns $\mu = 1$ and then proceeds to execute the $\mathsf{Confirmation}$ protocol with $\mathcal{F}$ (acting as a verifier). Otherwise, the oracle returns $\mu = 0$ and executes the $\mathsf{Disavowal}$ 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 quadruple $(m^*, \sigma^*, pk_A, pk_B)$ is valid and (1) $\mathcal{F}$ does not corrupt both sk_A and sk_B using oracle $\mathsf{Corrupt}$; (2) $(m^*, pk_A, pk_B, role)$ has never been queried to $\mathsf{SignTranscript}$ for any valid value of $role$; (3) $(m^*, \sigma', pk_A, pk_B)$ has never been queried to $\mathsf{Confirmation/disavowal}$ for any nominative signature σ' with respect to pk_A and pk_B (check *Signature Space* on page 4).

The forgery σ^* on m^* is valid if $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m^*, \sigma^*, pk_A, sk_B)$. $\mathcal{F}$'s advantage in this game 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.*

The second restriction above does not disallow $\mathcal{F}$ to query SignTranscript with $(m^*, pk_A, pk', role)$ provided that $pk' \neq pk_B$. This captures Property 6. Since $\mathcal{F}$ can also query SignTranscript with $(m', pk_A, pk_B, role)$ for any $m' \neq m^*$ with sk_B corrupted, Property 5 is also captured.

2.2 Invisibility

This notion corresponds to Property 2, which requires that only nominee B can determine whether a given quadruple (m, σ, pk_A, pk_B) is valid. This property also excludes the nominator A from determining the validity of a given quadruple. We adopt the formalization idea given by Galbraith and Mao [8]. 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.

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 [15] 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. In the context of nominative signature, security against impersonation refers to Property 3 in the Introduction section. 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 which corresponds to Property 4. 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 [3,15] and a strongly unforgeable (standard) signature scheme [1,2,17]. 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 an 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 $H : \{0,1\}^* \to G$. Let $\mathsf{param} = (k, G, q, g, H)$. We say that (g, g^u, g^v, g^w) is a DH-tuple [15] 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 [7,15][5].
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.

[5] First observed by Kurosawa and Heng [15], 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.

$\mathsf{Ver}^{\mathsf{nominee}}$: 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 [7,15].

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 [3] (precisely, we use the hash variant of Chaum's scheme [15]), 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 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 sets a party B up and claims 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, if a (t, ϵ, Q)-nominee can 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 [9] 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, if a (t, ϵ, Q)-nominator can 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 [9] 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 [1,2,17].*

All proofs are given in Appendix A. We require a stronger sense of signature scheme (namely, strong euf-cma secure) for invisibility, rather than a conventional signature scheme as required for achieving unforgeability. As shown in the proof (Appendix A), it prevents the distinguisher 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 [2,17,11] 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 calls 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 [15], that have been proven to satisfy the requirement of security against impersonation in a similar model (Theorem 3 of [15]).

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 [15].

4 Concluding Remarks

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. As a final remark, we believe that the security model is of independent interest and further enhancement of the security model is feasible. We consider this to be our further work.

References

1. J. An, Y. Dodis, and T. Rabin. On the security of joint signature and encryption. In *Proc. EUROCRYPT 2002*, pages 83–107. Springer-Verlag, 2002. LNCS 2332.
2. D. Boneh, E. Shen, and B. Waters. Strongly unforgeable signatures based on computational Diffie-Hellman. In *Proc. of PKC 2006*, pages 229–240. Springer-Verlag, 2006. LNCS 3958.
3. D. Chaum. Zero-knowledge undeniable signatures. In *Proc. EUROCRYPT 90*, pages 458–464. Springer-Verlag, 1990. LNCS 473.
4. D. Chaum. Designated confirmer signatures. In *Proc. EUROCRYPT 94*, pages 86–91. Springer-Verlag, 1994. LNCS 950.
5. D. Chaum and H. van Antwerpen. Undeniable signatures. In *Proc. CRYPTO 89*, pages 212–216. Springer-Verlag, 1990. LNCS 435.
6. D. Chaum and H. van Antwerpen. Cryptographically strong undeniable signatures, unconditionally secure for the signer. In *Proc. CRYPTO 91*, pages 470–484. Springer-Verlag, 1992. LNCS 576.
7. U. Feige and A. Shamir. Witness indistinguishable and witness hiding protocols. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 416–426, May 1990.
8. S. Galbraith and W. Mao. Invisibility and anonymity of undeniable and confirmer signatures. In *Topics in Cryptology – CT-RSA 2003*, pages 80–97. Springer-Verlag, 2003. LNCS 2612.
9. S. Goldwasser, S. Micali, and R. Rivest. A digital signature scheme secure against adaptive chosen-message attack. *SIAM J. Computing*, 17(2):281–308, Apr. 1988.
10. L. Guo, G. Wang, and D. Wong. Further discussions on the security of a nominative signature scheme. Cryptology ePrint Archive, Report 2006/007, 2006.
11. Q. Huang, D. S. Wong, and Y. Zhao. Generic transformation to strongly unforgeable signatures. Cryptology ePrint Archive, Report 2006/346 (Revised Date: 29 Nov 2006), 2006. `http://eprint.iacr.org/2006/346`.
12. Z. Huang and Y. Wang. Convertible nominative signatures. In *Proc. of Information Security and Privacy (ACISP'04)*, pages 348–357. Springer-Verlag, 2004. LNCS 3108.
13. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Proc. EUROCRYPT 96*, pages 143–154. Springer, 1996. LNCS 1070.

14. S. J. Kim, S. J. Park, and D. H. Won. Zero-knowledge nominative signatures. In *PragoCrypt'96, International Conference on the Theory and Applications of Cryptology*, pages 380–392, 1996.
15. K. Kurosawa and S. Heng. 3-move undeniable signature scheme. In *Proc. EUROCRYPT 2005*, pages 181–197, 2005. LNCS 3494.
16. R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk. Universal designated-verifier signatures. In *Proc. ASIACRYPT 2003*, pages 523–542. Springer, 2003. LNCS 2894.
17. R. Steinfeld, J. Pieprzyk, and H. Wang. How to strengthen any weakly unforgeable signature into a strongly unforgeable signature. To appear in CT-RSA 2007.
18. W. Susilo and Y. Mu. On the security of nominative signatures. In *Proc. of Information Security and Privacy (ACISP'05)*, pages 329–335. Springer-Verlag, 2005. LNCS 3547.

A Security Proofs

A.1 Proof of Lemma 1

Proof. Suppose a (t, ϵ, Q)-forger $\mathcal{F}$ has obtained the nominee B's private key $sk_B = (x_B, Sig_B)$ and is able to win Game Unforgeability by producing a valid nominative signature $\sigma^* = (\sigma_1^*, \sigma_2^*)$ on some message m^*. We show that in the random oracle model, $\mathcal{F}$ can be turned into a (t', ϵ')-algorithm $\mathcal{S}$ which existentially forges a message-signature pair against a signature scheme (Sig^*, Ver^*) under the model of [9].

Game Simulation: At the beginning of the simulation of Game Unforgeability, $\mathcal{S}$ generates param using SystemSetup, and sets nominator A's public key to $pk_A = (y_A, Ver^*)$ where $y_A = g^{x_A}$ for a randomly chosen $x_A \in_R \mathbb{Z}_q$. The private key of A is set to $sk_A = (x_A, \perp)$ where $\perp$ denotes an empty string as Sig^* is unavailable to $\mathcal{S}$. For nominee B, the public and private keys are all generated using KeyGen. When $\mathcal{F}$ is invoked, according to Game Unforgeability, 1^k, pk_A and pk_B are given to $\mathcal{F}$ and oracles CreateUser, Corrupt, SignTranscript and Confirmation/disavowal are also simulated. In the following, we describe how SignTranscript is simulated. For a SignTranscript query, there are three cases.

– Case (1 & 2): If $role =$ nil/nominee, a nominative signature is simulated on the querying message m by following the specification of SigGen. There is one exception: if A is indicated as the nominator (i.e. pk_1 in Game Unforgeability), $\mathcal{S}$ is unable to follow the protocol to compute A's standard signature. Therefore, $\mathcal{S}$ forwards the "message" (that is an undeniable signature generated under nominee's private key) to the signing oracle of Sig^* and relays the result back to $\mathcal{F}$.
– Case (3): If $role =$ nominator, $\mathcal{S}$, acting as nominee, simulates a run of SigGen with $\mathcal{F}$. $\mathcal{S}$ can simply follow the exact execution of SigGen.

For a Confirmation/disavowal query, since $\mathcal{S}$ has the first component of all parties' private keys, $\mathcal{S}$ can always carry out the confirmation/disavowal protocol. This also implies that $\mathcal{S}$ can always carry out simulations which are computationally indistinguishable from real simulations no matter they are under passive/active/concurrent attacks.

Reduction Techniques: First, we show that with probability at most $2^{-k}Q$, σ_1^* has been queried to oracle $\mathsf{SignTranscript}$. As restricted by $\mathsf{Game\ Unforgeability}$, $(m^*, pk_A, pk_B, role)$ should have never been queried to oracle $\mathsf{SignTranscript}$. Hence if oracle $\mathsf{SignTranscript}$ has output a nominative signature which contains the undeniable signature σ_1^*, it should be an undeniable signature for some message, say $\hat{m}$, with respect to some nominator and nominee identified by public keys pk_1 and pk_2, respectively. Since $\mathcal{S}$ simulates H by picking values to return uniformly at random from G, the chance that at least there is one execution of $\mathsf{SignTranscript}$ that has σ_1^* as the undeniable signature is at most $2^{-k}Q$. Hence when $\mathcal{F}$ outputs a forgery, σ_2^* must be a forgery with respect to (Sig^*, Ver^*) on "message" σ_1^* with exceptional probability of at most $2^{-k}Q$.

If the advantage of $\mathcal{F}$ in $\mathsf{Game\ Unforgeability}$ is ϵ, the probability that $\mathcal{S}$ existentially forges a signature with respect to (Sig^*, Ver^*) is at least $\epsilon' = (1 - 2^{-k}Q)\epsilon$. If each random oracle query takes at most time t_q to finish, the simulation time of the game is at most $t' = t + Qt_q + c$ where c denotes some constant time for system setup and key generation. □

A.2 Proof of Lemma 2

Proof. Suppose a (t, ϵ, Q)-forger $\mathcal{F}$ has nominator A's private key $sk_A = (x_A, Sig_A)$ and is able to win $\mathsf{Game\ Unforgeability}$ by producing a valid nominative signature $\sigma^* = (\sigma_1^*, \sigma_2^*)$ on some message m^*, we show that in the random oracle model $\mathcal{F}$ can be turned into a (t', ϵ')-algorithm $\mathcal{S}$ which can solve a CDH instance. Suppose the CDH instance is (g, U, V) where $U = g^u$ and $V = g^v$).

In the simulation of $\mathsf{Game\ Unforgeability}$, $\mathcal{S}$ sets the public key of nominee B to $pk_B = (g^u, Ver_B)$ where Ver_B is the signature verification algorithm generated according to the KeyGen algorithm. B's private key is set to $sk_B = (\perp, Sig_B)$ where Sig_B is the corresponding signature generation algorithm of Ver_B. The simulation is similar to that in the proof of Lemma 1 with some exception detailed in the following. For a $\mathsf{SignTranscript}$ query, there are three cases.

- Case (1 & 2): If $role = \mathsf{nil/nominator}$, a nominative signature is simulated on the querying message m by following the specification of SigGen. There is one exception: if B is indicated as the nominee, $\mathcal{S}$ is unable to follow the protocol to compute σ^{undeni} which should be equal to $H(m\|pk_1)^u$. To do so, $\mathcal{S}$ simulates H as follows.

  For each query of $H(message)$ for some $message \in \{0,1\}^*$, $\mathcal{S}$ randomly picks $r \in_R \mathbb{Z}_q$ and sets g^r as the reply.

  Hence $\mathcal{S}$ will set σ^{undeni} as U^r.

– Case (3): If $role =$ nominee, $\mathcal{S}$, acting as nominator, simulates a run of SigGen with $\mathcal{F}$. $\mathcal{S}$ can simply follow the exact execution of SigGen.

For a Confirmation/disavowal query on $(m, \sigma = (\sigma_1, \sigma_2), pk_1, pk_2)$, if B is the nominee, $\mathcal{S}$ has to carry out the confirmation/disavowal protocol as the prover. Although $\mathcal{S}$ does not know the discrete logarithm of U, $\mathcal{S}$ knows the corresponding discrete logarithm of $H(m\|pk_1)$ except when $m = m^*$ and $pk_1 = pk_A$ (note that $pk_2 = pk_B$). We will see shortly in the next paragraph that there is a case that $H(m^*\|pk_A)$ is set to V and hence $\mathcal{S}$ does not know the corresponding discrete logarithm. This case is not going to happen due to the restriction of Game Unforgeability that the tuple $(m^*, \sigma, pk_A, pk_B)$ cannot be queried to Confirmation/disavowal. Thus, $\mathcal{S}$ can always carry out the protocol to show whether $(g, U, H(m\|pk_1), \sigma_1)$ is a DH-tuple or not using WI protocols. This implies that $\mathcal{S}$ can always carry out simulations which are computationally indistinguishable from real simulations no matter they are under passive/active/concurrent attacks.

In the proof of Lemma 1, we show that with probability at least $(1 - 2^{-k}Q)$, σ_1^* has never been queried to oracle SignTranscript. In addition, without querying $H(m^*\|pk_A)$, $\mathcal{F}$ has only 2^{-k} chance to guess the value right. If $\mathcal{F}$ has queried H for $(m^*\|pk_A)$, and if $\mathcal{S}$ has guessed correctly the message m^*, then $\mathcal{S}$ can set $H(m^*\|pk_A)$ to V. Obviously, σ_1^* is the solution of the CDH instance. If $\mathcal{S}$ randomly picks a query of H as the guess of $H(m^*\|pk_A)$, the success probability of $\mathcal{S}$ is $1/Q$. Hence, $\mathcal{S}$ can solve the CDH problem instance with probability at least $\epsilon' = (1 - 2^{-k})(1 - 2^{-k}Q)Q^{-1}\epsilon$. Similar to the proof of Lemma 1, the running time of $\mathcal{S}$ is at most $t' = t + Qt_q + c$. □

A.3 Proof of Theorem 2

Proof. We first show that if there exists a distinguisher with advantage ϵ in Game Invisibility, we can construct a distinguisher with the same advantage in breaking the invisibility of the hash variant of Chaum's undeniable signature scheme described in [15]. Let $\mathcal{D}_{NS}$ denote the distinguisher against our scheme and $\mathcal{D}_{US}$ be the distinguisher against the hash variant of Chaum's scheme. We will show how $\mathcal{D}_{US}$ can use $\mathcal{D}_{NS}$ as a subroutine.

Game Simulation: At the beginning of the simulation of Game Invisibility, $\mathcal{D}_{US}$ uses KeyGen to generate nominator A's public key $pk_A = (y_A, Ver_A)$ and private key $sk_A = (x_A, Sig_A)$. Nominee $B's$ public key is set as $pk_B = (y_B, Ver_B)$ and private key as $sk_B = (\perp, Sig_B)$. Here, y_B is the target public key of $\mathcal{D}_{US}$ and the pair (Ver_B, Sig_B) is generated by $\mathcal{D}_{US}$ using KeyGen. When $\mathcal{D}_{NS}$ is invoked, according to Game Invisibility, 1^k, pk_A and pk_B are given to $\mathcal{D}_{NS}$ and oracles CreateUser, Corrupt, SignTranscript and Confirmation/disavowal are also simulated. In the following, we describe how SignTranscript is simulated.

For a SignTranscript query, there are three cases.

- Case (1 & 2): If $role$ = nil/nominator, a nominative signature is simulated on the querying message m by following the specification of SigGen. There is one exception: if B is indicated as the nominee (i.e. pk_2 in Game Invisibility), $\mathcal{D}_{US}$ is unable to follow the protocol to compute B's undeniable signature. Therefore, $\mathcal{D}_{US}$ forwards the "message= $m\|pk_1$" to the undeniable signing oracle of Chaum's scheme and relays the result back to $\mathcal{D}_{NS}$.
- Case (3): If $role$ = nominee, $\mathcal{D}_{US}$, acting as nominator, simulates a run of SigGen with $\mathcal{D}_{NS}$. $\mathcal{D}_{US}$ can simply follow the exact execution of SigGen.

For a Confirmation/disavowal query, there are two cases:

- Case (1): If B is indicated as the nominee (i.e. pk_2 in Game Invisibility), $\mathcal{D}_{US}$ is unable to follow the protocol to convince or deny a nominative signature $\sigma = (\sigma^{undeni}, \sigma^{standard})$. Therefore, $\mathcal{D}_{US}$ forwards the query $(m\|pk_1, \sigma^{undeni}, y_B)$ to the undeniable Confirmation/Disavowal oracle, and relays all the messages exchanged between the oracle and $\mathcal{D}_{NS}$ accordingly.
- Case (2): Otherwise, $\mathcal{D}_{US}$ can simply follow the exact execution of Confirmation/Disavowal protocol.

Since we use the witness indistinguishable protocols in [15] as the underlying Confirmation/Disavowal protocol, the above simulation can be carried out with an active/concurrent $\mathcal{D}_{NS}$ [15].

At some point in the attacking phase, $\mathcal{D}_{NS}$ will output a message m^* and request a challenge nominative signature σ^* on m^*. Upon receiving m^*, $\mathcal{D}_{US}$ sets a message as $m\|pk_A$ and request a challenging undeniable signature on this message. After obtaining the challenging undeniable signature σ^{undeni}, $\mathcal{D}_{US}$ computes a standard signature $\sigma^{standard} = Sig_B(\sigma^{undeni})$ and sets the challenging nominative signature as $(\sigma^{undeni}, \sigma^{standard})$. After receiving it, $\mathcal{D}_{NS}$ can still access all the oracles and $\mathcal{D}_{US}$ will simulate these oracles as described above. Also note that since the underlying standard signature is strongly unforgeable, it is also infeasible for $\mathcal{D}_{NS}$ to generate another valid standard signature $\bar{\sigma}^{standard} \neq \sigma^{standard}$. Hence, $\mathcal{D}_{US}$ can always carry out the oracle simulations. At the end, $\mathcal{D}_{NS}$ will output its guess b' and $\mathcal{D}_{US}$ will set b' as its own guess.

It is obvious that if the challenging signature σ^{undeni} is a valid undeniable signature of Chaum's scheme, $(\sigma^{undeni}, \sigma^{stand})$ will be a valid nominative signature of our scheme and vice versa. Therefore, $\mathcal{D}_{NS}$ will have the same advantage as $\mathcal{D}_{US}$. According to Theorem 2 of [15], $\mathcal{D}_{US}$ has a negligible advantage in breaking the invisibility of Chaum's scheme with the witness indistinguishable protocols described in [15], under the assumption that Decisional Diffie-Hellman (DDH) problem is hard. Thus, our nominative signature scheme also has the property of invisibility under the Decisional Diffie-Hellman (DDH) assumption, $\square$