An Efficient One-move Nominative Signature Scheme

Dennis Y. W. Liu, Qiong Huang, and Duncan S. Wong

Department of Computer Science
City University of Hong Kong
Hong Kong, China
dliu@cs.cityu.edu.hk, {csqhuang,duncan}@cityu.edu.hk

Abstract. A signer in a Nominative Signature (NS) scheme can arbitrarily choose a nominee, then jointly generate a signature in such a way that the signature can only be verified with the nominee's consent. NS is particularly useful in user certification systems. Currently, the only secure NS scheme available requires multi-round communications between the nominator and the nominee during signature generation. This implies that an NS-based user certification system requires a certification issuer to interact with a user using a complicated multi-round protocol for certificate issuance. It remains an open problem to construct an efficient and non-interactive NS scheme. In this paper, we solve this problem by proposing the first efficient one-move (i.e. non-interactive) NS scheme. In addition, we propose an enhanced security requirement called Strong Invisibility, and prove that our scheme satisfies this strong security requirement.

1 Introduction

A nominative signature (NS) scheme [11, 9, 16, 8, 13] allows a signer A called *nominator* to work jointly with a *nominee* B to generate a signature σ on a message m such that the validity of σ can only be verified by B. In addition, only B can convince a (third-party) *verifier* C the validity of σ.

Although the notion of NS has been introduced for over a decade [11], it was not until recently that the notion has finally been formalized [13]. In the past, besides lacking a formal definition, the application of NS has also been questioned. [13] gave the first convincing application for NS schemes — user certification systems. A user certification system has conventionally believed to be best built using a Universal Designated Verifier Signature (UDVS), which was introduced by [15]. [13] showed that an NS-based user certification system has several important advantages over a system built using a UDVS scheme.

A user certification system [15] concerns about letting a user B convince a (third-party) verifier C the validity of B's birth certificate, driving licence, academic transcripts or other documents, that are issued by an authority A, but not allowing C to further disseminate the validity information of any of B's certificates without B's consent.

In a UDVS-based user certification system, A is the signer of the UDVS scheme and a certificate s is a standard publicly verifiable signature. However, A has to be fully trusted by B (the signature holder in UDVS). If A is malicious, there are two attacks which will conflict with B's interest. First, A may maliciously reveal s to the public, and since s is publicly verifiable, 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 s and the public keys of A and C. 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 certificate in the application of user certification systems). Also, σ is not publicly verifiable. Note that A can still issue standard signatures on m or nominative signatures on m jointly with other nominees. But these events will just show that A is dishonest. To illustrate more clearly, let us consider the following a practical scenario.

Suppose in a hospital, a patient's medical records have to be certified and signed by a hospital authority. Due to the privacy of the patient, the patient does not want anybody to disseminate his/her medical records. That means, the patient wants to have full control on who can verify the validity of his/her medical records. NS plays an important role here. The hospital authority is acting as the nominator and the patient is acting as the nominee. The hospital authority and the nominee jointly create a nominative signature on some medical record. Some may notice that the hospital authority can simply release the medical records without participating in the nominative signature generation, but the patient can accuse the hospital of forging such a medical record. The role of NS in this scenario is to produce a mutual agreement on the validity of the medical records - without the hospital's authority, the professional validity of the medical records cannot be ensured, while without the patient's agreement, the hospital cannot forge any medical record of the patient.

Our Contributions. The only secure NS scheme available is due to [13]. Their construction is essentially built on Chaum-van Antwerpen's undeniable signature (US) [5, 6]. It requires at least four rounds of communications between the nominator and the nominee for signature generation. This implies that an NS-based user certification system requires a certification issuer to interact with a user using a complicated multi-round protocol for certificate issuance. It remains an open problem to construct an efficient non-interactive NS scheme.

In this paper, we solve this problem by proposing the first efficient non-interactive NS scheme. During the signature generation, the nominator only needs to send one message to the nominee. No interaction between the nominator and the nominee is required and the signature generation incurs simply a one-move message transfer from the nominator to the nominee. We show that our construction has much better performance than that of [13] in both network efficiency and computational complexity.

We further enhance the security model proposed in [13] and define a stronger, more realistic security requirement called **Strong Invisibility**. Strong Invisibility requires that a nominator cannot tell whether a nominative signature is valid, even by recalling the entire signature generation transcript and all the intermediate values and states of the signature generation operations that the nominator has carried out previously.

Paper Organization. In Sec. 2, related work of NS is reviewed. This is followed by the definition of NS in Sec. 3. Number-theoretic assumptions which are to be used in the security analysis of our proposed NS scheme are given in Sec. 4. Our NS scheme is then described in Sec. 5 and its security is analyzed in Sec. 6. Finally, the paper is concluded in Sec. 7.

2 Related Work

Since the introduction of NS [11], it has been considered as a dual scheme of undeniable signature (US) [5, 6]. Both NS and US are *non-self-authenticating*, namely, the public is not able to determine if a signature-message pair is valid or not merely from the signature itself. For US, it can only be verified with the aid of the signer, while for NS, it can only be verified with the aid of the nominee, rather than the nominator.

Nominative signature is also related to designated verifier signature (DVS) [10] and designated confirmer signature (DCS) [4]. But also, it has some significant distinctions from DVS and DCS. For DVS and DCS, a signature is always self-verifiable, namely, the signer can always determine if a signature is valid or not. For NS, however, the signer (i.e. the nominator) is not able to determine the validity of a nominative signature. DVS does not have the notion of proof delegation. The signer of a DVS specifies a designated verifier who can determine the validity of a signature. However, the verifier is not able to convince other parties on the validity of the signature. For NS, the signer nominates a party who can later convince other parties on the validity of a signature. When comparing between DCS and NS, we can see that for DCS, the signer can also play the role of prover for convincing other parties on the validity of a signature. For NS, however, it is mandatory that the signer is not able to perform the role of a prover. In other words, DCS allows a signer to request for a helper so that both of them can be the provers. NS requires a signer to nominate a prover so that the power of proving the validity of a signature is transferred from the signer to the nominee.

The notion and construction of nominative signature were first proposed by [11]. However, their construction was later found flawed by [9]. In the construction of [11], the nominator can always determine the validity of a nominative signature. [9] proposed the notion of convertible nominative signature, aiming at allowing only the nominee to convert a signature to a publicly-verifiable one. They also proposed a new scheme. [16] described an attack against this new scheme. However [8] showed that the attack of [16] was incomplete and inaccurate. Nevertheless, [8] described a new attack, which allows the nominator of Huang and Wang's scheme to generate valid signatures on his own and show the validity of the signature to anyone without the consent of the nominee.

In [9], a definition and some requirements for nominative signature were also proposed. However, their definition of nominative signature does not match with the scheme they proposed and the set of security requirements specified are incomplete and informal. [13] proposed the first set of formal definitions and security models for NS was proposed. However, the Invisibility requirement does not capture the stronger notion, namely Strong Invisibility (defined in Sec. 3 of this paper). The NS construction of [13] requires multi-round communications between the nominator and the nominee for signature generation. It is currently unknown if a one-move, non-interactive NS scheme can be built. In this paper, we answer this question positively by proposing the first efficient NS scheme which requires only one-move communication. In addition, our scheme is also proven secure under the enhanced set of security requirements.

3 Definitions and Adversarial Models

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

1. SystemSetup: 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: On input param, it generates a public/private key pair (pk, sk).
3. $\mathsf{Ver}^{\mathsf{nominee}}$: On input a message $m \in \{0,1\}^*$, a nominative signature σ, a public key pk_A and a private key sk_B, it returns valid or invalid.

An NS scheme proceeds as follows. SystemSetup is first invoked for generating param. KeyGen is then executed to initialize each entity that is to be involved in the subsequent part of the scheme. One entity is called nominator. We denote it by A. Let (pk_A, sk_A) be the public/private key pair of A. Let B be the nominee that A nominates. Let B's public/private key pair be (pk_B, sk_B).

To generate a nominative signature σ, A chooses a message m, and carries out SigGen protocol below with B. Formally, the SigGen protocol is carried out between two interactive PPT algorithms in k: nominator A (who is holding (pk_A, sk_A)) and nominee B (who is holding (pk_B, sk_B)).

> SigGen Protocol: The common inputs of A and B are param and 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, B either outputs a nominative signature σ or $\perp$ indicating the failure of the protocol run.

If the protocol consists of only one move of message transfer, then it is non-interactive.

Signature Space: This is determined by pk_A and pk_B. We emphasize that the signature space has to be specified explicitly in each actual NS scheme.

For a nominative signature σ in the signature space, the validity of σ can be determined by B using $\mathsf{Ver}^{\mathsf{nominee}}$. To convince a third party C on the validity or invalidity of σ, B as a prover and C as a verifier carries out the following Confirmation or Disavowal protocol.

> Confirmation/Disavowal Protocol: On input (m, σ, pk_A, pk_B), B sets a bit 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 all the algorithms and protocols are carried out according to the scheme specification, 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.

We now formalize the security games for NS. We begin with the oracles.

- CreateUser: On input an identity, say C, it generates a key pair (pk_C, sk_C) using KeyGen and returns pk_C.
- Corrupt: On input a public key pk, if pk is generated by CreateUser or in $\{pk_A, pk_B\}$, the corresponding private key is returned; otherwise, $\perp$ is returned. pk is said to be *corrupted*.
- SignTranscript: On input a message m, two distinct public keys, pk_1 (the nominator) and pk_2 (the nominee) such that at least one of them is uncorrupted, and one parameter called $role \in \{\mathsf{nil}, \mathsf{nominator}, \mathsf{nominee}\}$,
  - if $role = \mathsf{nil}$, $\mathcal{S}$ simulates a run of SigGen and returns $(\sigma, trans_\sigma)$ where σ is a valid nominative signature and $trans_\sigma$ is the transcript of the execution of SigGen. σ is said to be *valid* on m with respect to pk_1 and pk_2 if $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_1, sk_2)$ where sk_2 is the corresponding private key of pk_2.
  - if $role = \mathsf{nominator}$, $\mathcal{S}$ (as nominee with public key pk_2) simulates a run of SigGen with $\mathcal{F}$ (as nominator with pk_1);
  - if $role = \mathsf{nominee}$, $\mathcal{S}$ (as nominator with pk_1) simulates a run of SigGen with $\mathcal{F}$ (as nominee with public key pk_2).
- Confirmation/disavowal: On input a message m, a nominative signature σ and two public keys pk_1 (nominator), pk_2 (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, the oracle returns a bit $\mu = 1$ and a transcript of the Confirmation protocol. Otherwise, $\mu = 0$ and a transcript of the Disavowal protocol are returned.
  - In an active/concurrent attack, the oracle checks if m is valid as in the passive attack. If so, the oracle returns $\mu = 1$ and executes the Confirmation protocol with $\mathcal{F}$ (acting as a verifier). Otherwise, the oracle returns $\mu = 0$ and executes the Disavowal protocol with $\mathcal{F}$. The difference between active and concurrent attack is that $\mathcal{F}$ interacts serially with the oracle in the active attack while $\mathcal{F}$ interacts with different instances of the oracle concurrently in the concurrent attack.

A secure NS scheme should satisfy the following security requirements: **Unforgeability**, **Strong Invisibility** and **Impersonation**. The first and the last requirements are from [13]. The second one is new, and is a stronger notion when comparing with the **Invisibility** requirement of [13].

3.1 Unforgeability

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. $\mathcal{F}$ is invoked with inputs 1^k, pk_A and pk_B.
2. (*Attacking Phase*) $\mathcal{F}$ can make queries to the oracles mentioned above.
3. (*Output Phase*) $\mathcal{F}$ outputs a pair (m^*, σ^*).

$\mathcal{F}$ *wins* the game if $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m^*, \sigma^*, pk_A, sk_B)$ and (1) $\mathcal{F}$ has never corrupted *both* sk_A and sk_B; (2) $(m^*, pk_A, pk_B, role)$ has never been queried to SignTranscript for any valid value of $role$; (3) $(m^*, \sigma, pk_A, pk_B)$ has never been queried to Confirmation/disavowal for any nominative signature σ with respect to pk_A and pk_B (check *Signature Space* on page 4). $\mathcal{F}$'s advantage in this game is defined to be the probability that $\mathcal{F}$ wins.

Definition 1. *An NS is unforgeable if no PPT forger $\mathcal{F}$ has a non-negligible advantage in Game Unforgeability.*

3.2 Strong Invisibility

In [13], the notion *Invisibility* was defined. It essentially requires that given a nominative signature, except the nominee, no one, including the nominator, is able to determine the validity of the signature.

This requirement is not realistic enough. Note that for the nominator A, it has some additional information which may help itself to determine the signature's validity. For example, A may keep all the intermediate values, internal states and transcripts of the signature generation operations, for the purpose of determining the validity of a nominative signature in which A is the *claimed* nominator. In the Invisibility model of [13], the adversary is not able to access these information. Consider the adversary to be the nominator, the original Invisibility model becomes unrealistic.

We propose a stronger notion that prevents A from determining the validity of a nominative signature by "memorizing" all the transcripts that A has involved in previous runs of the SigGen protocol.

Game Strong Invisibility: The initialization phase is the same as that of Game Unforgeability. In the game, the adversary is a distinguisher $\mathcal{D}$ who can access all the oracles described above.

1. (*Challenge Signature Generation Phase*) At some point in the game, $\mathcal{D}$ sends a message m^* to the simulator while acting as a nominator for carrying out a protocol run of SignGen with the simulator which acts as the nominee. Let σ^{valid} be the nominative signature generated by the simulator at the end of the protocol run. Note that $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m^*, \sigma^{valid}, pk_A, sk_B)$.
   The challenge signature σ^* is then generated by the simulator based on the outcome of a random coin toss b. If $b = 1$, then set $\sigma^* = \sigma^{valid}$. If $b = 0$, then σ^* is chosen uniformly at random from the signature space of the nominative signature scheme with respect to pk_A and pk_B.
2. (*Guess Phase*) $\mathcal{D}$ continues querying the oracles, until it outputs a guess b'.

$\mathcal{D}$ *wins* the game if $b' = b$ and it does not violate any of the following restrictions:

1. $\mathcal{D}$ has never corrupted sk_B using oracle Corrupt (but $\mathcal{D}$ may have corrupted sk_A);
2. $(m^*, pk_A, pk_B, role)$ has never been queried to SignTranscript, for any value of $role$;
3. $(m^*, \sigma, pk_A, pk_B)$ has never been queried to Confirmation/disavowal for any nominative signature σ with respect to pk_A and pk_B.

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

Definition 2. *An NS has the property of strong invisibility if no PPT distinguisher $\mathcal{D}$ has a non-negligible advantage in Game Strong Invisibility.*

Remark: In the attacking phase of Game Invisibility defined in [13], $\mathcal{D}$ outputs a message m^* and requests for a challenge nominative signature σ^* on m^*. $\mathcal{D}$ is not allowed to interact with the nominee when generating σ^*. In Game Strong Invisibility, $\mathcal{D}$ as the nominator can actually interacts with the nominee when generating σ^*. This additional information makes Game Strong Invisibility stronger than Game Invisibility.

3.3 Impersonation

The validity of a nominative signature can only be determined with the aid of the nominee. Even the nominator should not be able to show the validity of a nominative signature. We consider the following game against an impersonator $\mathcal{I}$. It is similar to that in [13].

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. $\mathcal{I}$ is permitted to issue queries to all the oracles described above. $\mathcal{I}$ prepares a triple (m^*, σ^*, μ) where m^* is some message, σ^* is a nominative signature (i.e. σ^* is in the signature space of the underlying signature scheme 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.

The impersonator $\mathcal{I}$ *wins* the game if the simulator acting as the verifier outputs accept while $\mathcal{I}$ has never corrupted sk_B (but $\mathcal{I}$ may have corrupted sk_A using Corrupt). $\mathcal{I}$'s advantage is defined to be the probability that $\mathcal{I}$ wins.

Definition 3. *An NS is said to be secure against impersonation if there is no PPT impersonator $\mathcal{I}$ who has a non-negligible advantage in Game Impersonation.*

4 Preliminaries and Number-theoretic Assumptions

Let G, G_1 be cyclic groups of prime order p. Let g be the generator of G. Let $e : G \times G \rightarrow G_1$ be an efficiently computable map with the following properties. Bilinear: for all $a, b \in \mathbb{Z}$, $e(g^a, g^b) = e(g, g)^{ab}$; and Non-degenerate: $e(g, g) \neq 1$. We refer readers to [3] for more information on bilinear pairings.

The security of our NS construction, proposed in the next section, relies on several new number-theoretic assumptions. In this section, we show that all the computational assumptions that our scheme relies on are reducible to the l-BDHE assumption [2]. We also justify a new decisional assumption under the generic group model [14].

Bilinear Diffie-Hellman Exponent (BDHE) Problem. First proposed by Boneh, Boyen and Goh in [2], the computational l-BDHE problem is defined as follows. Given g, h and $y_i = g^{\alpha^i}$ in G for $i = 1, 2, \cdots, l-1, l+1, \cdots, 2l$, compute $e(g, h)^{(\alpha^l)} \in G_1$. An algorithm $\mathcal{A}_1$ has advantage ϵ in solving computational l-BDHE problem if

$$Pr[\mathcal{A}_1(g, h, y_1, \cdots, y_{l-1}, y_{l+1}, \cdots, y_{2l}) = e(g, h)^{(\alpha^l)}] \geq \epsilon$$

where the probability is over the random choices of $g, h \in G$, $\alpha \in \mathbb{Z}_p$, and the random bits consumed by $\mathcal{A}_1$.

Weak Computational Diffie-Hellman I (WCDH-I) Problem. Given $g, g^a, g^{a^2}, g^b \in G$, compute g^{ab}. An algorithm $\mathcal{A}_3$ has advantage ϵ in solving WCDH-I in G if

$$Pr[\mathcal{A}_3(g, g^a, g^{a^2}, g^b) = g^{ab}] \geq \epsilon$$

where the probability is over the random choices of $g \in G$, $a, b \in \mathbb{Z}_p$, and the random bits used by $\mathcal{A}_3$. The following theorem shows that computational l-BDHE assumption implies WCDH-I assumption.

Theorem 1. *If there exists a t-time algorithm $\mathcal{A}_3$ that has advantage ϵ in solving WCDH-I, then there exists a poly(t)-time algorithm $\mathcal{A}_1$ that has advantage ϵ in solving computational l-BDHE problem for any $l > 2$, where poly(t) is some polynomial in t.*

Proof. Let $(g, h, g^{\alpha}, g^{\alpha^2}, \cdots, g^{\alpha^{l-1}}, g^{\alpha^{l+1}}, \cdots, g^{\alpha^{2l}})$ be a given computational l-BDHE problem instance. $\mathcal{A}_1$ runs $\mathcal{A}_3$ with input $(g, g^{\alpha}, g^{\alpha^2}, g^{\alpha^{l-1}\omega})$ where $\omega \in_R \mathbb{Z}_p$. If $\mathcal{A}_3$ succeeds in solving WCDH-I, its output will be $g^{\alpha^l\omega}$. Hence $\mathcal{A}_1$ can obtain $g^{\alpha^l} = (g^{\alpha^l\omega})^{1/\omega}$ and output $e(g^{(\alpha^l)}, h) = e(g, h)^{(\alpha^l)}$, which is the solution to the l-BDHE problem instance.

Weak Computational Diffie-Hellman II (WCDH-II) Problem. Given $g, g^a, g^{a^2}, g^b \in G$, compute g^{a^2b}. An algorithm $\mathcal{A}_4$ has advantage ϵ in solving WCDH-II in G if

$$Pr[\mathcal{A}_4(g, g^a, g^{a^2}, g^b) = g^{a^2b}] \geq \epsilon$$

where the probability is over the random choices of $g \in G$, $a, b \in \mathbb{Z}_p$ and the random bits consumed by $\mathcal{A}_4$.

Theorem 2. *If there exists a t-time algorithm $\mathcal{A}_4$ that has advantage ϵ in solving WCDH-II, then there exists a poly(t)-time algorithm $\mathcal{A}_1$ that has advantage ϵ in solving computational l-BDHE problem for any $l > 2$, where poly(t) is some polynomial in t.*

Proof. Let $(g, h, g^{\alpha}, g^{\alpha^2}, \cdots, g^{\alpha^{l-1}}, g^{\alpha^{l+1}}, \cdots, g^{\alpha^{2l}})$ be a given computational l-BDHE problem instance. $\mathcal{A}_1$ runs $\mathcal{A}_4$ with input $(g, g^{\alpha}, g^{\alpha^2}, g^{\alpha^{l-2}\omega})$ where $\omega \in_R \mathbb{Z}_p$. If $\mathcal{A}_4$ succeeds in solving WCDH-II, its output will be $g^{\alpha^l\omega}$. Hence $\mathcal{A}_1$ can obtain $g^{\alpha^l} = (g^{\alpha^l\omega})^{1/\omega}$, and output $e(g^{(\alpha^l)}, h) = e(g, h)^{(\alpha^l)}$, which is the solution to the l-BDHE problem instance.

Weak Discrete Logarithm (WDLOG) Problem. Given g, g^a, $g^{a^2} \in G$, compute a. An algorithm $\mathcal{S}$ has advantage ϵ in solving WDLOG in G if

$$Pr[\mathcal{S}(g, g^a, g^{a^2}) = a] \geq \epsilon$$

where the probability is over the random choices of $g \in G$, $a \in \mathbb{Z}_p$ and the random bits consumed by $\mathcal{S}$.

Theorem 3. *If there exists a t-time algorithm $\mathcal{S}$ that has advantage ϵ in solving WDLOG, then there exists a poly(t)-time algorithm $\mathcal{A}_1$ that has advantage ϵ in solving computational l-BDHE problem for any $l > 2$, where poly(t) is some polynomial in t.*

Proof. Let $(g, h, g^{\alpha}, g^{\alpha^2}, \cdots, g^{\alpha^{l-1}}, g^{\alpha^{l+1}}, \cdots, g^{\alpha^{2l}})$ be a given computational l-BDHE problem instance. $\mathcal{A}_1$ runs $\mathcal{S}$ on $(g, g^{\alpha}, g^{\alpha^2})$. If $\mathcal{S}$ succeeds in solving WDLOG, it outputs α. Hence $\mathcal{A}_1$ can compute $e(g,h)^{(\alpha^l)}$, which is the solution to the l-BDHE problem instance.

Weak Decisional Diffie-Hellman (WDDH) Problem. Given $g, g^a, g^{a^2}, g^b, g^c, g^{ac}, g^d \in G$, decide if $d \equiv a^2bc \pmod{p}$. An algorithm $\mathcal{D}$ for the WDDH problem should output 1 if $d \equiv a^2bc \pmod{p}$; otherwise, output 0. $\mathcal{D}$ has advantage ϵ in solving WDDH in G_1 if

$$|Pr[\mathcal{B}(g, g^a, g^{a^2}, g^b, g^c, g^{ac}, g^{a^2bc}) = 0] - Pr[\mathcal{B}(g, g^a, g^{a^2}, g^b, g^c, g^{ac}, R) = 0]| \geq \epsilon$$

where $R \in_R G$. The probability is over the random choices of $g, R \in G$, $a, b, c \in \mathbb{Z}_p$ and the random bits used by $\mathcal{D}$.

Theorem 4. *Let $\mathcal{D}$ be an algorithm that solves the WDDH problem in the generic group model [14], making at most q queries to the oracles computing the group actions in G, G_1. Let $\xi : \mathbb{Z}_p \to \{0,1\}^*$ be an injective function which maps all $x \in \mathbb{Z}_p$ to the string representation $\xi(g^x)$ of $g^x \in G$. Suppose $a, b, c, r \in_R \mathbb{Z}_p$, ξ are chosen at random and d is a random bit. Set $\omega_0 = a^2bc$ and $\omega_1 = r$. Let $T_0 = g^{a^2bc}$ and $T_1 = g^r$. The probability ϵ that $D(\xi; 1, a, b, c, \omega_d, \omega_{1-d}) = d$ is bounded by $\frac{1}{2} + O(q^2/p)$.*

Proof. A simulator S plays the following game with $\mathcal{D}$. S maintains two lists $L_1 = \{(F_{1,i}, \xi_{1,i}) : i = 0, \cdots, k_1 - 1\}$ and $L_2 = \{(F_{2,i}, \xi_{2,i}) : i = 0, \cdots, k_2 - 1\}$, where $F_{*,*} \in Z_p[A, B, C, T_0, T_1]$ are polynomials in the indeterminates A, B, C, T_0, T_1 with coefficients in $\mathbb{Z}_p$. Set $F_{1,0} = 1$, $F_{1,1} = A$, $F_{1,2} = B$, $F_{1,3} = C$, $F_{1,4} = T_0$, $F_{1,5} = T_1$. The corresponding strings are set to arbitrary distinct strings in $\{0,1\}^*$ and given to $\mathcal{D}$. S responds to $\mathcal{D}$'s queries as follows:

Group Action. Given a multiply/divide selection bit and two indexes i and j with $0 \leq i, j < k_1$, compute $F_{1,k_1} \leftarrow F_{1,i} \pm F_{1,j}$ depending on whether a multiplication or a division is requested. If $F_{1,k_1} = F_{1,l}$ for some l with $0 \leq l < k_1$, S sets $\xi_{1,k_1} = \xi_{1,l}$; otherwise, S sets ξ_{1,k_1} to a string in $\{0,1\}^*$ distinct from $\xi_{1,0}, \cdots, \xi_{1,k_1-1}$. S appends new values F_{1,k_1}, ξ_{1,k_1} to L_1 and gives ξ_{1,k_1} to D. k_1 is incremented by 1. Group action queries in G_1 are treated similarly.

Pairing. Given two indexes i and j with $0 \leq i, j < k_2$, compute the product $F_{2,k_2} \leftarrow F_{1,i}F_{1,j}$. If $F_{2,k_2} = F_{2,l}$ for some l with $0 \leq l < k_2$, S sets $\xi_{2,k_2} = \xi_{2,l}$; otherwise, S sets ξ_{2,k_2} to a string in $\{0,1\}^*$ distinct from $\xi_{2,0}, \cdots, \xi_{2,k_2-1}$. S appends new values F_{2,k_2}, ξ_{2,k_2} to L_2 and gives ξ_{2,k_2} to D. k_2 is incremented by 1.

When $\mathcal{D}$ terminates, S chooses $a, b, c, r \in \mathbb{Z}_p$ randomly. $\mathcal{D}$ wins the game if

1. for some i, j where $i \neq j$, we have $F_{1,i}(a, b, c, a^2bc, r) = F_{1,j}(a, b, c, a^2bc, r)$, or $F_{1,i}(a, b, c, r, a^2bc) = F_{1,j}(a, b, c, r, a^2bc)$; and
2. for some i, j where $i \neq j$, we have $F_{2,i}(a, b, c, a^2bc, r) = F_{2,j}(a, b, c, a^2bc, r)$, or $F_{2,i}(a, b, c, r, a^2bc) = F_{2,j}(a, b, c, r, a^2bc)$.

Since $deg(F_{1,i}) \leq 1$ and $deg(F_{2,i}) \leq 2$, according to Lemma 1 in [Shoup97], for fixed i, j, let $F_\alpha = F_{\alpha,i} - F_{\alpha,j}$ where $\alpha \in \{1, 2\}$, the probabilities that the above two cases holds are 1/p and 2/p respectively. Total number of pairs of possible i, j is $\binom{k_1}{2} + \binom{k_2}{2} = \frac{{k_1}^2 - k_1}{2} + \frac{{k_2}^2 - k_2}{2}$. Since $q + 6 \geq k_1 + k_2$, it follows that the success probability that $\mathcal{D}$ wins the above game is bounded by $\frac{1}{2} + O(q^2/p)$.

5 Our Construction

We now propose a bilinear-pairing-based NS construction. The construction requires only one-move communication from the nominator A to the nominee B in the SigGen protocol.

SystemSetup: Let $k \in \mathbb{N}$ be a system parameter. The algorithm generates two cyclic groups G, G_1 of prime order $p \geq 2^k$, a generator g of G and a bilinear map $e : G \times G \rightarrow G_1$ with properties described in Sec. 4. It also specifies a hash function $H : \{0,1\}^* \rightarrow G$. Let $\mathsf{param} = (p, G, G_1, g, H)$.

KeyGen: On input param, it generates (y, x) where $x \in_R \mathbb{Z}_p$ and $y = g^x$. We use y_A to denote nominator A's public key and x_A to denote A's private key. Similarly, let (y_B, x_B) be the public/private key pair of nominee B.

SigGen Protocol: Let $m \in \{0,1\}^*$ be a message. A and B carry out the following.

1. A computes $s = H(m \| y_A \| y_B)^{x_A}$ and sends (m, s) to B.
2. B checks if $e(y_A, H(m \| y_A \| y_B)) \stackrel{?}{=} e(s, g)$. If not, B outputs $\perp$ for failure; otherwise, B chooses $r \in_R \mathbb{Z}_p$ and computes a nominative signature σ as $(\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ where $\sigma_1 = s^{x_B^2 r}$, $\sigma_2 = y_A^r$, $\sigma_3 = y_B^r$ and $\sigma_4 = g^r$.

Signature Space (page 4): We say that $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ is a *nominative signature* if (1) $\sigma_1, \sigma_2, \sigma_3, \sigma_4 \in G$, (2) $e(\sigma_2, g) = e(\sigma_4, y_A)$, and (3) $e(\sigma_3, g) = e(\sigma_4, y_B)$. In order to check the validity of a nominative signature, the following algorithm is executed by nominee B.

$\mathsf{Ver}^{\mathsf{nominee}}$: On input (m, σ, y_A, x_B) where $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ is a nominative signature (i.e. in the signature space defined above), the algorithm checks if $e(\sigma_1, g) \stackrel{?}{=} e(H(m \| y_A \| y_B), \sigma_2)^{x_B^2}$. If so, output valid; otherwise, output invalid.

Confirmation/Disavowal Protocol: If $\sigma = (\sigma_1, \sigma_2, \sigma_3, \sigma_4)$ is a nominative signature, B first runs $\mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, y_A, x_B)$. If the output is valid, B sends $\mu = 1$ to a verifier C. Otherwise, B sends $\mu = 0$ to C.

$$(e(\sigma_4, g), e(H(m \| y_A \| y_B), \sigma_2), e(\sigma_3, y_B), e(\sigma_1, g))$$

For the tuple above, if $\mu = 1$, B proves to C that it is a DH-tuple; if $\mu = 0$, B proves to C that it is a non-DH-tuple.

We say that $(\mathfrak{g}, \mathfrak{g}^a, \mathfrak{g}^b, \mathfrak{g}^c) \in G_1^4$ is a DH-tuple if $c \equiv ab \pmod{p}$, where $\mathfrak{g} = e(g, g)$; otherwise, it is a non-DH-tuple. According to [12], Witness Indistinguishable (WI) [7] protocols can be used to prove/disprove a DH-tuple, that is, it is sufficient for the prover to execute the protocols successfully using its knowledge of either one of the witnesses, i.e. a or b. In the Confirmation/Disavowal protocol above, B's knowledge is x_B^2. For concrete implementation, we use the protocols due to [12].

Remark: Our technique for achieving Strong Invisibility stems from raising the signature of A, namely s, by the double square of B's private key x_B. The purpose is to thwart A from telling whether a given σ is a valid nominative signature using bilinear map.

Performance. When compared with the only secure NS scheme currently available [13], we can see that in the SigGen protocol, their scheme requires the nominator and the nominee to carry out at least four message flows (assuming that piggybacking is employed). The large number of message flows is mainly due to the requirement of proving a DH-tuple. In our construction, we eliminate the DH-tuple proof from the SigGen protocol altogether. In addition, their scheme has the nominee B as the protocol initiator, and thus requires one more message flow for A to send the final nominative signature to B. While in our scheme, the nominator A is the initiator. Hence, A can send its contribution to the final nominative signature in one single transmission.

The most time-consuming part of our scheme is the Confirmation/Disavowal Protocol. Fortunately, real-time performance can be improved by pre-computation as most of the bilinear pairing operations can be pre-computed before the protocol is carried out.

On the security of our NS scheme, it is easy to see that the construction above satisfies the correctness requirement. In the next section, we show that the construction also satisfy all the security requirements defined in Sec. 3.

6 Security Analysis

To show that our construction described above is unforgeable with respect to Def. 3.1, in the following, we claim that

1. a malicious nominee alone cannot forge a valid nominative signature (Lemma 1); and
2. a malicious nominator alone cannot forge a valid nominative signature (Lemma 2).

A nominative signature is said to be valid if $\mathsf{Ver}^{\mathsf{nominee}}$ returns valid on the corresponding inputs. By combining these two claims, we can see that the proposed nominative signature scheme is unforgeable. The following analyzes are carried out under the random oracle model [1].

Lemma 1 (Cheating Nominee). *Let $k \in \mathbb{N}$ be a security parameter. For the NS proposed above, if a (t, ϵ, Q)-nominee can forge a valid nominative signature in Game Unforgeability with probability at least ϵ after running at most time t and making at most Q queries, there exists a (t', ϵ')-adversary which can solve a WCDH-I (Weak Computational Diffie-Hellman I) problem instance (Sec. 4) with probability at least $\epsilon' = (1 - 2^{-k})Q^{-1}\epsilon$ after running at most time $t' = t + Qt_q + c$ where t_q is the maximum time for simulating one oracle query and c denotes some constant time for system setup and key generation.*

Proof. Let $\mathcal{F}$ be a (t, ϵ, Q)-forger which has nominee B's private key x_B (obtained by querying Corrupt). We show that in the random oracle model [1], $\mathcal{F}$ can be turned into a (t', ϵ')-algorithm $\mathcal{S}$ which can solve the WCDH-I problem. Let $(g, U, V, W) \in G^4$ be a random WCDH-I problem instance where $U = g^u$, $V = g^{u^2}$ and $W = g^v$. $\mathcal{S}$ has to output $Z = g^{uv}$.

Game Simulation: $\mathcal{S}$ first generates param according to SystemSetup and sets nominator A's public key $y_A = U$. B's public/private key pair (y_B, x_B) is generated using KeyGen accordingly. For a SignTranscript query on input (m, y_1, y_2), there are three cases to handle.

- Case (1): If $role =$ nil, the simulation is carried out exactly according to the SigGen protocol except in the following two sub-cases:
  - If A is indicated as the nominator (i.e. $y_1 = y_A$), $\mathcal{S}$ sets s to $U^{r'}$ where $r' \in_R \mathbb{Z}_p$ and sets the return value of the random oracle query $H(m\|y_A\|y_2)$ to $g^{r'}$.
  - If A is indicated as the nominee (i.e. $y_2 = y_A$), since $\mathcal{S}$ knows x_1, $\mathcal{S}$ sets σ_1 to $V^{r'x_1r}$, σ_2 to y_1^r, σ_3 to U^r, and σ_4 to g^r, where $r, r' \in_R \mathbb{Z}_p$ and sets the return value of the random oracle query $H(m\|y_1\|y_A)$ to $g^{r'}$. Note that $V^{r'x_1r} = s^{u^2r}$, where $s = H(m\|y_1\|y_A)^{x_1}$.
- Case (2): If $role =$ nominator, $\mathcal{S}$ simulates the behavior of a nominee and interacts with $\mathcal{F}$ according to the SigGen protocol, except the following sub-case: if A is indicated as the nominee (i.e. $y_2 = y_A$), similar to the second sub-case above, $\mathcal{S}$ sets σ_1 to $V^{r'x_1r}$ where $r' \in_R \mathbb{Z}_p$ and $g^{r'}$ is the return value of random oracle query $H(m\|y_1\|y_A)$.
- Case (3): If $role =$ nominee, $\mathcal{S}$ simulates the behavior of a nominator according to the SigGen protocol, except the following sub-case: if A is indicated as the nominator (i.e. $y_1 = y_A$), similar to the first sub-case in Case (1) above, $\mathcal{S}$ computes s as $U^{r'}$ where $r' \in_R \mathbb{Z}_p$ and $H(m\|y_A\|y_2)$ is set to $g^{r'}$.

For a Confirmation/disavowal query on (m, σ, y_1, y_2), $\mathcal{S}$ simulates the Confirmation/disavowal protocol accordingly except the following case: if A is indicated as the nominee (i.e. $y_2 = y_A$ in the query), $\mathcal{S}$ does not know A's private key component, i.e. u, to prove a DH-tuple/non-DH-tuple $(e(\sigma_4, g), e(H(m\|y_1\|y_A), \sigma_2), e(\sigma_3, y_A), e(\sigma_1, g))$. In this case, $\mathcal{S}$ uses its knowledge of (r', x_1) to execute the WI protocol, where $r' \in_R \mathbb{Z}_p$ and $g^{r'}$ is the answer of query $H(m\|y_1\|y_A)$.

Reduction: Without querying $H(m^*\|y_A\|y_B)$, due to the random oracle assumption of H, $\mathcal{F}$ has at most 2^{-k} chance to guess the value right. If $\mathcal{F}$ has queried H on $(m^*\|y_A\|y_B)$, and if $\mathcal{S}$ has guessed correctly on the forging message m^*, $\mathcal{S}$ could have set $H(m^*\|y_A\|y_B)$ to W. Note that when $H(m^*\|y_A\|y_B)$ is set to W, $\mathcal{S}$ cannot simulate Confirmation/disavowal for queries on (m^*, σ, y_A, y_B) for any nominative signature σ. This case is not going to happen due to the restriction of Game Unforgeability that the tuple (m^*, σ, y_A, y_B) cannot be queried to Confirmation/disavowal.

If $\mathcal{S}$ randomly picks a query of H as the guess of $H(m^*\|y_A\|y_B)$, the probability of guessing correctly is at least $1/Q$. $\mathcal{S}$ can solve the WCDH-I problem instance with probability at least $\epsilon' = (1 - 2^{-k})Q^{-1}\epsilon$, since $H(m^*\|y_A\|y_B)^{x_A x_B^2 r} = g^{vux_B^2 r}$ and thus Z can be computed from the forged nominative signature. The running time of $\mathcal{S}$ is at most $t' = t + Qt_q + c$.

Lemma 2 (Cheating Nominator). *Let $k \in \mathbb{N}$ be a security parameter. For the NS proposed above, if a (t, ϵ, Q)-nominator can forge a valid nominative signature in* Game Unforgeability *with probability at least ϵ after running at most time t and making at most Q queries, there exists a (t', ϵ')-adversary which can solve a WCDH-II problem instance (Sec. 4) with probability at least $\epsilon' = (1 - 2^{-k})Q^{-1}\epsilon$ after running at most time $t' = t + Qt_q + c$ where t_q is the maximum time for simulating one oracle query and c denotes some constant time for system setup and key generation.*

Proof. We show how to construct a (t', ϵ')-algorithm $\mathcal{S}$ to solve the WCDH-II problem from a (t, ϵ, Q)-forger $\mathcal{F}$ who has nominator A's private key x_A (obtained by querying Corrupt) in Game Unforgeability. Suppose (g, U, V, W) is the given random WCDH-II problem instance, where $U = g^u$, $V = g^{u^2}$ and $W = g^v$. A WCDH-II solver has to output $Z = g^{u^2 v}$.

During the simulation, $\mathcal{S}$ follows the specification of the scheme accordingly but sets the public key y_B of nominee B to U. The rest of the simulation is similar to that in the proof of Lemma 1 with some exceptions detailed as follows. For a SignTranscript query, there are three cases.

- Case (1): If $role =$ nil, SigGen is simulated accordingly except in the following two sub-cases:
  - If B is the nominator (i.e. $y_1 = y_B$), $\mathcal{S}$ sets s to $U^{r'}$ where $r' \in_R \mathbb{Z}_p$ and $H(m\|y_B\|y_2) = g^{r'}$.
  - If B is the nominee (i.e. $y_2 = y_B$), $\mathcal{S}$ sets σ_1 to $V^{r' x_1 r}$, σ_2 to y_1^r, and σ_3 to U^r, where $r, r' \in_R \mathbb{Z}_p$, and sets the return value of the random oracle query $H(m\|y_1\|y_B)$ to $g^{r'}$.
- Case (2): If $role =$ nominee, there is one special case needs to be handled. If B is the nominator, $\mathcal{S}$ sets s to $U^{r'}$ as in the first sub-case in Case (1) above.
- Case (3): If $role =$ nominator, the special case needs to be handled is when B is indicated as the nominee. In this case, $\mathcal{S}$ sets σ_1 to $V^{r' x_1 r}$ as in the second sub-case in Case (1) above.

For a Confirmation/disavowal query on (m, σ, y_1, y_2), $\mathcal{S}$ simulates the oracle as in the proof of Lemma 1. In particular, if B is the nominee, $\mathcal{S}$ does not know B's private key (i.e. u) for proving a DH-tuple/non-DH-tuple $(e(\sigma_4, g), e(H(m\|y_1\|y_B), \sigma_2), e(\sigma_3, y_B), e(\sigma_1, g))$ as described in the protocol. In this case, $\mathcal{S}$ will use its knowledge of (r', x_1) to execute the WI protocol where $H(m\|y_1\|y_B) = g^{r'}$ and $x_1 = \log_g y_1$.

Without querying $H(m^*\|y_A\|y_B)$, $\mathcal{F}$ has at most 2^{-k} chance to guess the value right. If $\mathcal{F}$ has queried H on $(m^*\|y_A\|y_B)$, and if $\mathcal{S}$ has guessed correctly the message m^*, $\mathcal{S}$ could set $H(m^*\|y_A\|y_B)$ to W. As explained in the proof of Lemma 1, $\mathcal{F}$ cannot query Confirmation/disavowal with (m^*, σ, y_A, y_B) for any nominative signature σ with respect to y_A and y_B. $\mathcal{S}$ can simulate the game without early abortion.

If $\mathcal{S}$ randomly picks a query of H as the guess of $H(m^*\|y_A\|y_B)$, the success probability of $\mathcal{S}$ is at least $1/Q$. Hence, $\mathcal{S}$ can solve the WCDH-II problem instance with probability at least $\epsilon' = (1 - 2^{-k})Q^{-1}\epsilon$, since the first component of the nominative signature is $H(m^*\|y_A\|y_B)^{x_A x_B^2 r} = g^{v x_A u^2 r}$ and thus Z can be computed from it. The running time of $\mathcal{S}$ is at most $t' = t + Qt_q + c$.

Theorem 5. *The NS proposed above is unforgeable (Def. 1) if both WCDH-I and WCDH-II problems are hard.*

This theorem follows directly from Lemma 1 and 2.

Theorem 6 (Strong Invisibility). *The NS proposed above has the property of strong invisibility (Def. 2) under WDDH assumption (Sec. 4).*

Proof. We show that if there exists a distinguisher $\mathcal{D}$ with advantage ϵ in Game Strong Invisibility, we can construct a WDDH distinguisher $\mathcal{D}^{WDDH}$ with advantage $\epsilon/2$. Given a random WDDH problem instance $g, T, U, V, W, X, Z \in G$ where $T = g^u$, $U = g^{u^2}$, $V = g^v$, $W = g^w$, $X = g^{uw}$, $Z = g^z$, a WDDH solver is to determine if $z \equiv u^2vw \pmod{p}$.

The simulation carried out by $\mathcal{D}^{WDDH}$ is similar to that in the proof of Lemma 2, that is, $\mathcal{D}^{WDDH}$ sets nominee B's public key y_B to T. There are two special cases in the simulation:

1. If $H(m\|y_1\|y_B)$ is queried or SignTranscript is queried on (m, y_1, y_B), $\mathcal{D}^{WDDH}$ will set $H(m\|y_1\|y_B)$ to $g^{r'}$ where $r' \in_R \mathbb{Z}_p$ and set σ_1 to $U^{r'x_1r}$, σ_2 to y_1^r, σ_3 to T^r and σ_4 to g^r, where x_1 is the private key corresponding to y_1 and $r \in_R \mathbb{Z}_p$.
2. Let Q be the maximum number of queries made. This also includes the hash query for the challenge message $(m^*\|y_A\|y_B)$. Suppose $\mathcal{D}^{WDDH}$ guesses correctly on the challenge message. $\mathcal{D}^{WDDH}$ sets the return value of $H(m^*\|y_A\|y_B)$ to V and challenge nominative signature $\sigma^* = (\sigma_1^*, \sigma_2^*, \sigma_3^*, \sigma_4^*)$ to (Z^{x_A}, W^{x_A}, X, W). Note that in this case, the signature is a valid nominative signature if $z \equiv u^2vw \pmod{p}$. Although $\mathcal{D}^{WDDH}$ is unable to prove or disprove a DH-tuple or a non-DH-tuple in the form of $(e(\sigma_4^*, g), e(H(m^*\|y_A\|y_B), \sigma_2^*), e(\sigma_3^*, y_B), e(\sigma_1^*, g))$ as $\mathcal{D}^{WDDH}$ does not know v or u^2, as explained in the proof of Lemma 1, $\mathcal{D}$ is not allowed to query Confirmation/disavowal with (m^*, σ, y_A, y_B) for any nominative signature σ with respect to y_A and y_B. Therefore, $\mathcal{D}^{WDDH}$ would not abort.

At the end of the game, $\mathcal{D}^{WDDH}$ outputs whatever $\mathcal{D}$ outputs. For event that $\mathcal{D}^{WDDH}$ guesses correctly on the challenge message, if $\mathcal{D}$ distinguishes the validity of σ^* successfully, so does $\mathcal{D}^{WDDH}$ on solving the WDDH problem instance. Hence $\mathcal{D}^{WDDH}$ has success probability of at least $\epsilon' = Q^{-1}\epsilon$. Similar to the evaluation of Lemma 1, the running time of $\mathcal{D}^{WDDH}$ is at most $t' = t + Qt_q + c$.

Theorem 7 (Security Against Impersonation). *The nominative signature scheme proposed in Sec. 5 is secure against impersonation with respect to Def. 3 under the Weak Discrete Logarithm (WDLOG) Assumption (Sec. 4) in the random oracle model.*

Both confirmation and disavowal protocols in the scheme proposed in Sec. 5 apply directly the techniques due to Kurosawa and Heng [12]. The security of our protocols is built upon that of theirs. The security model of theirs is similar to the security against impersonation in Game Impersonation. The difference is that our game has an extended set of oracles for the adversary to access due to the setting of NS.

Proof. Suppose there exists a (t, ϵ, Q)-impersonator $\mathcal{I}$ that wins in Game Impersonation, we construct a (t', ϵ')-algorithm $\mathcal{M}$ that solves a WDLOG problem instance. Suppose the input of $\mathcal{M}$ is (g, g^u, g^{u^2}). In the simulation, $\mathcal{M}$ performs similarly to that of the simulator in the proof of Lemma 2, but sets $pk_B = g^u$. Based on the proof techniques in [12], the advantage that $\mathcal{M}$ can extract the discrete logarithm of $e(y_B^r, y_B)$ to the base $e(g^r, g)$ is $\epsilon' = (\epsilon - 1/p)^2/2Q$. Note that the value of the discrete logarithm is $x_B^2 = u^2$. Thus, $\mathcal{M}$ can find the value of u by computing the square root of x_B^2. The running time of $\mathcal{S}$ is at most $t' = t + Qt_q + c$.

7 Conclusion

We proposed the first efficient non-interactive NS scheme which requires only one-move message transfer from the nominator to the nominee for signature generation. For making the security requirement of Invisibility realistic, we introduced a stronger requirement called Strong Invisibility, which captures the requirement that even the signer or the nominator is unable to determine the validity of a nominative signature even by recalling the entire signature generation transcripts. The technique we used in our NS construction is novel and may be useful for constructing some related schemes. We leave this as our further investigation.

References

1. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In *First ACM Conference on Computer and Communications Security*, pages 62–73. ACM, 1993.
2. D. Boneh, X. Boyen, and E.-J. Goh. Hierarchical identity based encryption with constant size ciphertext. In *Proc. EUROCRYPT 2005*, pages 440–456. Springer-Verlag, 2005. LNCS 3494.
3. D. Boneh and M. Franklin. Identity-based encryption from the Weil pairing. In *Proc. CRYPTO 2001*, pages 213–229. Springer-Verlag, 2001. LNCS 2139.
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. L. Guo, G. Wang, and D. Wong. Further discussions on the security of a nominative signature scheme. Cryptology ePrint Archive, Report 2006/007, 2006.
9. 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.
10. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In *Proc. EUROCRYPT 96*, pages 143–154. Springer, 1996. LNCS 1070.
11. 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.
12. K. Kurosawa and S. Heng. 3-move undeniable signature scheme. In *Proc. EUROCRYPT 2005*, pages 181–197, 2005. LNCS 3494.
13. D. Y. W. Liu, D. S. Wong, X. Huang, G. Wang, Q. Huang, Y. Mu, and W. Susilo. Nominative signature: Application, security model and construction. Cryptology ePrint Archive, Report 2007/069, 2007. http://eprint.iacr.org/2007/069.

14. V. Shoup. Lower bounds for discrete logarithms and related problems. In *Proc. EUROCRYPT 97*, pages 256–266. Springer-Verlag, 1997. LNCS 1233.
15. R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk. Universal designated-verifier signatures. In *Proc. ASIACRYPT 2003*, pages 523–542. Springer, 2003. LNCS 2894.
16. 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.