Modified Huang-Wang's Convertible Nominative Signature Scheme

Wei Zhao, Dingfeng Ye
State Key Laboratory of Information Security
Graduate University of Chinese Academy of Sciences
Beijing 100049, P. R. China
{wzh,ydf}@is.ac.cn

Abstract

At ACISP 2004, Huang and Wang first introduced the concept of convertible nominative signatures and also proposed a concrete scheme. However, it was pointed out by many works that Huang-Wang's scheme is in fact not a nominative signature. In this paper, we first present a security model for convertible nominative signatures. The properties of Unforgeability, Invisibility, Non-impersonation and Non-repudiation in the setting of convertible nominative signatures are defined formally. Then we modify Huang-Wang's scheme into a secure one. Formal proofs are provided to show that the modified Huang-Wang's scheme satisfies all the security properties under some conventional assumptions in the random oracle model.

Keywords: Digital signature, nominative signature, convertible, selectively, provable security.

1. Introduction

Digital signature, introduced by Diffie and Hellman [3], is a cryptographic means through which the authenticity, data integrity and non-repudiation can be verified. Standard digital signatures have the property that anyone can check whether an alleged message-signature pair is valid or not with respect to a given public key. This publicly verifiable property is necessarily required for some applications of digital signatures such as official announcements. However, this may not be a desired property in some applications, where message to be authenticated are personally private or commercially sensitive. To restrict the public verifiability, some kinds of digital signatures have been proposed, such as nominative signatures (NS).

The concept of nominative signatures was due to Kim, Park and Won [7]. A nominative signature scheme allow a nominative A (i.e. the signer) and a nominee B (i.e. the verifier) to jointly generate a signature σ so that the validity of σ can only be verified by B. Furthermore, if σ is valid, B can convince a third party C of the validity of σ using confirmation protocol; otherwise, B can convince a third party C of the invalidity of σ using disavowal protocol. As suggested in [5, 7, 9], nominative signatures have potential applications in the scenarios where a signed message is personally private or commercially sensitive, such as a tax bill, a medical examination report, ID certification system.

At ACISP 2004, Huang and Wang [5] first added the "convertible" property to nominative signatures, and introduced the concept of convertible nominative signatures (CNS). Moreover, they proposed a concrete scheme based on Kim et al.'s nominative signature scheme [7]. Their scheme enables the nominee to convert a nominative signature into a publicly verifiable one, if necessary.

Unfortunately, in [4, 12, 13], it was found that Huang-Wang's scheme is not nominative in fact. Specially, the nominator in Huang-Wang's scheme can verify the validity of a nominative signature and also show to anyone that the nominative signature is indeed a valid one without the help of the nominee. Hence, Huang-Wang's scheme fails to meet the crucial security requirements of nominative signature: invisibility and non-impersonation.

In this paper, we first give a formal security model of convertible nominative signatures. In the model, the security properties of convertible nominative signatures include Unforgeability, Invisibility, Non-impersonation and Non-repudiation. Then we modify Huang-Wang's scheme to make it satisfy all the properties. Moreover, formal security analysis is provided to show that the modified scheme is provably secure under some standard assumptions in the random oracle model [1].

The rest of paper is organized as follows. In Section 2, we review some basic knowledge and definitions required throughout the paper. In Section 3, we present the definition and security models of convertible nominative signatures. We describe our modified Huang-Wang's convertible nominative signature scheme together with its security analysis in the random oracle model in Section 4. Finally, we conclude the paper in Section 5.

2. Preliminaries

Let p, q be large primes that satisfy $q|p-1$, and g be an element in $\mathbb{Z}_p^*$ with order q. Let $H : \{0,1\}^* \rightarrow \mathbb{Z}_q$ be a public secure hash function. Hereafter, we will use the notation $a \in_R \mathbb{A}$ to mean that a is chosen randomly from $\mathbb{A}$ and use the symbol $\|$ to mean concatenation.

2.1. Intractability Problems

The following three problems are assumed to be hard for any polynomial time algorithm.

1. Discrete Logarithm Problem: Given g, $g^a \in \mathbb{Z}_p^*$ where $a \in \mathbb{Z}_q^*$, find a.

2. Computational Diffie-Hellman Problem: Given g, g^a, $g^b \in \mathbb{Z}_p^*$ where a, $b \in \mathbb{Z}_q^*$, find g^{ab}.

3. Decisional Diffie-Hellman Problem: Given g, g^a, g^b, $g^c \in \mathbb{Z}_p^*$ where a, b and $c \in \mathbb{Z}_q^*$, decide whether $c \stackrel{?}{=} ab$.

2.2. Signature of Equality

Following signature of equality [2] will be used in our convertible nominative signature scheme to convert given nominative signatures into publicly verifiable ones.

A pair (c, s) satisfying $c{=}H(g\|h\|y\|z\|g^s y^c\|h^s z^c\|m)$ is *signature of equality* of the discrete logarithm of y with respect to the base g and the discrete logarithm of z with respect to the base h for the message m and is denoted by $SEQDL(g,\, h,\, y,\, z,\, m)$.

A $SEQDL(g,\, h,\, y,\, z,\, m)$ can only be computed if the secret key $x = log_g y = log_h z$ is known, by choosing $k \in_R \mathbb{Z}_q^*$, and computing c and s according to

$$c = H(g\|h\|y\|z\|g^k\|h^k\|m),$$

$$s = k - cx \pmod q.$$

3. Definition and Security Model of Convertible Nominative Signature

In this section, we extend the definition and security model of nominative signature [8, 9] to the setting of convertible nominative signature. We will let A, B and C to denote the nominator, the nominee, and the verifier (a third party) throughout the paper.

3.1. Definition of Convertible Nominative Signature

The convertible nominative signature scheme consists of the following algorithms and protocols:

- **System Setup:** a probabilistic algorithm that on input 1^k where $k \in \mathbb{N}$ is a security parameter, generates the common parameters denoted by cp.
- **Key Generation:** a probabilistic algorithm that on input cp, generates a public/private key pair (pk, sk) for a user in the system.
- **Signing Protocol:** an interactive (or non-interactive) algorithm. The common inputs of A and B are cp and a message m. A has an additional input pk_B, indicating that A nominates B as the nominee; and B has an additional input pk_A, indicating that A is the nominator. At the end of the protocol, either A or B outputs a convertible nominative signature σ, or $\perp$ indicating the failure of the protocol.
- **Ver$^{\text{nominee}}$(nominee-only verification):** a deterministic algorithm that on input the common parameters cp, a nominative message-signature pair (m, σ), A'public key pk_A and B'private key sk_B, returns $valid$ or $invalid$.
- **Confirmation/Disavowal Protocol:** an interactive (or non-interactive) algorithm between B and C. On input the common parameters cp and $(m,\ \sigma,\ pk_A,\ pk_B)$, B sets a bit μ to 1 if $valid \leftarrow$ **Ver$^{\text{nominee}}$** (m, σ, 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.
- **Selectively Convert:** a probabilistic (or deterministic) algorithm that on input the common parameters cp, the public/private key pair (pk_B, sk_B), A'public key pk_A and a valid message-signature pair (m, σ), outputs a selective proof $P^{m,\,\sigma}_{pk_A, pk_B}$ of the given message-signature pair.
- **Selectively Verify:** a deterministic algorithm that on input the common parameters cp, the public keys pk_A and pk_B, the message-signature pair (m, σ) and the selective proof $P^{m,\,\sigma}_{pk_A, pk_B}$, outputs $accept$ or $reject$.

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

1. $valid \leftarrow$**Ver$^{\text{nominee}}$**(m, σ, pk_A, sk_B);
2. C outputs $accept$ at the end of **Confirmation** protocol;
3. On input (m, σ) together with a valid selective proof $P^{m,\,\sigma}_{pk_A, pk_B}$, **Selectively Verify** algorithm outputs $accept$.

Validity of a Convertible Nominative Signature: A convertible nominative signature σ is said to be valid on m with respect to pk_A and pk_B if $valid \leftarrow \mathbf{Ver^{nominee}}(m, \sigma, pk_A, sk_B)$ where sk_B is the corresponding private key of pk_B.

The security model of convertible nominative signature will be defined using the game between an adversary and a simulator. We allow the adversary $\mathcal{F}$ to access the following oracles and submit their queries to the simulator $\mathcal{S}$ adaptively:

- CreateUser Oracle: On input an identity, say I, it generates a key pair (pk_I, sk_I) using **Key Generation** algorithm and returns pk_I.
- Corrupt Oracle: On input a public key pk, if pk is generated by CreaterUser Oracle or in $\{pk_A, pk_B\}$, the corresponding private key is returned; otherwise, $\perp$ is returned. pk is said to be corrupted.
- Signing Oracle: 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 $role \in \{$nil, nominator, nominee$\}$,
  - if $role$ is nil, $\mathcal{S}$ simulates a run of **Signing** protocol and then returns a valid convertible nominative signature σ and a transcript of the execution of **Signing** protocol.
  - If $role$ is nominator, $\mathcal{S}$ (as nominee with public key pk_2) simulates a run of **Signing** protocol with $\mathcal{F}$ (as nominator with public key pk_1).
  - If $role$ is nominee, $\mathcal{S}$ (as nominator with public key pk_1) simulates a run of **Signing** protocol with $\mathcal{F}$ (as nominee with public key pk_2).
- Confirmation/Disavowal Oracle: On input a message m, a nominator signature σ and two public keys pk_1 (the nominator) and 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 $\mathbf{Ver^{nominee}}(m, \sigma, pk_1, sk_2) = valid$, the oracle returns a bit $\mu = 1$ and a transcript of **Confirmation** protocol. Otherwise, $\mu = 0$ and a transcript of **Disavowal** protocol is returned.
  - In an active/concurrent attack, if $\mathbf{Ver^{nominee}}(m, \sigma, pk_1, sk_2)$=$valid$, the oracle returns $\mu = 1$ and executes **Confirmation** protocol with $\mathcal{F}$ (acting as a verifier). Otherwise, the oracle returns $\mu = 0$ and executes **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.
- Selectively Convert Oracle: On input a message m, a nominative signature σ and two public keys pk_1 (the nominator) and pk_2 (the nominee), it runs **Selectively Convert** algorithm to generate the selective proof $P^{m,\sigma}_{pk_A, pk_B}$ and returns it to $\mathcal{F}$.

The security notions for convertible nominative signature include: Unforgeability, Invisibility, Non-impersonation and Non-repudiation. We will make detailed descriptions for them in the following subsections.

3.2. Unforgeability

The *existential unforgeability* means that an adversary should not be able to forge a valid convertible nominative signature if at least one of the private keys of A and B is not known. The adversary in our definition is allowed to access to the CreatUser Oracle, Corrupt Oracle, Signing Oracle and Confirmation/Disavowal Oracle. Furthermore, we also allow the adversary to submit queries to Selectively Convert Oracle. This is to ensure that the knowledge of the selective proof cannot help the adversary to forge a new valid message-signature pair.

To discuss the unforgeability of our convertible nominative signatures, we divide the potential adversaries into the following three types:

- **Adversary 0** who has only the public keys of the nominator A and the nominee B.
- **Adversary I** who has the public keys of the nominator A and the nominee B and also has B's private key;
- **Adversary II** who has the public keys of the nominator A and the nominee B and also has A's private key.

We can easily find that if a convertible nominative signature scheme is unforgeable against **Adversary I** (or **Adversary II**), then it is also unforgeable against **Adversary 0**.

Game Unforgeability (Adversary I): Let $\mathcal{S}$ be the simulator and $\mathcal{F}_I$ be the adversary.

1. (Initialization Phase) Let $k \in \mathbb{N}$ be a security parameter. First, $cp \leftarrow$ **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 **Key Generation** algorithm. $\mathcal{F}_I$ is invoked with inputs 1^k, pk_A, pk_B.

2. (Attacking Phase) $\mathcal{F}_I$ can make queries to the oracles mentioned above;

3. (Output Phase) $\mathcal{F}_I$ outputs a pair (m^*, σ^*).

$\mathcal{F}_I$ $wins$ the game if $valid \leftarrow \mathbf{Ver}^{\mathbf{nominee}}(m^*, \sigma^*, pk_A, sk_B)$ and (1) $\mathcal{F}_I$ has never corrupted pk_A; (2) $(m^*, pk_A, pk_B, role)$ has never been queried to Signing Oracle for any valid value of $role$. $\mathcal{F}_I$'s advantage in this game is defined to be $Adv(\mathcal{F}_I) = \Pr[\mathcal{F}_I \text{ wins}]$.

Game Unforgeability (Adversary II): It is defined similarly to the above game. Specially, the descriptions of all phases are the same as the above game, so we omit them. When all phases are over,

$\mathcal{F}_{II}$ $wins$ the game if $valid \leftarrow \mathbf{Ver}^{\mathbf{nominee}}(m^*, \sigma^*, pk_A, sk_B)$ and (1) $\mathcal{F}_{II}$ has never corrupted pk_B; (2) $(m^*, pk_A, pk_B, role)$ has never been queried to Signing Oracle for any valid value of $role$; (3) $(m^*, \sigma', pk_A, pk_B)$ has never been queried to Confirmation/Disavowal Oracle for any convertible nominative signature σ' with respect to pk_A and pk_B. $\mathcal{F}_{II}$'s advantage in this game is defined to be $Adv(\mathcal{F}_{II}) = \Pr[\mathcal{F}_{II} \text{ wins}]$.

Definition 1 *A convertible nominative signature scheme is said to be existential unforgeable if no probabilistic polynomial time (PPT) adversaries $\mathcal{F}_I$ and $\mathcal{F}_{II}$ have a non-negligible advantage in the above games.*

3.3. Invisibility

We now extend the property *invisibility* for nominative signatures into the setting of convertible nominative signature. This property essentially means that it is impossible for an adversary to determine whether a given message-signature pair (m, σ) is valid without the help of the nominee and the selective proof $P^{m,\sigma}_{pk_A, pk_B}$.

Game Invisibility: Let $\mathcal{D}'$ be the simulator and $\mathcal{D}$ be the distinguisher.

1. (Initialization Phase) The initialization phase is the same as that of **Game Unforgeability**.

2. (Preparation Phase) The distinguisher $\mathcal{D}$ can adaptively access to all the oracles. At some point, $\mathcal{D}$ submits the challenge $(m^*, pk_A, pk_B, role)$ to Signing Oracle.

   Then $\mathcal{D}$ (acting as nominator) will carry out a run of **Signing** protocol with the simulator $\mathcal{D}'$ (acting as nominee). Let σ^{valid} be the convertible nominative signature generated by the simulator $\mathcal{D}'$ at the end of the protocol.

   The challenge signature σ^* is then generated based on the outcome of a random coin toss b. If $b = 1$, $\mathcal{D}'$ sets $\sigma^* = \sigma^{valid}$. If $b = 0$, σ^* is chosen uniformly at random from the signature space of the convertible nominative signature scheme with respect to pk_A and pk_B. Then the challenging signature σ^* is returned to $\mathcal{D}$.

3. (Guessing Phase) Finally, the distinguisher $\mathcal{D}$ outputs a guess b'.

$\mathcal{D}$ $wins$ the game if $b' = b$ and (1) pk_B has never been submitted to Corrupt Oracle; (2) $(m^*, pk_A, pk_B, role)$ has never been submitted to Signing Oracle; (3) $(m^*, \sigma^*, pk_A, pk_B)$ has never been submitted to Selectively Convert Oracle; (4) $(m^*, \sigma', pk_A, pk_B)$ has never been submitted to Confirmation/Disavowal Oracle for any convertible nominative signature σ' on m^* with respect to pk_A and pk_B. $\mathcal{D}$'advantage in this game is defined to be $Adv(\mathcal{D}) = |\Pr[b' = b] - \frac{1}{2}|$.

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

3.4. Non-impersonation

The notion of *non-impersonation* means that the validity of a nominative signature can only be determined by the help of the nominee, someone else including the nominator should not be able to show the validity of the nominative signature to a third party. Concretely, this notion requires that:

1. Only with the knowledge of the public key of the nominee B, it should be difficult for an impersonator $\mathcal{I}_I$ to execute **Confirmation/Disavowal** protocol.

2. Only with the knowledge of the public key of the nominee B, it should be difficult for an impersonator $\mathcal{I}_{II}$ to generate the selective proof for a message-signature pair.

Game Impersonation of Confirmation/Disavowal Protocol: Let $\mathcal{S}$ be the simulator and $\mathcal{I}_I$ be the impersonator.

1. (Initialization Phase) The initialization phase is the same as that of **Game Unforgeability**.

2. (Preparation Phase) In this phase, impersonator $\mathcal{I}_I$ is permitted to access all the oracles. $\mathcal{I}_I$ prepares a triple (m^*, σ^*, μ) where m^* is some message, σ^* is a convertible nominative signature and μ is a bit.

3. (Attacking Phase) If $\mu = 1$, $\mathcal{I}_I$ (as nominee) executes **Confirmation** protocol with the simulator $\mathcal{S}$ (as

a verifier) on common inputs $(m^*, \sigma^*, pk_A, pk_B)$. If $\mu = 0, \mathcal{I}_I$ executes **Disavowal** protocol with simulator $\mathcal{S}$ on the same inputs.

The impersonator $\mathcal{I}_I$ *wins* the game if the simulator acting as the verifier outputs *accept* while $\mathcal{I}_I$ has the following restrictions: $\mathcal{I}_I$ has never submitted pk_B to the Corrupt Oracle. $\mathcal{I}_I$'s advantage in this game is defined to be $Adv(\mathcal{I}_I) = \Pr[\mathcal{I}_I \text{ wins}]$.

Game Impersonation of Selectively Convert Algorithm: Let $\mathcal{S}$ be the simulator and $\mathcal{I}_{II}$ be the impersonator.

1. (Initialization Phase) The initialization phase is the same as that of **Game Unforgeability**.
2. (Preparation Phase) The impersonator $\mathcal{I}_{II}$ is invoked on input $1^k, pk_A, pk_B$ and permitted to issue queries to all the oracles.
3. (Impersonation Phase) The impersonator $\mathcal{I}_{II}$ outputs a valid selective proof $P^{m^*,\sigma^*}_{pk_A,pk_B}$ for a message-signature pair (m^*, σ^*).

The impersonator $\mathcal{I}_{II}$ *wins* the game if $P^{m^*,\sigma^*}_{pk_A,pk_B}$ satisfies **Selectively Verify** algorithm but: (1) pk_B has never been submitted to Corrupt Oracle; (2) $(m^*, \sigma^*, pk_A, pk_B)$ has never queries Selectively Convert Oracle. $\mathcal{I}_{II}$'s advantage in this game is defined to be $Adv(\mathcal{I}_{II}) = \Pr[\mathcal{I}_{II} \text{ wins}]$.

Definition 3 *A convertible nominative signature scheme is said to be secure against impersonation if no PPT impersonators $\mathcal{I}_I$ and $\mathcal{I}_{II}$ have a non-negligible advantage in the above games.*

3.5. Non-repudiation

The notion of *non-repudiation* requires that the nominee cannot convince a verifier C that a valid (invalid) convertible nominative signature is invalid (valid).

Game Non-repudiation: Let $\mathcal{S}$ be the simulator and $\mathcal{B}$ be the cheating nominee.

1. (Initialization Phase) The initialization phase is the same as that of **Game Unforgeability**.
2. (Preparation Phase) $\mathcal{B}$ prepares (m^*, σ^*, μ) where m^* is some message and σ^* is a nominative signature. $\mu = 1$ if $\mathbf{Ver}^{\mathbf{nominee}}(m^*, \sigma^{valid}, pk_A, sk_B) = valid$; otherwise, $\mu = 0$.
3. (Repudiation Phase) If $\mu = 1$, $\mathcal{B}$ executes **Disavowal** protocol with the simulator $\mathcal{S}$ (acting as a verifier) on $(m^*, \sigma^{valid}, pk_A, pk_B)$ but the first bit sent to $\mathcal{S}$ is 0. If $\mu = 0$, $\mathcal{B}$ executes **Confirmation** protocol with simulator $\mathcal{S}$ but the first bit sent to $\mathcal{S}$ is 1.

$\mathcal{B}$ *wins* the game if the simulator acting as the verifier outputs *accept*. $\mathcal{B}$'s advantage in this game is defined to be $Adv(\mathcal{B}) = \Pr[\mathcal{B} \text{ wins}]$.

Definition 4 *A convertible 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 the above game.*

4. Modified Huang-Wang's Convertible Nominative Signature Scheme

In this section, we will describe the modified Huang-Wang's convertible nominative signature scheme and make a detailed formal security analysis in the random oracle model [1].

4.1. Scheme

We now modify the Huang-Wang's convertible nominative signature scheme [5] into a secure one. The modified Huang-Wang's scheme is as follows.

- **System Setup:** Let p, q be two large primes such that $q|p-1$, and g an element in $\mathbb{Z}_p^*$ of order q. Assume that the discrete logarithm problem in the group $\langle g \rangle$ is hard. In addition, two one-way hash functions $H_1 : \{0,1\}^* \rightarrow \langle g \rangle$ and $H_2 : \{0,1\}^* \rightarrow \mathbb{Z}_q$ is publicly available.
- **Key Generation:** The nominator A and the nominee B set their public/private key pairs as (y_A, x_A) and (y_B, x_B) respectively, where $x_A, x_B \in_R \mathbb{Z}_q^*$, $y_A = g^{x_A} \bmod p$ and $y_B = g^{x_B} \bmod p$.
- **Signing Protocol:** To generate a nominative signature $\sigma = (b, c, s)$ for a message m, the nominator A and the nominee B jointly perform as follows.
  1. The nominee B first picks $R \in_R \mathbb{Z}_q^*$, then sends (a, c) to the nominator A by computing
  $$a = g^R \bmod p,$$
  $$c = H_1(m \| y_A \| y_B)^{x_B} \bmod p.$$
  2. Upon receiving (a, c), the nominator A chooses $r \in_R \mathbb{Z}_q^*$, and sends (b, c, s') to B by computing
  $$b = ag^{-r} \bmod p,$$
  $$e = H_2(y_A \| y_B \| b \| c \| m),$$
  $$s' = r - x_A \cdot e \pmod q.$$

3. Then nominee B checks whether both of the following equations hold:

$$e \equiv H_2(y_A \| y_B \| b \| c \| m),$$

$$a \equiv g^{s'} y_A^e b \bmod p.$$

If not, outputs "$False$". Otherwise, nominee B outputs $\sigma = (b, c, s)$ as the nominative signature for message m by setting $s = s' + x_B - R \pmod q$.

We say that $\sigma = (b,\ c,\ s)$ is a convertible nominative signature (i.e. σ is in the signature space with respect to pk_A and pk_B) if $b, c \in \mathbb{Z}_p$, $s \in \mathbb{Z}_q$ and

$$g^s y_A^e b \equiv y_B \bmod p.$$

- **Ver$^{\text{nominee}}$:** Given a nominative signature $\sigma = (b, c, s)$ and a message m, the nominee B accepts σ as $valid$ if and only if

$$g^s y_A^e b \equiv y_B \bmod p,$$

$$c = H_1(m \| y_A \| y_B)^{x_B} \bmod p.$$

- **Confirmation/Disavowal Protocol:** On input (m, σ, y_A, y_B) where σ is a convertible nominative signature, if **Ver$^{\text{nominee}}$**$(m, \sigma, y_A, x_B) = valid$, B sends $\mu = 1$ to verifier C; otherwise, $\mu = 0$ is sent to C. B then proves to C that the tuple $(g, y_B, H_1(m \| y_A \| y_B), c)$ is a DH-tuple or not according to the value of u using WI protocols [6].

- **Selectively Convert:** When the nominee B wants to convert a nominative signature $\sigma = (b, c, s)$ into a publicly verifiable one, he chooses $k \in_R \mathbb{Z}_q^*$ and computes the selective proof $P^{m,\sigma}_{pk_A, pk_B}$ as

$$SEQDL(g, H_1(m \| y_A \| y_B), y_B, c, \sigma) = (c', s')$$

where $c' = H_2\ (g \parallel H_1\ (m \parallel\ y_A \parallel\ y_B) \parallel\ y_B \parallel\ c \parallel\ g^k \parallel\ (H_1(m \| y_A \| y_B))^k \| \sigma)$ and $s' = k - c \cdot x_B \pmod p$. Then B publishes (c', s').

- **Selectively Verify:** Anyone can verify the nominative signature $\sigma = (b, c, s)$ with its selective proof $P^{m,\sigma}_{pk_A, pk_B} = (c', s')$ by verifying the corresponding signature of equality $SEQDL$.

Remark. We say that (g, g^u, g^v, g^w) is a DH-tuple if $w \equiv uv \pmod q$; otherwise, it is a non-DH-tuple. As shown in [6], using WI protocol, a prover who knows the knowledge of either one of the witnesses, i.e. u or v, can prove that whether the tuple (g, g^u, g^v, g^w) is a DH-tuple or not. In **Confirmation/Disavowal** protocol of our scheme, B's knowledge is x_B. We will use the WI protocol [6] for concrete implementation.

4.2. Security Analysis

In this section, we give a formal security analysis of the modified Huang-Wang's scheme.

Lemma 1 (Adversary I) *The modified Huang-Wang's convertible nominative signature scheme is existential unforgeability against Adversary I if DLP problem is hard.*

Proof : Suppose there exists a (t, ϵ, Q)-forger $\mathcal{F}_I$ who can forge a valid signature with probability at least ϵ after running at most time t and making at most Q queries, then we show that there exists a (t', ϵ')-algorithm $\mathcal{S}$ who can solve the DLP problem in $\mathbb{G}$ by running $\mathcal{F}_I$ as a subroutine. Let $(g, U = g^u)$ be a random instance of the DLP problem where $g,\ g^u \in \mathbb{Z}_p^*$, $\mathcal{S}$ will simulate all the oracles and answer $\mathcal{F}_I$'s queries as follows.

$\mathcal{S}$ first generates cp according to **System Setup** algorithm and sets nominator A's public key $y_A = U$. B's public/private key pair (y_B, x_B) is generated using **Key Generation** algorithm accordingly.

- Random Oracles: In order to respond $\mathcal{F}_I$'s queries to random oracles, $\mathcal{S}$ will maintain two lists: H_1-$list$ and H_2-$list$.

  1. H_1-query: At any time, $\mathcal{F}_I$ can make a H_1 query for $m \| y_1 \| y_2$. In response, $\mathcal{S}$ will maintain a H_1-$list$ which stores his response to such queries. For a new query, $\mathcal{S}$ checks H_1-$list$ to see if the same query has been made before, if so, the same answer will be returned; otherwise, $\mathcal{S}$ chooses a random number r_1 from $\mathbb{Z}_q^*$ and sets $H_1(m \| y_1 \| y_2) = g^{r_1} \bmod p$. Then $\mathcal{S}$ adds $(m \| y_1 \| y_2, g^{r_1} \bmod p, r_1)$ into H_1-$list$ and returns $g^{r_1} \bmod p$ as the answer.
  2. H_2-query: When $\mathcal{F}_I$ make a H_2 query for $y_1 \| y_2 \| b \| c \| m$, in response, $\mathcal{S}$ will maintain a H_2-$list$ which stores his response to such queries. For a new query, $\mathcal{S}$ checks H_2-$list$ to see if the same query has been made before, if so, the same answer will be returned; otherwise, $\mathcal{S}$ chooses a random number r_2 from $\mathbb{Z}_q$ and sets $H_2(y_1 \| y_2 \| b \| c \| m) = r_2$. Then $\mathcal{S}$ adds $(y_1 \| y_2 \| b \| c \| m, r_2)$ into H_2-$list$ and returns r_2 as the answer.

- CreaterUser oracle: For a CreateUser query for identity I, in response, $\mathcal{S}$ will generate the public/private key pair (y_I, x_I) using **Key Generation** algorithm and return y_I.

- Corrupt Oracle: $\mathcal{F}_I$ can make a corrupt query for public key y_I, $\mathcal{S}$ will return x_I as the answer. As restricted, $\mathcal{F}_I$ cannot query Corrupt Oracle for A's private key.

- Signing Oracle: We assume that when $\mathcal{F}_I$ requests a signature on (m, y_1, y_2), it has already made the corresponding H_1 query on (m, y_1, y_2). At any time, $\mathcal{F}_I$ can submit a signing query (m, y_1, y_2), there are three cases to handle.
  - Case (1): If $role$ =nil, the simulation will be carried out exactly according to **Signing** protocol except in the following two sub-cases:
    1. If $y_1 = y_A$, i.e. A is indicated as the nominator. For this case, since $\mathcal{S}$ does not know A's private key, he is not able to generate a valid nominative signature using **Signing** protocol directly. In this situation, $\mathcal{S}$ will compute the nominative signature (b, c, s) by following the steps below: (1) chooses randomly $R \in \mathbb{Z}_q^*$ and sets $a = g^R \bmod p$, $c = (g^{r_1})^{x_2} \bmod p = y_2^{r_1} \bmod p$ where $g^{r_1} \bmod p$ is the answer of H_1 query; (2) chooses randomly $k \in \mathbb{Z}_q^*$, sets $e = r_2$, $s' = k$ and $b = ag^{-s'}y_A^{-e} \bmod p = ag^{-k}y_A^{-r_2} \bmod p$; (3) sets $s = s' + x_2 - R \pmod q$.
    2. If $y_2 = y_A$, i.e. A is indicated as nominee. For this case, $\mathcal{S}$ first chooses randomly $R \in \mathbb{Z}_q^*$ and sets $a = g^R \bmod p$ and $c = (g^r)^{x_A} \bmod p = y_A^{r_1} \bmod p$ where $g^{r_1} \bmod p$ is the answer of H_1 query; then since $\mathcal{S}$ does not know A's private key, it is not able to compute s directly, in this situation, $\mathcal{S}$ will choose randomly $l \in \mathbb{Z}_q$ and sets $s = l$ and $b = y_A g^{-s} y_1^{-e} \bmod p = y_A g^{-l} y_1^{-r_2} \bmod p$ where r_2 is the answer of H_2 query; finally, $\mathcal{S}$ returns (b, c, s) as the response.
  - Case (2): If role=nominator, $\mathcal{S}$ simulates the behavior of a nominee and interacts with $\mathcal{F}_I$ according to **Signing** protocol except the following subcase: if $y_2 = y_A$, similar to the subcase 2 in case (1).
  - Case (3): If role=nominee, $\mathcal{S}$ simulates the behavior of a nominator and interacts with $\mathcal{F}_I$ according to **Signing** protocol except the following subcase: if $y_1 = y_A$, similar to the subcase 1 in case (1).
- Confirmation/Disavowal Oracle: When $\mathcal{F}_I$ makes a confirmation/disavowal query on (m, σ, y_1, y_2), $\mathcal{S}$ simulates **Confirmation/Disavowal** protocol accordingly except the following case: if $y_2 = y_A$, i.e. A is indicated as the nominee, $\mathcal{S}$ does not know A's private key to prove a DH-tuple/non-DH-tuple $(g, y_B, H_1(m\|y_1\|y_A, c)$. In this situation, $\mathcal{S}$ will use its knowledge r_1 to execute the WI protocol, where $g^{r_1} \bmod p$ is the answer of query $H_1(m\|y_1\|y_A)$.
- Selectively Convert Oracle: When $\mathcal{F}_I$ makes a selectively convert query on (m, σ, y_1, y_2), $\mathcal{S}$ chooses randomly $c \in \mathbb{Z}_p$ and $s \in \mathbb{Z}_q$ and sets (c, s) as the answer.

After all the queries, $\mathcal{F}_I$ outputs a valid forgery $(m^*, \sigma^*, y_A, y_B)$ with the restrictions defined in Section 3.2. Therefore, $\sigma^* = (b^*, c^*, s^*)$ satisfies $g^{s^*} y_A^{e^*} b^* \equiv y_B \pmod p$, $c^* = H_1(m^*\|y_A\|y_B)^{x_B} \pmod p$ and $e^* = H_2(y_A\|y_B\|b^*\|c^*\|m^*)$, then $\mathcal{F}_I$ can forge a s^* satisfying $g^{s^*} y_A^{e^*} \equiv g^{r'} \pmod p$. In other words, he can forge a valid Schnorr's signature [11]. It is known that Schnorr's signature scheme is existential unforgeable under DLP problem, hence $\mathcal{S}$ will solve DLP problem.

To complete the proof, it remains to calculate the probability ε' that $\mathcal{S}$ solves the DLP problem and the time t' that $\mathcal{S}$ runs. The success probability of $\mathcal{S}$ is at least ϵ. The time t' of running is at most $t + Qt_q + c$ where t_q is the maximum time for simulating one oracle query and c denotes some constant time of system setup and key generation. This completes our proof. □

We leave the following security proofs in the appendices.

Lemma 2 (Adversary II) *The modified Huang-Wang's convertible nominative signature scheme is existentially unforgeable against Adversary II if CDH problem is hard.*

Theorem 1 *The modified Huang-Wang's convertible nominative signature scheme is **existential unforgeable** if both DLP and CDH problems are hard.*

Proof: The proof of this theorem follows from Lemma 1 and Lemma 2.

Theorem 2 *The modified Huang-Wang's convertible nominative signature scheme has the property of **invisibility** if the DDH problem is hard.*

Theorem 3 *The modified Huang-Wang's convertible nominative signature scheme is secure against **impersonation** if DLP problem is hard.*

Theorem 4 *The modified Huang-Wang's convertible nominative signature scheme is secure against **repudiation** by nominee.*

4.3. Comparison

Compared with Huang-Wang's scheme, our scheme additionally employ a hash function in the signing protocol, so it is slightly less efficient. However it offers formal security analysis under a reasonable security model, while Huang-Wang's scheme is in fact not a secure scheme.

5. Conclusions

In this paper, we first presented a security model of convertible nominative signatures and then modified Huang-Wang's scheme to be secure in this model. Meanwhile, all the security properties of the modified Huang-Wang's scheme were formally proven under some conventional complexity assumptions in the random oracle model.

References

[1] M. Ballare and P. Rogaway, "Random Oracles Are Practical: A Paradigm for Designing Efficient Protocols", *Proceedings of the First Annual Conference on Computer and communications Security*, ACM, 1993, pp.62-73.

[2] J. Camenisch, "Efficient and Generalized Group Signatures", *Advance in Cryptology-EUROCRYPT'97*, LNCS 1233, pp.465-479, Springer-Verlag, 1997.

[3] W. Diffie and M. Hellam, "New Directions in Cryptography", IEEE IT22, pp.644-654, 1976.

[4] L. Guo, G. Wang and D. Wang, "Further Discussions on the Security of a Nominative Signature Scheme", IACR erpint archive, http://eprint.iacr.org/2006/007.

[5] Z. Huang and Y. Wang, "Convertible Nominative Signatures", In: *Proc. of Information Security and Privacy (ACISP'04)*, LNCS 3108, pp.181-357, Springer-Verlag, 2005.

[6] K. Kurosawa and S. Heng, "3-Move Unideniable Signature Scheme", In: R.J.F.Cramer(ed.) *EUROCRYPT 2005*, LNCS 3494, pp.181-197, Springer-Verlag, 2005.

[7] S. J. Kim, S. J. Park and D. H. Won, "Zero-knowledge Nominative Sigantures", In *Pragocrypt'96, International Conference on the Theory and Applications of Cryptology*, pp.380-392, 1996.

[8] D. Y. W. Liu, Q. Huang and D. S. Wong, "An Efficient One-move Nominative Signature Scheme", IACR eprint archive, http://eprint.iacr.org/2007/260.

[9] D. Y. W. Liu, D. S. Wong, X. Huang, G. Wang, Q. Huang, Y. Mu and W. Susilo, "Formal Definition and Construction of Nomiantive Signature", S.Qing, H.Imai and G.Wang(eds): *ICICS 2007*, LNCS 4861, pp.57-68, 2007.

[10] D. Pointcheval and J. Stern, "Security Proofs for Signature Schemes", In: U.M.Maurer(eds) *EUROCRYPT 1996*, LNCS, vol.1070, pp.387-398, Springer, Heidelberg(1996).

[11] C. P. Schnorr, "Efficient Signature Generation for Smart Cards", *Journal of Crryptology*, 1991(4), pp.239-252.

[12] W. Susilo and Y. Mu, "On the Security of Nominative Signatures", In: *Proc. of Information Security and Privacy (ACISP05)*, LNCS 3547, pp.329-335. Springer-Verlag, 2005.

[13] G. Wang and F. Bao, "Security Remarks on a Convertible Nomiantive Signature Scheme", In *IFIP International Federation for Information Processing*, Volume 232, New Approaches for Security, Privacy and Trust in Complex Environments, eds. H.Venter, M.Eloff, L.Labuschague, J.Eloff, R.Vonsolms, (Boston:Springer), pp.265-275.

A. Proof of Lemma 2

Proof : Suppose there exists a (t, ϵ, Q)-forger $\mathcal{F}_{II}$ who can forge a valid signature with probability at most ϵ after running at most time t and making at most Q queries, then we show that there exists a (t', ϵ')-algorithm $\mathcal{S}$ who can solve the CDH problem in $\mathbb{G}$ by running $\mathcal{F}_{II}$ as a subroutine. Let $(g, U = g^u, V = g^v)$ be a random instance of the CDH problem where g, g^u, $g^v \in \mathbb{Z}_p^*$, $\mathcal{S}$ will simulate all the oracles and answer $\mathcal{F}_{II}$'s queries as follows.

$\mathcal{S}$ first generates cp according to **System Setup** algorithm and sets nominator B's public key $y_B = U$. A's public/private key pair (y_A, x_A) is generated using **Key Generation** algorithm accordingly. Let q_{H_1} be the number of H_1 queries that $\mathcal{F}_{II}$ issues.

- Random Oracles: In order to respond $\mathcal{F}_{II}$'s queries to random oracles, $\mathcal{S}$ will maintain two lists: H_1-$list$ and H_2-$list$.

  1. H_1-query: At any time, $\mathcal{F}_{II}$ can make a H_1 query for $m\|y_1\|y_2$. In response, $\mathcal{S}$ will maintain a H_1-$list$ which stores his response to such queries. Among the q_{H_1} H_1 queries, $\mathcal{S}$ randomly chooses one of the H_1 queries that are in the form $m\|y_A\|y_B$, say $\widetilde{m}\|y_A\|y_B$, and sets $H_1(\widetilde{m}\|y_A\|y_B)$=$(g^v)^{r_1}$ mod p. Other queries, however, will be set as g^{r_1} mod p where r_1 is chosen randomly from $\mathbb{Z}_q^*$. Then $\mathcal{S}$ adds $(m\|y_1\|y_2, H_1(m\|y_1\|y_2), r_1)$ into H_1-$list$ and returns $H_1(m\|y_1\|y_2)$ as the answer.

  2. H_2-query: When $\mathcal{F}_{II}$ make a H_2 query for $y_1\|y_2\|b\|c\|m$, in response, $\mathcal{S}$ will maintain a H_2-$list$ which stores his response to such queries. For a new query, $\mathcal{S}$ checks H_2-$list$ to see if the same query has been made before, if so, the same answer will be returned; otherwise, $\mathcal{S}$ chooses a random number r_2 from $\mathbb{Z}_q$ and sets $H_2(y_1\|y_2\|b\|c\|m) = r_2$. Then $\mathcal{S}$ adds $(y_1\|y_2\|b\|c\|m, r_2)$ into H_2-$list$ and returns r_2 as the answer.

- CreateUser Oracle: For a CreateUser query for identity I, in response, $\mathcal{S}$ will generate the public/private key pair (y_I, x_I) using **Key Generation** algorithm and return y_I.

- Corrupt Oracle: $\mathcal{F}_{II}$ can make a corrupt query for public key y_I, $\mathcal{S}$ will return x_I as the answer. As restricted, $\mathcal{F}_{II}$ cannot query Corrupt Oracle for B's private key.

- Signing Oracle: We assume that when $\mathcal{F}_{II}$ requests a signature on (m, y_1, y_2), it has already made the corresponding H_1 query on (m, y_1, y_2) and H_2 query on $(y_1\|y_2\|b\|c\|m)$. At any time, $\mathcal{F}_{II}$ can submit a signing query (m, y_1, y_2), there are three cases to handle.
  - Case (1): If $role$ =nil, the simulation will be carried out exactly according to **Signing** protocol except in the following two sub-cases:
    1. If $y_1 = y_B$, i.e. B is indicated as the nominator. For this case, since $\mathcal{S}$ does not know B's private key, he is not able to generate a valid nominative signature using **Signing** protocol directly. In this situation, $\mathcal{S}$ will compute the nominative signature (b, c, s) by following the steps below: (1) chooses randomly $R \in \mathbb{Z}_q^*$ and sets $a = g^R \bmod p$, $c = (g^{r_1})^{x_2} \bmod p = y_2^{r_1} \bmod p$ where g^{r_1} is the answer of H_1 query; (2) chooses randomly $k \in \mathbb{Z}_q^*$, sets $e = r_2$, $s' = k$ and $b = ag^{-s'}y_B^{-e} \bmod p = ag^{-k}y_B^{-r_2} \bmod p$; (3) sets $s = s' + x_2 - R \pmod q$.
    2. If $y_2 = y_B$, i.e. B is indicated as the nominee. If both $y_1 = y_A$ and $m = \widetilde{m}$ are satisfied, $\mathcal{S}$ aborts and fails to solve the CDH problem. Otherwise, $\mathcal{S}$ first chooses randomly $R \in \mathbb{Z}_q^*$ and sets $a = g^R \bmod p$ and $c = (g^{r_1})^{x_B} \bmod p = y_B^{r_1} \bmod p$ where $g^{r_1} \bmod p$ is the answer of H_1-query; then since $\mathcal{S}$ does not know B'private key, it will chooses random $l \in \mathbb{Z}_q$ and sets $s = l$ and $b = y_B g^{-l} y_1^{-e} \bmod p = y_B g^{-l} y_1^{-r_2} \bmod p$ where r_2 is the answer of H_2 query; finally, $\mathcal{S}$ returns (b, c, s) as the response.
  - Case (2): If role=nominator, $\mathcal{S}$ simulates the behavior of a nominee and interacts with $\mathcal{F}_{II}$ according to **Signing** protocol except the following subcase: if $y_2 = y_B$, similar to the subcase 2 in case (1).
  - Case (3): If role=nominee, $\mathcal{S}$ simulates the behavior of a nominator and interacts with $\mathcal{F}_{II}$ according to **Signing** protocol except the following subcase: if $y_1 = y_B$, similar to the subcase 1 in case (1).

- Confirmation/Disavowal Oracle:When $\mathcal{F}_{II}$ makes a confirmation/disavowal query on (m, σ, y_1, y_2), $\mathcal{S}$ simulates **Confirmation/Disavowal** protocol accordingly except the following case: if $y_2 = y_B$, i.e. B is indicated as the nominee, $\mathcal{S}$ does not know B's private key to prove a DH-tuple/non-DH-tuple $(g, y_B, H_1(m\|y_1\|y_2), c)$. In this situation, if both $y_1 = y_A$ and $m = \widetilde{m}$ are satisfied, $\mathcal{S}$ aborts; otherwise, $\mathcal{S}$ will use its knowledge r_1 to execute the WI protocol, where $g^{r_1} \bmod p$ is the answer of query $H_1(m\|y_1\|y_2)$. In the following, we will see that at least $1/q_{H_1}$ chance the case that $\mathcal{S}$ aborts will not happen.

- Selectively Convert Oracle: When $\mathcal{F}_{II}$ makes a selectively convert query on (m, σ, y_1, y_2), $\mathcal{S}$ chooses randomly $c \in \mathbb{Z}_q$ and $s \in \mathbb{Z}_q$ and sets (c, s) as the answer.

After all the queries, $\mathcal{F}_{II}$ outputs a valid forgery $(m^*, \sigma^*, y_A, y_B)$ with the restrictions defined in Section 3.2. If $\widetilde{m} = m^*$, then $c^* = H_1(m^*\|y_A\|y_B)^{x_B} \bmod p = (g^{uv})^{r_1} \bmod p$. Therefore, $\mathcal{S}$ can obtain $g^{uv} = (c^*)^{1/r_1} \bmod p$ and thus solves the CDH problem.

To complete the proof, it remains to calculate the probability ϵ' that $\mathcal{S}$ does not abort and the time t' that $\mathcal{S}$ runs. The probability that $\mathcal{S}$ does not abort, i.e. the success probability that $\mathcal{S}$ guesses $\widetilde{m} = m^*$ correctly, is $q_{H_1}^{-1}$. So, the success probability that $\mathcal{S}$ solves CDH problem is $q_{H_1}^{-1}\epsilon \geq Q^{-1}\epsilon$. Note that $\mathcal{F}_{II}$ is not allowed to submit confirmation/disavowal query on (m^*, σ', y_A, y_B), Hence the simulation will not have early abortion for the case $m^* = \widetilde{m}$. The time t' of running is at most $t + Qt_q + c$ where t_q is the maximum time for simulating one oracle query and c denotes some constant time of system setup and key generation. This completes our proof. □

B. Proof of Theorem 2

Proof : Suppose there exists a (t, ϵ, Q)-distinguisher $\mathcal{D}$ who can win **Game Invisibility** with probability at least ϵ after running at most time t and making at most Q queries, then we show that there exists a (t', ϵ')-algorithm $\mathcal{D}'$ who can solve the DDH problem by running $\mathcal{D}$ as a subroutine. Let $(g, U = g^u, V = g^v, Z = g^z)$ be a random instance of the DDH problem where g, g^u, g^v, $g^z \in \mathbb{Z}_p^*$, $\mathcal{D}'$ will simulate all the oracles and answer $\mathcal{D}$'s queries as in Lemma 2.

After all the queries, $\mathcal{D}$ submits the challenging message m^*. We assume that $\mathcal{D}$ has already made the corresponding H_1 query on (m^*, y_A, y_B) and H_2 query on $(y_A\|y_B\|b^*\|c^*\|m^*)$, but it has never submitted a Corrupt Oracle on y_B, $(m^*, y_A, y_B, role)$ has never been queried to Signing Oracle for any valid value of $role$. If $m^* = \widetilde{m}$, then $H_1(m^*\|y_A\|y_B) = (g^v)^{r_1} \bmod p$, $\mathcal{D}'$ returns the challenging signature $\sigma^* = (b^*, c^*, s^*)$ where $c^* = Z^{r_1} \bmod p$. Otherwise, $\mathcal{D}'$ aborts and fails to solves DDH problem.

After receiving the challenging signature σ^*, $\mathcal{D}$ can still submit queries to all the oracles with the restrictions defined in Section 3.3. Finally, $\mathcal{D}$ submits his guess b' to $\mathcal{D}'$. $\mathcal{D}'$ forwards b' as his answer to the DDH problem. Note

that if $b' = 1$, then σ^* is a valid signature of message m^* with probability $1/2+\epsilon$, which means $c^* = (g^{uv})^{r_1} \bmod p$. Since $\mathcal{D}'$ computes c^* as $Z^{r_1} \bmod p$, hence $z = uv \bmod q$. Otherwise, σ^* is a invalid signature of message and $z \neq uv \bmod q$. Therefore, if $\mathcal{D}'$ does not abort during the simulation, it can solve the instance of DDH problem with the advantage at least ϵ.

To complete the proof, it remains to calculate the probability ϵ' that $\mathcal{D}'$ does not abort and the time t' that $\mathcal{D}'$ runs. The probability that $\mathcal{S}$ does not abort, i.e. the success probability that $\mathcal{D}'$ guesses $\widetilde{m} = m^*$ correctly, is $q_{H_1}^{-1}$. So, the success probability that $\mathcal{S}$ solves DDH problem is $q_{H_1}^{-1}\epsilon \geq Q^{-1}\epsilon$. Note that $\mathcal{D}$ is not allowed to submit confirmation/disavowal query on (m^*, σ', y_A, y_B), hence the simulation will not have early abortion for the case $m^* = \widetilde{m}$. The time t' of running is at most $t + Qt_q + c$ where t_q is the maximum time for simulating one oracle query and c denotes some constant time of system setup and key generation. This completes our proof. □

C. Proof of Theorem 3

The proof of this theorem consists of the following two lemmas:

Non-impersonation of Confirmation/Disavowal Protocols:

Lemma 3 *The modified Huang-Wang's convertible nominative signature scheme is secure against impersonation of confirmation/disavowal protocol if the DLP problem is hard.*

Proof : Suppose there exists a (t, ϵ, Q)-impersonator $\mathcal{I}_I$ who can win **Game Impersonation of Confirmation/Disavowal protocol** with probability at least ϵ after running at most time t and making at most Q queries, then we show that there exists a (t', ϵ')-algorithm $\mathcal{S}$ who can solve the DLP problem by running $\mathcal{I}_I$ as a subroutine.

Let $(g, U = g^u)$ be a random instance of the DLP problem where g, $g^u \in \mathbb{Z}_p^*$. $\mathcal{S}$ first generate cp according to **System Setup** algorithm and sets nominator B's public key $y_B = U$. A's public/private key pair (y_A, x_A) is generated using **Key Generation** algorithm accordingly. $\mathcal{S}$ will simulate all the oracles and answer $\mathcal{I}_I$'s queries similarly to the simulator in the proof of Lemma 2 with the exception that $\mathcal{S}$ will always return $g^r \bmod p$ as the answer for any H_1-query.

Based on the proof techniques in [6], the advantage that $\mathcal{S}$ can extract the discrete logarithm of y_B to the base g, i.e. $x_B = u$, is at least $\epsilon' = (\epsilon - \frac{1}{q})^2/2$. The time t' of running is at most $t + Qt_q + c$ where t_q is the maximum time for simulating one oracle query and c denotes some constant time of system setup, key generation and the impersonation of **Confirmation/Disavowal** protocol which $\mathcal{I}_I$ executes with $\mathcal{S}$. This completes our proof. □

Non-impersonation of Selectively Convert Algorithm:

Lemma 4 *The modified Huang-Wang's convertible nominative signature scheme is secure against impersonation of selectively convert algorithm if the DLP problem is hard.*

Proof : Suppose there exists a (t, ϵ, Q)-impersonator $\mathcal{I}_{II}$ who can win **Game Impersonation of Selectively Convert Algorithm** with probability at least ϵ after running at most time t and making at most Q queries, then we show that there exists a (t', ϵ')-algorithm $\mathcal{S}$ who can solve the DLP problem by running $\mathcal{I}_{II}$ as a subroutine.

Let $(g, U = g^u)$ be a random instance of the DLP problem where g, $g^u \in \mathbb{Z}_p^*$, $\mathcal{S}$ will simulate all the oracles and answer $\mathcal{I}_{II}$'s queries as in Lemma 2 with the exception that $\mathcal{S}$ will always return $g^r \bmod p$ as the answer for any H_1-query.

After all the queries, $\mathcal{I}_{II}$ outputs a valid forgery $P_{y_A,y_B}^{m^*,\sigma^*}=(c_1^*, s_1^*)$ on $(m^*, \sigma^*, y_A, y_B)$ with the restrictions defined in Section 3.4. It is obvious that (c_1^*, s_1^*) satisfies $c_1^* = H_2(\, g \parallel h \parallel y_B \parallel c^* \parallel g^{s_1^*} y_B^{c_1^*} \parallel h^{s_1^*} c^{*c_1^*} \parallel \sigma^*) \bmod q$ where $h = H_1(m^* \| y_A \| y_B)$. Using the forking lemma [10], $\mathcal{I}_{II}$ can output another forgery $P_{y_A,y_B}^{m^*,\sigma^*} = (c_2^*, s_2^*)$ on $(m^*, \sigma^*, y_A, y_B)$. We have

$$c_1^* \neq c_2^* \pmod q$$

$$s_1^* + x_B c_1^* = s_2^* + x_B c_2^* \pmod q$$

From the above equations, $\mathcal{S}$ can obtain $x_B = (s_2^* - s_1^*)/(c_1^* - c_2^*) \bmod q$ and thus solves the DLP problem.

To complete the proof, it remains to calculate the success probability ϵ' of $\mathcal{S}$ and the time t' that $\mathcal{S}$ runs. The success probability ϵ' of $\mathcal{S}$ is at least ϵ^2 due to the forking lemma [10]. The time t' of running is at most $2t + 2Qt_q + c$ where t_q is the maximum time for simulating one oracle query and c denotes some constant time of system setup and key generation. □

D. Proof of Theorem 4

Proof : This secure property follows directly the soundness property of the WI proofs in [6]. □