Ambiguous One-Move Nominative Signature Without Random Oracles*

Dennis Y. W. Liu[1,2], Duncan S. Wong[2], and Qiong Huang[3]

[1] School of Professional and Continuing Education, University of Hong Kong
[2] Department of Computer Science, City University of Hong Kong
[3] College of Informatics, South China Agricultural University
dennis.liu@hkuspace.hku.hk, duncan@cityu.edu.hk, csqhuang@gmail.com

Abstract. Nominative Signature is a useful tool in situations where a signature has to be created jointly by two parties, a nominator (signer) and a nominee (user), while only the user can verify and prove to a third party about the validity of the signature. In this paper, we study the existing security models of nominative signature and show that though the existing models have captured the essential security requirements of nominative signature in a strong sense, especially on the unforgeability against malicious signers/users and invisibility, they are yet to capture a requirement regarding the privacy of the signer and the user, and this requirement has been one of the original ones since the notion of nominative signature was first introduced. In particular, we show that it is possible to build a highly efficient nominative signature scheme which can be proven secure in the existing security models, while in practice it is obvious to find out from the component(s) of a nominative signature on whether a particular signer or user has involved in the signature generation, which may not be desirable in some actual applications. We therefore propose an enhanced security property, named "Ambiguity", and also propose a new *one-move* nominative scheme for fulfilling this new security requirement without random oracles, and among the various types of nominative signature, one-move is the most efficient type. Furthermore, this new scheme is at least 33% more efficient during signature generation and 17% shorter in signature size when compared with the existing one-move signature schemes without random oracles even that the existing ones in the literature may not satisfy this new Ambiguity requirement.

Keywords: nominative signature, undeniable signature, non-self-authenticating signature, security model

1 Introduction

In nominative signature (NS) [16,20], there are two parties: a signer (or *nominator*) A and a user (or *nominee*) B. To generate a nominative signature σ, A and B have to work together. However, once σ is generated, no one can verify its validity unless B and B is the only one who can show the validity or invalidity of an alleged nominative signature to a third party via running a confirmation/disavowal protocol. NS is useful in applications that there is a need in the division of signing and verifying abilities that two parties have to jointly create a non-self-authenticating signature, and only one of them is able to perform the verification of the signature, both to himself and to any third-parties. In [12], Huang et al. exemplified a practical application of NS in a healthcare system. In the system, a hospital may certify some personal medical records for a patient, for example, after a body checkup. For privacy, the patient would like to control on who can verify these personal medical records and how many of these records that a third party, for example, an insurance company, can verify. By using NS, the hospital and the patient will serve as the signer (or the nominator) and the user (or the nominee), respectively. Some may note that the hospital may simply release a medical document without participating in the nominative signature generation, but the patient can accuse the hospital of making false claims on the patient's medical records. The role of NS in this scenario is to produce a mutual agreement on the validity of the patient's personal medical records.

Since its introduction in 1996 [16], NS has been refined on its definitions and security models, and most of the security requirements of NS have been properly modeled to date [20,12,21,19] that include (1 and 2): unforgeability against malicious signers and users, (3) non-transferrability, (4) invisibility and (5) user-only conversion. Unforgeability against malicious signers (*resp.* users) prevents a signer (*resp.* user) from generating a nominative signature alone. A signer and a user have to work together in order to generate a valid one. Non-transferrability requires that a third party is not able to transfer the proof transcript of a confirmation/disavowal protocol to further convince other verifiers on the validity/invalidity of an alleged

* This paper is the full version of an extended abstract which will appear at ICISC 2013.

nominative signature. Invisibility is another main requirement of NS and restricts any party but the user to tell the validity of an alleged nominative signature, and user-only conversion is an optional property of NS that allows only the user to transform a valid nominative signature to a publicly verifiable one.

Despite the well modeling [20,12,21,19] of the five security requirements above, we have an ingrained view that a nominative signature σ is valid only if all the components of σ are verified positively. Suppose a nominative signature σ consists of 5 tuples $\sigma = (\alpha, \beta, \Delta, \Lambda, \theta)$, the nominative signature verification carried out by a user always checks the well-formedness and the validity of all these 5 tuples, and if any of the tuples is found to be invalid, σ is considered invalid. However, an invalid σ may contain some valid and **self-authenticating** components and these components might have already leaked the involvement of the signer A or the user B in the nominative signature generation. For example, suppose (α, β) in σ represents a digital signature generated by a signer A. If the other components $(\Delta, \Lambda, \theta)$ in σ are invalid, σ is deemed invalid. However, from (α, β), one can already tell that A has indeed involved in the generation of σ regardless whether σ is valid or not, as only the user B can determine its validity (note that σ can still be invalid even if (α, β) is a valid digital signature generated by A as the other components $(\Delta, \Lambda, \theta)$ can be invalid while only B can determine their validity and which makes such a scheme satisfy the invisibility requirement). More details are given in Sec. 3.2.

Let us use the aforementioned hospital-and-patient scenario as an illustration. Suppose the above NS is used and there is an alleged nominative signature $\sigma = (\alpha, \beta, \Delta, \Lambda, \theta)$ for certifying a patient's personal medical records. The hospital is the signer A and the patient is the user B in the NS. When issuing a certificate for B's personal medical records, the hospital A works jointly with B and generates a nominative signature σ. Although the validity of σ is unknown to the public due to non-transferrability and invisibility of the NS, everyone can check the validity of the digital signature components (α, β) of the hospital in σ. This may leak the fact that the hospital A has indeed got involved on issuing a certificate regarding B's personal medical records though σ as a whole is not able to be verified by the public.

Our Results. In this paper, we describe an NS scheme and proves that it is secure in the existing security models which capture the five conventional security requirements, namely unforgeability against malicious signers or users, non-transferrability, invisibility, and user-only conversion. Then we show that this nominative signature leaks the information on the involvement of a particular signer A. For capturing that no one, except the user B, is able to tell whether a particular signer or a user has participated in the generation of a nominative signature, we formalize a new security model, called "Ambiguity" and show that any NS scheme which can be proven secure under this new security model would not contain any sensible components which may leak the involvement of any particular party.

Besides the new Ambiguity model, we propose a new and highly efficient NS scheme which is proven secure under the existing models with respect to the five conventional security requirements. The new scheme is at least 33% more efficient, in terms of modular exponentiations during signature generation, and 17% shorter in signature size when compared with the most efficient NS schemes in the literature that has been proven secure without the random oracles. Also, we propose an improved NS scheme which satisfies not only the existing models, but also the new Ambiguity security requirement. Table 1 shows an efficiency comparison between this new NS scheme and the most efficiency NS schemes available in the literature that have been proven secure without random oracles, where the columns σ, A_{Key}, and B_{Key} represent the signature size, signer's key size, and user's key size, respectively; the column $SigGen$ represents the number of modular exponentiation operations carried out by a signer (Sign) and a user (Receive) individually; and the column *Ambiguity* indicates whether the corresponding scheme supports the new notion Ambiguity or not.

Scheme[a]	σ	A_{Key}	B_{Key}	$SigGen$[b]	*Ambiguity*
SH11 [21]	$3G+\mathbb{Z}_p$	$2G+(n+2)\mathbb{Z}_p$[c]	$5G+[2(n+1)+3]\mathbb{Z}_p$	3 + 8	$\surd$
LW12 [19]	$4G+2\mathbb{Z}_p$	$2G+2\mathbb{Z}_p$	$3G+3\mathbb{Z}_p$	1 + 5	$\times$
Our Scheme I	$4G+1\mathbb{Z}_p$	$2G+2\mathbb{Z}_p$	$[3+(m+1)]G+2\mathbb{Z}_p$ [d]	1 + 3	$\times$
Our Scheme II (EGE)	$4G+3\mathbb{Z}_p$	$2G+2\mathbb{Z}_p$	$3G+5\mathbb{Z}_p$	1 + 7	$\surd$
Our Scheme II (LE)	$6G+3\mathbb{Z}_p$	$2G+2\mathbb{Z}_p$	$6G+5\mathbb{Z}_p$	1 + 9	$\surd$

[a] EGE - ElGamal Encryption, LE - Linear Encryption
[b] No. of Modular Exponentiations in signature generation (Sign + Receive)
[c] n: No. of bits of message to be signed
[d] m: The public generators of group G included in the key of the programmable hash function (PHF)

Table 1. Efficiency Comparison Against Existing One-Move NS Schemes Without Random Oracles

Outline. In Sec. 2, we give the definition of nominative signature and review the existing security models. In Sec. 3, we discuss about the privacy of signer and user in a nominative signature and introduce the notion of Ambiguity. We propose a new scheme, which is proven secure in the existing security models, and show that the public can tell whether a particular signer has involved in the signature generation regardless an alleged nominative signature is valid or not. We then propose a security model for capturing Ambiguity. In Sec. 4, we propose a new nominative signature scheme and shows that it satisfies all the existing security requirements and also the new Ambiguity requirement without random oracles. In Sec. 5, we evaluate its efficiency and compare it with existing schemes. The paper is concluded in Sec. 6. In Appendix A, we also review the related work and results of nominative signature in the literature.

2 Nominative Signature: Definitions

A *One-Move* Nominative Signature (NS) consists of six probabilistic polynomial-time (PPT) algorithms (SystemSetup, SKeyGen, UKeyGen, NSVer, Conv, Ver) and three protocols (SigGen, Confirmation and Disavowal).

1. SystemSetup: On input a security parameter 1^k, where $k \in \mathbb{N}$, it outputs a list of system parameters denoted by param.
2. SKeyGen: On input param, it generates a public/private key pair (pk_A, sk_A) for the signer (i.e. nominator).
3. UKeyGen: On input param, it generates a public/private key pair (pk_B, sk_B) for the user (i.e. nominee).
4. NSVer: On input a message $m \in \{0,1\}^*$, a nominative signature σ, a signer public key pk_A and a user private key sk_B, it outputs valid or invalid.
5. Conv: On input a message-signature pair (m, σ), pk_A and sk_B, it outputs a standard (publicly verifiable) signature σ^{std} if valid $\leftarrow$ NSVer(m, σ, pk_A, sk_B); otherwise, it outputs $\perp$ symbolizing the failure of conversion.
6. Ver: On input $(m, \sigma^{std}, pk_A, pk_B)$, it outputs valid or invalid.
7. SigGen Protocol: A one-move protocol in which A makes one-move message transfer to B only. The common input of A and B is $(\mathsf{param}, m, pk_A, pk_B)$. A and B take sk_A, sk_B as their secret inputs, respectively. At the end of the protocol, A outputs nothing and B outputs a nominative signature σ. Let $\mathcal{S}(pk_A, pk_B)$ be the signature space.
8. Confirmation/Disavowal Protocol: On input (m, σ, pk_A, pk_B), B sets a bit μ to 1 if valid $\leftarrow$ NSVer(m, σ, pk_A, sk_B); otherwise, μ is set to 0. B then 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.

An NS scheme proceeds as follows. SystemSetup is first invoked. SKeyGen and UKeyGen are then executed to initialize a signer A and a user B. On a message m, A and B carries out SigGen protocol. As SigGen is one-move, A generates a *partial* nominative signature denoted by σ' and sends it to B. B then generates and outputs a nominative signature denoted by σ. Formally, SigGen consists of two algorithms, (Sign, Receive), which are carried out by signer A (who is holding (pk_A, sk_A)) and user B (who is holding (pk_B, sk_B)), respectively. SigGen protocol proceeds as follows:

1. A generates $\sigma' \leftarrow \mathsf{Sign}(\mathsf{param}, pk_B, m, sk_A)$ and sends σ' to B;
2. B generates $\sigma \leftarrow \mathsf{Receive}(\mathsf{param}, pk_A, m, \sigma', sk_B)$.

At the end of the protocol, B either outputs a nominative signature σ or $\perp$ indicating the failure of the protocol run.

Unlike the original definition in [20], the SigGen protocol defined above is specific to the one-move setting, that is, signer A initiates and generates a *partial* nominative signature σ', then B generates the final nominative signature σ upon receiving σ'. Note that the signature space should be specified explicitly in each NS construction.

For a nominative signature σ in the signature space $\mathcal{S}(pk_A, pk_B)$ (defined above in SigGen), the validity of σ can be determined by B using NSVer. If σ is valid, B can prove its validity to a third party C using the Confirmation protocol, otherwise, B can prove its invalidity to C using the Disavowal protocol.

Correctness: An NS scheme is correct if for all param $\leftarrow$ SystemSetup(1^k), $(pk_A, sk_A) \leftarrow$ SKeyGen(param), $(pk_B, sk_B) \leftarrow$ UKeyGen(param), message-signature pairs (m, σ) such that $\sigma \leftarrow$ Receive(param, pk_A, m, Sign(param, $pk_B, m, sk_A), sk_B$), and $\sigma^{std} \leftarrow$ Conv(m, σ, pk_A, sk_B), we have

- valid $\leftarrow$ NSVer(m, σ, pk_A, sk_B);

– C outputs accept at the end of the Confirmation protocol, and;
– valid $\leftarrow \mathsf{Ver}(m, \sigma^{std}, pk_A, pk_B)$.

For correctness, we also require that given a message m and a nominative signature σ in the signature space $\mathcal{S}(pk_A, pk_B)$, if invalid $\leftarrow \mathsf{NSVer}(m, \sigma, pk_A, sk_B)$, C outputs accept at the end of the Disavowal protocol.

The *soundness* of Confirmation (*resp.* Disavowal) protocol requires that no PPT user can convince a third party that an invalid (*resp.* valid) nominative signature is valid (*resp.* invalid).

Before describing the security games for NS, we begin with the description of oracles.

– OCreateSigner: This oracle generates a key pair (pk_A, sk_A) using SKeyGen, and returns pk_A.
– OCreateUser: This oracle generates a key pair (pk_B, sk_B) using UKeyGen, and returns pk_B.
– OCorrupt: On input a public key pk, if pk is generated by OCreateSigner or OCreateUser, the corresponding private key is returned; otherwise, $\perp$ is returned. pk is said to be *corrupted*.
– OSign: On input a message m, two distinct public keys, pk_1 (signer) and pk_2 (user), it returns σ' where σ' is a partial nominative signature generated using Sign.
– OReceive: On input a message m, a partial nominative signature σ', two distinct public keys, pk_1 (signer) and pk_2 (user), it returns a nominative signature σ.
– OProof: On input a message m, a nominative signature σ and two public keys pk_1 (signer) and pk_2 (user), the oracle, acting as the user (prover) and runs $\mathsf{NSVer}(m, \sigma, pk_1, sk_2)$ where sk_2 is the corresponding private key of pk_2. If the output of NSVer is valid, the oracle returns 1 and carries out the Confirmation protocol. Otherwise, it returns 0 and runs the Disavowal protocol.
– OConvert: On input (m, σ, pk_1, pk_2) such that valid $\leftarrow \mathsf{NSVer}(m, \sigma, pk_1, sk_2)$, it runs Convert and returns σ^{std}.

In all the oracles described above, the public keys in the queries of the oracles are assumed to be generated by the corresponding OCreateSigner or OCreateUser. This approach aligns with the multi-user setting and also the usual formalization under the registered-key model [2] and is based on that of [21].

We now review the security models for capturing the five conventional requirements of NS that are given in [20,12,21,19]. These requirements are: (1) **Unforgeability Against Malicious Users**, (2) **Unforgeability Against Malicious Signers**, (3) **Invisibility**, (4) **Non-transferability**, and (5) **User-only Conversion**.

2.1 Unforgeability Against Malicious Users

We require that a user cannot forge a nominative signature without the involvement of a signer.

Game Unforgeability Against Malicious Users: Let S be a challenger and F a forger.

1. (*Initialization*) Let $k \in \mathbb{N}$ be a security parameter. S runs param $\leftarrow \mathsf{SystemSetup}(1^k)$ and $(pk_A, sk_A) \leftarrow \mathsf{SKeyGen}(\mathsf{param})$, then invokes F with (param, pk_A).
2. (*Attacking Phase*) F adaptively queries OCreateSigner, OCreateUser, OCorrupt, and OSign.
3. (*Output Phase*) F outputs $(m^*, \sigma^*, pk_B, sk_B)$.

F *wins* the game if valid $\leftarrow \mathsf{NSVer}(m^*, \sigma^*, pk_A, sk_B)$ provided that

1. F has never queried $\mathsf{OCorrupt}(pk_A)$ for getting sk_A;
2. (pk_B, sk_B) is created through querying OCreateUser;
3. (m^*, pk_A, pk_B) has never been queried to OSign.

F's advantage in this game is defined as the probability that F wins.

Definition 1. *An NS is unforgeable against malicious users if no PPT forger F has a non-negligible advantage in Game Unforgeability Against Malicious Users.*

Oracles OProof and OConvert are not provided in the game as F can readily carry out these protocol/algorithm as a (malicious) user by making use of OCreateUser and OCorrupt oracles.

2.2 Unforgeability Against Malicious Signers

A malicious signer should not be able to forge a nominative signature without the help of a user.

Game Unforgeability Against Malicious Signers: Let S be a challenger and F a forger.

1. (*Initialization*) On input a security parameter $k \in \mathbb{N}$, S runs param $\leftarrow \mathsf{SystemSetup}(1^k)$ and $(pk_B, sk_B) \leftarrow \mathsf{UKeyGen}(\mathsf{param})$ and invokes F with (param, pk_B).

2. (*Attacking Phase*) F adaptively queries OCreateSigner, OCreateUser, OCorrupt, OReceive, OProof and OConvert.
3. (*Output Phase*) F outputs $(m^*, \sigma^*, pk_A, sk_A)$.

F *wins* the game if $\mathsf{valid} \leftarrow \mathsf{NSVer}(m^*, \sigma^*, pk_A, sk_B)$ provided that

1. F has never queried $\mathsf{OCorrupt}(pk_B)$ for getting sk_B;
2. (pk_A, sk_A) is created through OCreateSigner;
3. $(m^*, \sigma'^*, pk_A, pk_B)$ has never been queried to OReceive such that σ^* is the return.

F's advantage in this game is defined as the probability that F wins.

Definition 2. *An NS is unforgeable against malicious signers if no PPT forger F has a non-negligible advantage in Game Unforgeability Against Malicious Signers.*

OSign is not provided in the game above as F can readily carry out Sign as (malicious) signers by making use of OCreateSigner and OCorrupt.

2.3 Invisibility

We require that no verifier C (including signer A) but user B can tell the validity of a nominative signature. In the formalization, we define an auxiliary algorithm called NSSim (Nominative Signature Simulator). The algorithm takes $(\mathsf{param}, pk_A, pk_B, m, \sigma^{valid})$ as input, where σ^{valid} is a valid nominative signature for message m under pk_A and pk_B, outputs $\sigma^{invalid}$ so that $\sigma^{invalid} \in \mathcal{S}(pk_A, pk_B)$ but $\sigma^{invalid}$ is no longer a valid nominative signature for m under (pk_A, pk_B). The purpose of introducing NSSim is to explicitly define the capability of the public to convert a valid nominative signature to an invalid one while both σ^{valid} and $\sigma^{invalid}$ should look indistinguishable to C, and only B can tell which signature is valid and which one is not. Also note that NSSim has to be explicitly described in the construction of an NS scheme in order to have the new scheme be proven satisfying the Invisibility requirement.

Game Invisibility: The adversary in the game is a distinguisher D.

1. (*Initialization*) Same as that of Game Unforgeability Against Malicious Signers.
2. (*Attacking Phase*) Same as that of Game Unforgeability Against Malicious Signers.
3. (*Challenge Signature Generation Phase*) D chooses and sends a message m^* and (pk_A, pk_B) to the challenger while acting as a signer (indexed by pk_A) to carry out a run of SigGen with the challenger which acts as a user (indexed by pk_B). Let σ^{valid} be the nominative signature generated by the challenger in a SigGen protocol run, that is, $\mathsf{valid} \leftarrow \mathsf{NSVer}(m^*, \sigma^{valid}, pk_A, sk_B)$. The challenger then tosses a random coin $b \in_R \{0, 1\}$. If $b = 1$, the challenges sends σ^{valid} to D; otherwise, the challenge sends $\sigma^* \leftarrow \mathsf{NSSim}(\mathsf{param}, pk_A, pk_B, m, \sigma^{valid})$ to D.
4. (*Guess Phase*) D continues querying the oracles until it outputs a guess b'.

D *wins* the game if $b' = b$ provided that

1. D has never queried $\mathsf{OCorrupt}(pk_B)$ for getting sk_B;
2. (pk_A, sk_A) is created by querying OCreateSigner;
3. $(m^*, \sigma'^*, pk_A, pk_B)$ has never been queried to OReceive such that it returns σ^*;
4. $(m^*, \sigma^*, pk_A, pk_B)$ has never been queried to OProof or OConvert.

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

Definition 3. *An NS satisfies invisibility if no PPT distinguisher D has a non-negligible advantage in Game Invisibility.*

2.4 Non-transferability

This security property requires that a verifier C cannot convince other verifiers using a previous confirmation/disavowal proof transcript about the validity/invalidity of a given nominative signature. For a secure NS scheme, the Confirmation and Disavowal protocols should be perfect zero-knowledge so that no PPT verifier (including the signer) can transfer the proof transcript. The perfect zero-knowledge property implies non-transferability of proof transcripts as a verifier C can simulate the proof transcripts that look indistinguishable from the actual proof transcripts. A third party C can sample a signature from the signature space and create a proof transcript which looks indistinguishable from those generated from the Confirmation/Disavowal protocols running between the corresponding user B and the third party C.

2.5 User-only Conversion

The following game captures the requirement that no one but the user can convert a valid nominative signature to a publicly-verifiable one.

Game User-only Conversion: Let C be an adversary.

- (*Initialization*) Same as that of Game Unforgeability Against Malicious Signers.
- (*Attacking Phase*) Same as that of Game Unforgeability Against Malicious Signers.
- (*Challenge Signature Generation Phase*) C is given a challenge message-nominative-signature pair (m^*, σ^*) and the public keys of signer and user, respectively, (pk_A, pk_B) such that σ^* is valid on m^* under (pk_A, pk_B).
- (*Conversion Phase*) C outputs (m^*, σ^{std}).

C *wins* if $\mathsf{valid} \leftarrow \mathsf{Ver}(m, \sigma^{std}, pk_A, pk_B)$ provided that

1. C has never queried $\mathsf{OCorrupt}(pk_B)$ for getting sk_B;
2. (pk_A, sk_A) is created by querying $\mathsf{OCreateSigner}$;
3. $(m^*, \sigma^*, pk_A, pk_B)$ has never been queried to OProof or $\mathsf{OConvert}$.

C's advantage is defined as the probability that C wins.

Definition 4. *An NS satisfies user-only conversion if no PPT adversary C has a non-negligible advantage in Game User-only Conversion.*

Theorem 1 ([19]). *If an NS satisfies invisibility with respect to Def. 3, the scheme satisfies user-only conversion.*

3 Nominative Signature Supporting Ambiguity

In this section, we motivate the formalization of a new security requirement, "Ambiguity". It helps prevent an adversary from determining whether a signer A or a user B has involved in the generation of an alleged nominative signature σ. To the best of our knowledge, although Ambiguity has been considered as a folklore in the research of nominative signature for all these years since its first introduction in 1996 [16] and has also been studied in many other related cryptographic primitives, for example, the Ambiguous Optimistic Fair Exchange (AOFE) [14], it has never been formalized in the context of nominative signature, while it is an important and practical requirement. In the following, we first give a new NS construction that satisfies the existing security models (as defined in Sec. 2 above. The new construction is also more efficient, in terms computational complexity and signature size, than existing NS schemes. We, however, show that a nominative signature generated using this NS already leaks the involvement of a particular signer regardless the validity of the nominative signature. We make use of this NS scheme to illustrate the importance and practicality of this new "Ambiguity" security requirement.

3.1 An Efficient NS Construction (Our Scheme I)

This scheme employs the Boneh-Boyen short signature (BB) [4] and the Huang-Wong short convertible undeniable signature (HW) [13]. Particularly in SigGen, a signer A generates a BB signature $\sigma' = (\sigma^{BB}, r_A)$ and sends it to a user B, which signs on σ' using HW convertible undeniable signature. Below are the details, and the computational assumptions of the construction are given in Appendix B.

$\mathsf{SystemSetup}$: Given a security parameter $k \in \mathbb{N}$, the algorithm selects a bilinear group G with generator g of prime order p, and a collision resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$. It also selects a keyed group hash function [11] $\mathcal{H} = (\mathsf{PHF.Gen}, \mathsf{PHF.Eval})$, such that $\kappa \leftarrow \mathsf{PHF.Gen}(1^k)$ is the key, and we denote $\mathcal{H}_\kappa(m) = \mathsf{PHF.Eval}(\kappa, m)$, where $m \in \{0,1\}^*$. Let $\mathsf{param} = (k, H, \mathcal{H}, G, g, p)$.

$\mathsf{SKeyGen}$: On input param, it randomly generates $x_{A_1}, x_{A_2} \in_R \mathbb{Z}_p^*$ and calculates $y_{A_1} = g^{x_{A_1}}$ and $y_{A_2} = g^{x_{A_2}}$. Set the public key $pk_A = (y_{A_1}, y_{A_2})$ and private key $sk_A = (x_{A_1}, x_{A_2})$ for signer A.

$\mathsf{UKeyGen}$: On input param, it randomly picks $x_{B_1}, x_{B_2} \in_R \mathbb{Z}_p, \eta \in_R G$ and gets $\kappa \leftarrow \mathsf{PHF.Gen}(1^k)$. Calculate $y_{B_1} = g^{x_{B_1}}, y_{B_2} = g^{1/x_{B_2}}$, and set the public key $pk_B = (y_{B_1}, y_{B_2}, \eta, \kappa)$ and private key $sk_B = (x_{B_1}, x_{B_2})$ for user B.

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

1. A randomly picks $r_A \in \mathbb{Z}_p \setminus -\{\frac{x_{A_1}+H(m||y_B)}{x_{A_2}}\}$ where $y_B = y_{B_1}||y_{B_2}||\eta||\kappa$, computes $\sigma^{BB} = g^{1/(x_{A_1}+H(m||y_B)+x_{A_2}r_A)}$, and sends $\sigma' \leftarrow (\sigma^{BB}, r_A)$ to B. Here, the inverse $1/(x_{A_1} + H(m||y_B) + x_{A_2}r_A)$ is computed modulo p.
2. B verifies if $e(g,g) \stackrel{?}{=} e(\sigma^{BB}, y_{A_1} g^{H(m||y_B)} y_{A_2}^{r_A})$. B then randomly picks $\tau \in_R Z_p$, computes $\Delta \leftarrow \mathcal{H}_\kappa(\sigma')^{1/(x_{B_1}+\tau)}$, $\Lambda \leftarrow y_{B_2}^{\tau}$, $\theta \leftarrow \eta^{\tau}$, and sets $\sigma_U = (\Delta, \Lambda, \theta)$. The nominative signature is set to $\sigma = (\sigma', \sigma_U)$.

Signature Space: $\sigma = (\sigma', \sigma_U)$ is said to be in the signature space $\mathcal{S}(pk_A, pk_B)$ if σ' is a valid BB signature under pk_A on $m\|y_B$ and $\Delta, \Lambda, \theta \in G$.

NSVer: On input (m, σ, pk_A, sk_B), if $e(\Delta, y_{B_1}\Lambda^{x_{B_2}}) = e(\mathcal{H}_\kappa(\sigma'), g)$ and $e(\Lambda^{x_{B_2}}, \eta) = e(g, \theta)$, it outputs valid; otherwise, it outputs invalid.

Confirmation/Disavowal Protocol: If valid $\leftarrow$ NSVer(m, σ, pk_A, sk_B), B sends $\mu = 1$ and carries out the following proof system for showing the validity of σ_U to a verifier:

$$PoK\{x_{B_2} : e(\Delta, \Lambda)^{x_{B_2}} = e(\mathcal{H}_\kappa(\sigma'), g) \cdot e(\Delta, y_{B_1})^{-1}\};$$

otherwise, B sends $\mu = 0$ and carries out the following proof system with the verifier:

$$PoK\{x_{B_2} : e(\Delta, \Lambda)^{x_{B_2}} \neq e(\mathcal{H}_\kappa(\sigma'), g) \cdot e(\Delta, y_{B_1})^{-1}\}.$$

There exist efficient (3-move) *special honest-verifier zero-knowledge* protocols [7,8] for the instantiation of above proof systems. They can also be transformed into 4-move perfect zero-knowledge proofs of knowledge [9] so that there exists a PPT simulator that produces indistinguishable views for any verifier.

Conv: On input (m, σ, pk_A, sk_B) where σ is a valid nominative signature on m respect to pk_A and pk_B, the algorithm computes $cvt = \Lambda^{x_{B_2}}$ and sets $\sigma_U^{std} = (\sigma_U, cvt)$. It outputs a digital signature as $\sigma^{std} = (\sigma', \sigma_U^{std})$.

Ver: On input $(m, \sigma^{std}, pk_A, pk_B)$, it outputs valid if (1) $e(\Delta, y_{B_1} cvt) = e(\mathcal{H}_\kappa(\sigma'), g)$, and (2) $e(cvt, \eta) = e(g, \theta)$; otherwise, it outputs invalid.

For Invisibility (Sec. 2.3), we define $\sigma^{invalid} \leftarrow$ NSSim$(\mathsf{param}, pk_A, pk_B, m, \sigma^{valid})$ as follows. Given $\sigma^{valid} := (\sigma', \sigma_U)$, NSSim outputs $\sigma^{invalid}$ as (σ', σ_U^*) for randomly chosen $\Delta^*, \Lambda^*, \theta^* \in_R G$.

The security analysis for the scheme above is given in Appendix C.

3.2 Security Model: Ambiguity

The existing security model treats a nominative signature σ *as a whole* when determining its validity, that is, σ is considered valid if all individual components of σ are considered valid. However, this security model does not consider the **self-authenticating** individual components in σ. Those components may leak certain important information, for example, a particular signer/user's involvement in the signature generation. In our construction given in Sec. 3.1, the signer A creates a partial NS signature σ' which is a standard signature on $(m\|pk_B)$. The user B then creates an undeniable signature σ_U on σ' to form the final NS signature $\sigma = (\sigma', \sigma_U)$. Note that σ' can only be generated by A while the unforgeability property of the NS scheme still holds as neither A nor B is able to forge the entire signature σ alone. On invisibility, as the second part of σ, that is, σ_U, can only be verified by B while the public (including A) cannot tell whether σ_U is valid or not, even σ' is publicly verifiable, no one (including A) can conclude on whether the nominative signature σ *as a whole* is valid or not. However, the partial NS signature σ' is self-authenticated and already reveals A's participation in the signature generation regardless the validity or invalidity of the second part σ_U. We believe that this may act against the interest of the signer/user in real life situations. The hospital-and-patient scenario mentioned previously provides a good example. Though the certificate (i.e. σ) of a patient's personal medical records may not be self-authenticated that public verifiers are not able to check whether the certificate (*as a whole*) is valid or invalid, the first part of the certificate, i.e. σ', has already leaked the fact that the patient's personal medical records had been signed by a specific hospital. The patient, however, may not be happy to disclose this fact to the public. In real life situations, we believe that it is crucial to hide completely the information about whether a particular signer A or user B has got involved in the generation of an alleged nominative signature σ.

Informally speaking, given an alleged NS signature, we require that other than user B, no one (including signer A) can tell whether A or B has been involved in the signature generation protocol SigGen. Here, we propose two games for formalizing Signer Ambiguity and User Ambiguity.

Game Signer Ambiguity: The initialization and attacking phases are the same as that of Game Unforgeability Against Malicious Signers. In particular, the challenger S runs $(pk_B, sk_B) \leftarrow \mathsf{UKeyGen}(\mathsf{param})$ and sends pk_B to the adversary/distinguisher D_A while keeping sk_B secret. Below are the subsequent phases.

1. (*Challenge Selection Phase*) D_A arbitrarily chooses and sends two distinct challenge messages m_0^*, m_1^* and key pairs (pk_{A_0}, sk_{A_0}) and (pk_{A_1}, sk_{A_1}) of two signers to S. D_A then further runs $\sigma_i' \leftarrow \mathsf{Sign}(\mathsf{param}, pk_B, m_i^*, sk_{A_i})$ and sends σ_i' to S, for $i = 0, 1$.
2. (*Challenge Signature Generation Phase*) Upon receiving $\langle m_0^*, m_1^*, (pk_{A_0}, sk_{A_0}), (pk_{A_1}, sk_{A_1}), \sigma_0', \sigma_1' \rangle$, S tosses a coin $b \in_R \{0, 1\}$ and sends $\sigma^* \leftarrow \mathsf{Receive}(\mathsf{param}, pk_{A_b}, m_b^*, \sigma_b', sk_B)$ to D_A.
3. (*Guess Phase*) D_A outputs a guess bit b' for b.

D_A *wins* the game if $b' = b$ provided that

1. D_A has never queried $\mathsf{OCorrupt}(pk_B)$ for getting sk_B;
2. (pk_{A_0}, sk_{A_0}) and (pk_{A_1}, sk_{A_1}) are created by querying $\mathsf{OCreateSigner}$;
3. $(m^*, \sigma^*, pk_{A_i}, pk_B)$ has never been queried to OProof or $\mathsf{OConvert}$, for $i = 0, 1$.

D_A's advantage in this game is defined as $\mathsf{P}[b' = b] - \frac{1}{2}$.

Definition 5. *An NS has the property of Signer Ambiguity if no PPT distinguisher D_A has a non-negligible advantage in Game Signer Ambiguity.*

Game User Ambiguity: The initialization and attacking phases are the same as that of Game Unforgeability Against Malicious Signers. In particular, the challenger S generates $(pk_{B_0}, sk_{B_0}) \leftarrow \mathsf{UKeyGen}(\mathsf{param})$ and $(pk_{B_1}, sk_{B_1}) \leftarrow \mathsf{UKeyGen}(\mathsf{param})$ and sends pk_{B_0} and pk_{B_1} to the adversary/distinguisher D_A while keeping sk_{B_0} and sk_{B_1} secret. Below are the subsequent phases.

1. (*Challenge Selection Phase*) D_A arbitrarily chooses and sends two distinct challenge messages m_0^*, m_1^* and a pair (pk_A, sk_A) to S. D_A further runs $\sigma_i' \leftarrow \mathsf{Sign}(\mathsf{param}, pk_{B_i}, m_i^*, sk_A)$ and sends σ_i' to S, for $i = 0, 1$.
2. (*Challenge Signature Generation Phase*) Upon receiving $\langle m_0^*, m_1^*, pk_A, sk_A, \sigma_0', \sigma_1' \rangle$, S tosses a coin $b \in_R \{0, 1\}$, computes $\sigma_b^* \leftarrow \mathsf{Receive}(\mathsf{param}, pk_A, m_b^*, \sigma_b', sk_{B_b})$, and sends σ_b^* to D_A.
3. (*Guess Phase*) D_A outputs a guess bit b' for b.

D_A *wins* the game if $b' = b$ provided that

1. D_A has never queried $\mathsf{OCorrupt}(pk_{B_i})$ for getting sk_{B_i} for $i = 0, 1$;
2. (pk_A, sk_A) is created by querying $\mathsf{OCreateSigner}$;
3. $(m^*, \sigma^*, pk_A, pk_{B_i})$ has never been queried to OProof or $\mathsf{OConvert}$, for $i = 0, 1$.

D_A's advantage in this game is defined as $\mathsf{P}[b' = b] - \frac{1}{2}$.

Definition 6. *An NS has the property of User Ambiguity if no PPT distinguisher D_A has a non-negligible advantage in Game User Ambiguity.*

[19] does not satisfy User Ambiguity as the adversary can make use of r_A, which is generated by the adversary as signer A but is not masked in σ_b^*, to tell whether σ_0' or σ_1' is used in the generation of σ_b^* in the Game User Ambiguity above.

Definition 7. *An NS has the property of ambiguity if it satisfies both Signer Ambiguity and User Ambiguity.*

We say that a secure nominative signature (NS) scheme satisfies: (1) **Unforgeability Against Malicious Users**, (2) **Unforgeability Against Malicious Signers**, (3) **Invisibility**, (4) **Non-transferability**, (5) **User-only Conversion** and (6) **Ambiguity**.

Remark: In [12], Liu et al. proposed a security requirement called Strong Invisibility, which does not imply "Ambiguity". A nominative signature σ could have some individual components which are already self-authenticating, for example, the σ' part in the NS scheme proposed in Sec. 3.1, and from σ', it is publicly verifiable (i.e. self-authenticating) that A has involved (regardless of voluntarily or involuntarily) in the generation of σ'. Strong Invisibility [12] addresses a different issue, namely, the remaining components of a nominative signature σ, that is, those non-self-authenticating part of a nominative signature σ, for example, the $\sigma_U = (\Delta, \Lambda, \theta)$ components of σ in Sec. 3.1, can only be verified by user B even A "memorizes" all her SigGen transcripts.

4 A New NS Construction Supporting Ambiguity (Our Scheme II)

We propose a new NS scheme for achieving "Ambiguity". The scheme is based on the one in [19], and major modifications include the encryption of the randomness, r_A, using ElGamal encryption (EGE), and the inclusion of the ciphertext in the signature. We will also describe an alternative approach using Linear Encryption (LE) later in the same section.

SystemSetup: On input a security parameter $k \in \mathbb{N}$, the algorithm sets three cyclic groups G_1, G_2, and G_T of prime order $p \geq 2^k$ and a bilinear map $e : G_1 \times G_2 \rightarrow G_T$. It also picks a collision resistant hash function $H : \{0,1\}^* \rightarrow \mathbb{Z}_p$, and randomly selects generators $g_1 \in G_1$, $g_2 \in G_2$, and $g_3 \in \mathbb{Z}_p^*$. Set $\mathsf{param} = (p, G_1, G_2, G_T, g_1, g_2, g_3, H)$.

SKeyGen: On input param, it randomly picks $x_{A_1}, x_{A_2} \in_R \mathbb{Z}_p^*$, and computes $y_{A_1} = {g_2}^{x_{A_1}}$ and $y_{A_2} = {g_2}^{x_{A_2}}$. Set public key $pk_A = (y_{A_1}, y_{A_2})$, and private key $sk_A = (x_{A_1}, x_{A_2})$ for signer A.

UKeyGen: On input param, it randomly generates $x_{B_1}, x_{B_2}, x_{B_3}, x_{B_4} \in_R \mathbb{Z}_p^*$, and computes $y_{B_1} = {g_2}^{x_{B_1}}$, $y_{B_2} = {g_2}^{x_{B_2}}$, $y_{B_3} = {g_1}^{x_{B_3}}$ and $y_{B_4} = {g_3}^{x_{B_4}}$. Set public key $pk_B = (y_{B_1}, y_{B_2}, y_{B_3}, y_{B_4})$ and private key $sk_B = (x_{B_1}, x_{B_2}, x_{B_3}, x_{B_4})$ for user B.

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

1. A randomly picks $r_A \in_R \mathbb{Z}_p \setminus -\{\frac{x_{A_1}+H(m||y_B)}{x_{A_2}}\}$ where $y_B = y_{B_1}\|y_{B_2}\|y_{B_3}\|y_{B_4}$, computes $\sigma^{BB} = {g_1}^{1/(x_{A_1}+H(m||y_B)+x_{A_2}r_A)}$, and sends $\sigma' \leftarrow (\sigma^{BB}, r_A)$ to B.
2. B checks if $e(g_1, g_2) \stackrel{?}{=} e(\sigma^{BB}, y_{A_1} {g_2}^{H(m||y_B)} {y_{A_2}}^{r_A})$. If so, B computes $\sigma_1 = \sigma^{BB} {y_{B_3}}^{r_1}$ and $\alpha_1 = {g_1}^{r_1}$ where $r_1 \in_R \mathbb{Z}_p$, then randomly picks $r_B \in_R \mathbb{Z}_p \setminus -\{\frac{x_{B_1}+H(\sigma_1)}{x_{B_2}}\}$, and computes $\sigma_2 = {g_1}^{1/(x_{B_1}+H(\sigma_1)+x_{B_2}r_B)} {y_{B_3}}^{r_2}$ and $\alpha_2 = {g_1}^{r_2}$ where $r_2 \in_R \mathbb{Z}_p$. B also computes $c_A = r_A {y_{B_4}}^{r_3}$, $\alpha_3 = {g_3}^{r_3}$ where $r_3 \in_R \mathbb{Z}_p$. The signature is $\sigma = (\sigma_1, \sigma_2, c_A, r_B, \alpha_1, \alpha_2, \alpha_3)$.

Signature Space: σ is said to be in the signature space $\mathcal{S}(pk_A, pk_B)$ if σ_1, σ_2, α_1, $\alpha_2 \in G_1$, $c_A, \alpha_3 \in \mathbb{Z}_p^*$, $r_B \in \mathbb{Z}_p$.

NSVer: On input (m, σ, pk_A, sk_B) where σ is in $\mathcal{S}(pk_A, pk_B)$, set $r'_A = c_A / {\alpha_3}^{x_{B_4}}$ and check if

$$e(\alpha_1, y_{A_1} {g_2}^{H(m\|y_B)} {y_{A_2}}^{r_A})^{x_{B3}} \stackrel{?}{=} e(\sigma_1, y_{A_1} {g_2}^{H(m||y_B)} {y_{A_2}}^{r_A}) / e(g_1, g_2)$$
$$\wedge \; e(\alpha_2, y_{B_1} {g_2}^{H(\sigma_1)} {y_{B_2}}^{r_B})^{x_{B3}} \stackrel{?}{=} e(\sigma_2, y_{B_1} {g_2}^{H(\sigma_1)} {y_{B_2}}^{r_B}) / e(g_1, g_2)$$

If so, output valid; otherwise, output invalid.

Confirmation/Disavowal Protocol: If $\mathsf{valid} \leftarrow \mathsf{NSVer}(m, \sigma, pk_A, sk_B)$, B sends $\mu = 1$ and carries out the following zero-knowledge proof of knowledge with a verifier:

$$PoK\{(x_{B3}, x_{B4}, r_A) \; : \; {g_1}^{x_{B3}} = y_{B3} \; \wedge \; {g_3}^{x_{B4}} = y_{B4}$$
$$\wedge \; e(\alpha_1, y_{A_1} {g_2}^{H(m||y_B)} {y_{A_2}}^{r_A})^{x_{B3}} = e(\sigma_1, y_{A_1} {g_2}^{H(m||y_B)} {y_{A_2}}^{r_A}) / e(g_1, g_2)$$
$$\wedge \; e(\alpha_2, y_{B_1} {g_2}^{H(\sigma_1)} {y_{B_2}}^{r_B})^{x_{B3}} = e(\sigma_2, y_{B_1} {g_2}^{H(\sigma_1)} {y_{B_2}}^{r_B}) / e(g_1, g_2)\};$$

otherwise, B sends $\mu = 0$ and carries out the following zero-knowledge proof of knowledge with the verifier:

$$PoK\{(x_{B3}, x_{B4}, r_A) \; : \; {g_1}^{x_{B3}} = y_{B3} \; \wedge \; {g_3}^{x_{B4}} = y_{B4}$$
$$\wedge \; (e(\alpha_1, y_{A_1} {g_2}^{H(m||y_B)} {y_{A_2}}^{r_A})^{x_{B3}} \neq e(\sigma_1, y_{A_1} {g_2}^{H(m||y_B)} {y_{A_2}}^{r_A}) / e(g_1, g_2)$$
$$\vee \; e(\alpha_2, y_{B_1} {g_2}^{H(\sigma_1)} {y_{B_2}}^{r_B})^{x_{B3}} \neq e(\sigma_2, y_{B_1} {g_2}^{H(\sigma_1)} {y_{B_2}}^{r_B}) / e(g_1, g_2))\}.$$

Conv: On input (m, σ, pk_A, sk_B) where σ is a valid nominative signature on m respect to pk_A and pk_B, the algorithm computes $r_A = c_A / {\alpha_3}^{x_{B_4}}$, then randomly picks $r_4 \in_R \mathbb{Z}_p$, and sets $\delta = (\delta_1, \delta_2, r_A, r_4)$ where $\delta_1 = \sigma_1 / {\alpha_1}^{x_{B_3}}$ and $\delta_2 = {g_1}^{1/(x_{B_1}+H(\delta_1)+x_{B_2}r_4)}$.

Ver: On input (m, δ, pk_A, pk_B), it outputs valid if $e(g_1, g_2) = e(\delta_1, y_{A_1} {g_2}^{H(m||y_B)} {y_{A_2}}^{r_A})$ and $e(g_1, g_2) = e(\delta_2, y_{B_1} {g_2}^{H(\delta_1)} {y_{B_2}}^{r_4})$; otherwise, it outputs invalid.

For Invisibility (Sec. 2.3), we define $\sigma^{invalid} \leftarrow \mathsf{NSSim}(\mathsf{param}, pk_A, pk_B, m, \sigma^{valid})$ as follows. Given $\sigma^{valid} := (\sigma_1, \sigma_2, c_A, r_B, \alpha_1, \alpha_2, \alpha_3)$, NSSim outputs $\sigma^{invalid}$ as $(\sigma_1^*, \sigma_2^*, c_A, r_B, \alpha_1, \alpha_2, \alpha_3)$ where $\sigma_1^* = \sigma_1 R_1$, $\sigma_2^* = \sigma_2 R_2$ where R_1 and R_2 are randomly chosen from G_1.

Remark: In the security analysis, we will see that the "ambiguity" of the scheme above is based on the eXternal Diffie-Hellman (XDH) assumption [1] (Appendix B.4). An alternative approach is to use Linear Encryption [5], a natural extension of ElGamal encryption, to encrypt σ^{BB}. Linear encryption (LE) can be secure even in groups where a DDH-deciding algorithm exists. By using LE, we need to change pk_B to a triple of generators $g_i, g_{ii}, g_{iii} \in G_1$, and sk_B to exponents $x_i, x_{ii} \in \mathbb{Z}_p$ such that $g_i^{x_i} = g_{ii}^{x_{ii}} = g_{iii}$. To encrypt σ^{BB}, we randomly choose $a, b \in_R \mathbb{Z}_p$ and compute $(T_1, T_2, T_3) = (g_i^a, g_{ii}^b, \sigma^{BB} g_{iii}^{a+b})$. To recover σ^{BB}, we compute $T_3/(T_1^{x_i} T_2^{x_{ii}})$. LE is semantically secure against chosen-plaintext attacks, assuming the Decision Linear Problem (DLIN) assumption (Appendix B.5) holds, which is a weaker assumption and the scheme can remain secure even in groups where a DDH-deciding algorithm exists. This LE-based variant can therefore provide a potentially larger class of groups to choose from during implementation, while it is less efficient than the original EGE-based scheme. A detailed comparison will be given in Sec. 5.

4.1 Ambiguity Proof

Lemma 1. *[Signer Ambiguity] Suppose the NS above is unforgeable. If a polynomial-time algorithm D_A has an advantage ϵ in Game Signer Ambiguity, we can build a polynomial-time algorithm $D_{\mathsf{IND\text{-}CPA}}$ that breaks indistinguishability under chosen-plaintext attacks (IND-CPA) of ElGamal encryption PKE with advantage $\epsilon_1 = \frac{\epsilon}{2}$.*

Game 0 (Game Signer Ambiguity)	**Game 1** (IND-CPA)
$(pk_{A_0}, sk_{A_0}) \leftarrow \mathsf{USigGen}(\mathsf{param})$	$(pk, sk) \leftarrow \mathsf{Gen}(1^k)$
$(pk_{A_1}, sk_{A_1}) \leftarrow \mathsf{USigGen}(\mathsf{param})$	
$(m_0^*, m_1^*, pk_B, sk_B, \sigma_0', \sigma_1') \leftarrow D_A(pk_{A_0}, pk_{A_1})$	$(M_0^*, M_1^*) \leftarrow D_{\mathsf{IND\text{-}CPA}}(pk, sk)$
$b \leftarrow_R \{0,1\}$	$b \leftarrow_R \{0,1\}$
$\sigma_b^* \leftarrow \mathsf{Receive}(\mathsf{param}, pk_{A_b}, m_b^*, \sigma_b', sk_B)$	$c_b^* \leftarrow \mathsf{Enc}(M_b)$
$b' \leftarrow D_A(\sigma_b^*)$	$b' \leftarrow D_{\mathsf{IND\text{-}CPA}}(c_b^*)$

Fig. 1. Hopping Games for Signer Ambiguity

Proof. Refer to Fig. 1, suppose there is an adversary D_A which has a non-negligible advantage ϵ in **Game 0**, we construct an algorithm $D_{\mathsf{IND\text{-}CPA}}$ which has a non-negligible advantage ϵ_1 in **Game 1** where $\epsilon_1 = \frac{\epsilon}{2}$. $D_{\mathsf{IND\text{-}CPA}}$ invokes D_A with $y_{B_3} = pk$ obtained from challenger S_1 of the IND-CPA Game for PKE. $D_{\mathsf{IND\text{-}CPA}}$ randomly picks $x_{B_1}, x_{B_2}, x_{B_4} \in_R \mathbb{Z}_p^*$ and computes $y_{B_1} = g_2^{x_{B_1}}$, $y_{B_2} = g_2^{x_{B_2}}$, $y_{B_4} = g_3^{x_{B_4}}$. Set $y_{B_3} \leftarrow pk$ and $x_{B_3} \leftarrow sk$, and $pk_B = (y_{B_1}, y_{B_2}, y_{B_3}, y_{B_4})$ and $sk_B = (x_{B_1}, x_{B_2}, x_{B_3}, x_{B_4})$.

- $\mathsf{OReceive}$: Upon receiving (m, σ') from D_A for the generation of σ, $D_{\mathsf{IND\text{-}CPA}}$ generates $\sigma_1 = \sigma^{BB} {y_{B_3}}^{r_1}$ and $\alpha_1 = {g_1}^{r_1}$ where $r_1 \in_R \mathbb{Z}_p$, randomly picks $r_B \in \mathbb{Z}_p \setminus -\{\frac{x_{B_1} + H(\sigma_1)}{x_{B_2}}\}$ and generates $\sigma_2 = {g_1}^{1/(x_{B_1} + H(\sigma_1) + x_{B_2} r_B)} {y_{B_3}}^{r_2}$ and $\alpha_2 = {g_1}^{r_2}$ where $r_2 \in_R \mathbb{Z}_p$. Also, $D_{\mathsf{IND\text{-}CPA}}$ generates $c_A = r_A {y_{B_4}}^{r_3}$, $\alpha_3 = {g_3}^{r_3}$, where $r_3 \in_R \mathbb{Z}_p$. $D_{\mathsf{IND\text{-}CPA}}$ returns (m, σ_q) where $\sigma_q = (\sigma_1, \sigma_2, c_A, r_B, \alpha_1, \alpha_2, \alpha_3)$, and maintains a list $L_{OReceive}$ which contains mapping of randomness r_1 to σ_q of all $\mathsf{OReceive}$ queries.
- $\mathsf{OConvert}$: If (m, σ) is valid, due to the unforgeability against malicious signer property of the NS scheme, (m, σ) must be generated from a previous $\mathsf{OReceive}$ query. $D_{\mathsf{IND\text{-}CPA}}$ looks up $L_{OReceive}$, extracts r_1, calculates $\delta_1 = \sigma_1 / {y_{B_3}}^{r_1}$, $r_a = c_A / {\alpha_3}^{x_{B_4}}$ and generates a standard signature (δ_2, r_4) on δ_1. $D_{\mathsf{IND\text{-}CPA}}$ then returns $\delta = (\delta_1, \delta_2, r_a, r_4)$ to D_A.
- OProof: Given a (m, σ) pair with respect to pk_A and pk_B, $D_{\mathsf{IND\text{-}CPA}}$ is able to simulate the protocol to D_A by using the standard rewinding strategy because of the zero-knowledgeness of the protocol.

At some point, D_A sends m_0^* and m_1^* to $D_{\mathsf{IND\text{-}CPA}}$. $D_{\mathsf{IND\text{-}CPA}}$ run the SigGen protocol with D_A twice. The two partial nominative signatures are $\sigma'_{A_0} = ({\sigma_{A_0}}^{BB}, r_{A0})$ (on m_0^* under pk_{A_0}) and $\sigma'_{A_1} = ({\sigma_{A_1}}^{BB}, r_{A1})$ (on m_1^* under pk_{A_1}). $D_{\mathsf{IND\text{-}CPA}}$ sets $M_0 = {\sigma_{A_0}}^{BB}$ and $M_1 = {\sigma_{A_1}}^{BB}$ and submits (M_0, M_1) to the challenger of the IND-CPA Game of PKE. After S_1 returns the challenge "ciphertext" (σ_1^*, α_1^*), $D_{\mathsf{IND\text{-}CPA}}$ picks $r_B^* \in_R \mathbb{Z}_p \setminus -\{\frac{x_{B_1} + H(\sigma_1^*)}{x_{B_2}}\}$ and generates $\sigma_2^* = {g_1}^{1/(x_{B_1} + H(\sigma_1^*) + x_{B_2} r_B^*)} {y_{B_3}}^{r_2^*}$, $\alpha_2^* = {g_1}^{r_2^*}$ where $r_2^* \in_R \mathbb{Z}_p$, $c_A = r_{A0} {y_{B_4}}^{r_3^*}$ and $\alpha_3^* = {g_3}^{r_3^*}$, where $r_3^* \in_R \mathbb{Z}_p$. The challenge nominative signature is (m^*, σ^*) where

$\sigma^* = (\sigma_1^*, \sigma_2^*, c_A^*, r_B^*, \alpha_1^*, \alpha_2^*, \alpha_3^*)$. The probability that σ^* is a valid challenge signature is $\frac{1}{2}$ ((σ_1^*, α_1^*) encrypts M_0) and D_A will not abort. If σ^* is an invalid challenge ((σ_1^*, α_1^*) encrypts M_1), the probability that D_A can only succeed in guessing the bit is at most $\frac{1}{2}$. For the event that if D_A is able to distinguish that σ^* is valid under pk_{A_0} or pk_{A_1}, $D_{\mathsf{IND\text{-}CPA}}$ is also able to distinguish that $c^* = (\sigma_1^*, \alpha_1^*)$ is a ciphertext of M_0^* or M_1^*. Therefore, the advantage of $D_{\mathsf{IND\text{-}CPA}}$ is $\epsilon_1 = \frac{\epsilon}{2}$.

Lemma 2. *[User Ambiguity] Assume the NS proposed above is unforgeable. If D_A has an advantage ϵ in Game User Ambiguity, we can build a D_A that breaks anonymity under chosen-plaintext attack (ANO-CPA) [3] of ElGamal encryption scheme PKE with advantage $\epsilon_2 = \frac{\epsilon}{4}$.*

Game 0 (Game User Ambiguity)	**Game 1** (NS-ANO-CPA)
$(pk_{B_0}, sk_{B_0}) \leftarrow \mathsf{UKeyGen}(\mathsf{param})$	$(pk_{B_0}, sk_{B_0}) \leftarrow \mathsf{UKeyGen}(\mathsf{param})$
$(pk_{B_1}, sk_{B_1}) \leftarrow \mathsf{UKeyGen}(\mathsf{param})$	$(pk_{B_1}, sk_{B_1}) \leftarrow \mathsf{UKeyGen}(\mathsf{param})$
$(m_0^*, m_1^*, pk_A, sk_A, \sigma_0', \sigma_1') \leftarrow D_A(pk_{B_0}, pk_{B_1})$	$(m_0^*, m_1^*, pk_A, sk_A, \sigma_0', \sigma_1') \leftarrow D_{\mathsf{NS\text{-}ANO\text{-}CPA}}(pk_{B_0}, pk_{B_1})$
$b \leftarrow_R \{0,1\}$	$b \leftarrow_R \{0,1\}$, $d \leftarrow_R \{0,1\}$
$\sigma_b^* \leftarrow \mathsf{Receive}(\mathsf{param}, pk_A, m_b^*, \sigma_b', sk_{B_b})$	$\sigma_b^* \leftarrow \mathsf{Receive}(\mathsf{param}, pk_A, m_d^*, \sigma_d', sk_{B_b})$
$b' \leftarrow D_A(\sigma_b^*)$	$b' \leftarrow D_{\mathsf{NS\text{-}ANO\text{-}CPA}}(\sigma_b^*)$

Game 2 (ANO-CPA)
$(pk_0, sk_0) \leftarrow \mathsf{Gen}(1^k)$
$(pk_1, sk_1) \leftarrow \mathsf{Gen}(1^k)$
$M \leftarrow D_{\mathsf{ANO\text{-}CPA}}(pk_0, pk_1)$
$b \leftarrow_R \{0,1\}$
$c_b^* \leftarrow \mathsf{Enc}(M_b)$
$b' \leftarrow D_{\mathsf{ANO\text{-}CPA}}(c_b^*)$

Fig. 2. Hopping Games for User Ambiguity

Proof. [**Game 1**]: Refer to Fig. 2, suppose there is an adversary D_A which has a non-negligible advantage ϵ in **Game 0**, we can construct an algorithm $D_{\mathsf{NS\text{-}ANO\text{-}CPA}}$ which has a non-negligible advantage ϵ_1 in **Game 1** where $\epsilon_1 = \frac{\epsilon}{2}$. $D_{\mathsf{NS\text{-}ANO\text{-}CPA}}$ invokes D_A with the same sets of public key pairs pk_{B_0} and pk_{B_1}, obtained from challenger S_2 of **Game 1**.

The simulation of the oracles are the same as those in Game Signer Ambiguity. When D_A outputs a pair (m_0^*, m_1^*), it runs SigGen twice. The two partial nominative signatures are σ_0' and σ_1' which are submitted to $D_{\mathsf{NS\text{-}ANO\text{-}CPA}}$. $D_{\mathsf{NS\text{-}ANO\text{-}CPA}}$ submits σ_0' and σ_1' to S_2. S_2 returns σ_b^* to $D_{\mathsf{NS\text{-}ANO\text{-}CPA}}$, which then routes σ_b^* to D_A. The probability that σ_b^* is a valid challenge is $\frac{1}{2}$ (sk_{Bi} is used to generate σ^* from m_i^*, σ_i' for $i \in \{0,1\}$) and D_A will not abort. If σ_b^* is an invalid challenge (sk_{B1} is used to generate σ^* from m_0^*, σ_0' or sk_{B0} is used to generate σ^* from m_1^*, σ_1'), the probability that D_A can only succeed in guessing the bit is at most $\frac{1}{2}$. Therefore, if D_A is able to distinguish that (m_i^*, σ_b^*) is valid under (pk_A, pk_{B_i}) for $i \in \{0,1\}$, $D_{\mathsf{NS\text{-}ANO\text{-}CPA}}$ is also able to distinguish that (m_i^*, σ_b^*) is valid under (pk_A, pk_{B_i}) for $i \in \{0,1\}$ Obviously, $\epsilon_1 = \frac{\epsilon}{2}$.

[**Game 2**]: Suppose there is an algorithm $D_{\mathsf{NS\text{-}ANO\text{-}CPA}}$ which has a non-negligible advantage ϵ_1 in **Game 1**, we can construct an algorithm $D_{\mathsf{ANO\text{-}CPA}}$ which has a non-negligible advantage ϵ_2 in **Game 2** where $\epsilon_2 = \frac{\epsilon_1}{2}$. $D_{\mathsf{ANO\text{-}CPA}}$ invokes $D_{\mathsf{NS\text{-}ANO\text{-}CPA}}$ by generating randomly $x_{B_1 i}, x_{B_2 i}, x_{B_4 i} \in \mathbb{Z}_p^*$ and calculates $y_{B_1 i} = g_2^{x_{B_1 i}}$, $y_{B_2 i} = g_2^{x_{B_2 i}}$, $y_{B_4 i} = g_3^{x_{B_4 i}}$. Set $y_{B_3 i} \leftarrow pk_i$ and $x_{B_3 i} \leftarrow sk_i$ which are obtained from the challenger S_3 of the ANO-CPA Game of PKE. Let $(y_{B_1 i}, y_{B_2 i}, y_{B_3 i}, y_{B_4 i})$ be the public keys, pk_{Bi}, and $(x_{B_1 i}, x_{B_2 i}, x_{B_3 i}, x_{B_4 i})$ be the private keys, sk_{Bi}, of user B_i where $i = 0, 1$.

The encryption oracle for PKE can be simulated perfectly with the public key pk_0 and pk_1. When $D_{\mathsf{NS\text{-}ANO\text{-}CPA}}$ outputs a pair (m_0^*, m_1^*), it runs SigGen twice. The two partial nominative signatures are σ_i' (under $m_i^* || pk_{B_i}$) for $i = 0, 1$, which are submitted to $D_{\mathsf{ANO\text{-}CPA}}$. $D_{\mathsf{ANO\text{-}CPA}}$ calculates ${\sigma_1}^* = \sigma_0^{BB} {y_{B_3 0}}^{r_1^*}$, $\alpha_1^* = {g_1}^{r_1^*}$ where $r_1^* \in_R \mathbb{Z}_p$, encrypts ${r_A}^*$ using $x_{B_3 0}$ to form (c_A^*, α_3^*) and signs ${\sigma_1}^*$ using the BB signature scheme to obtain $M = {g_1}^{1/(x_{B_1 0} + H({\sigma_1}^*) + x_{B_2 0} {r_B}^*)}$. $D_{\mathsf{ANO\text{-}CPA}}$ submits M to S_3. S_3 returns a challenge ciphertext $c_b = (\sigma_2, \alpha_2)$. $D_{\mathsf{ANO\text{-}CPA}}$ then routes $\sigma_b^* = (\sigma_1^*, \sigma_2^*, c_A^*, r_B^*, \alpha_1^*, \alpha_2^*, \alpha_3^*)$ to $D_{\mathsf{NS\text{-}ANO\text{-}CPA}}$. The probability that σ_b^* is a valid challenge to $D_{\mathsf{ANO\text{-}CPA}}$ is $\frac{1}{2}$ ((σ_2^*, α_2^*) is encrypted under $x_{B_3 0}$) and $D_{\mathsf{ANO\text{-}CPA}}$ will not abort. If σ_b^* is an invalid challenge ((σ_2^*, α_2^*) is encrypted under $x_{B_3 1}$), the probability that $D_{\mathsf{NS\text{-}ANO\text{-}CPA}}$ can only succeed in guessing the bit is at most $\frac{1}{2}$. Therefore, if $D_{\mathsf{NS\text{-}ANO\text{-}CPA}}$ is able to distinguish that (m_i^*, σ_b^*) is valid under (pk_A, pk_{B_i}) for $i \in \{0,1\}$, $D_{\mathsf{ANO\text{-}CPA}}$ is also able to distinguish c_b^* where it is a valid encryption of M under pk_0 or pk_1. Obviously, $\epsilon_2 = \frac{\epsilon_1}{2}$.

Theorem 2 (Ambiguity). *The NS scheme proposed above satifies Ambiguity (Def. 7) if the scheme is unforgeable and ElGamal encryption satisfies ANO-CPA and IND-CPA.*

The theorem follows directly from Lemma 1 and 2.

The improved NS scheme also satisfies (1) **Unforgeability Against Malicious Users**, (2) **Unforgeability Against Malicious Signers**, (3) **Invisibility**, (4) **Non-transferability**, (5) **User-only Conversion**. The proof is similar to that for the scheme in [19] and is omitted here.

5 Efficiency Analysis and Comparison

In Table 2, we compare our Scheme I (the one does not satisfy Ambiguity) and Scheme II (the one supports Ambiguity) with the most efficient NS schemes in the literature. The comparison includes signature size, signer A and user B key sizes (termed as A_{Key} and B_{Key}), signature generation efficiency in terms of modular exponentiation calculation by A (Sign) and B (Receive) individually, and the security assumptions for unforgeability and invisibility. The table also shows whether the schemes satisfy Ambiguity, and whether the schemes can be proven secure without the assumption of random oracles.

Scheme	σ	A_{Key}	B_{Key}	$SigGen$[a]
ZY09 [25]	1G	2G+2$\mathbb{Z}_p^*$	2G+2$\mathbb{Z}_p^*$	1 + 2
SH11 [21]	3G+$\mathbb{Z}_p$	2G+(n+2)$\mathbb{Z}_p$[b]	5G+[2(n+1)+3]$\mathbb{Z}_p$	3 + 8
LW12 [19]	4G+2$\mathbb{Z}_p$	2G+2$\mathbb{Z}_p$	3G+3$\mathbb{Z}_p$	1 + 5
Our Scheme I	4G+1$\mathbb{Z}_p$	2G+2$\mathbb{Z}_p$	[3+(m+1)]G+2$\mathbb{Z}_p$ [c]	1 + 3
Our Scheme II (EGE)	4G+3$\mathbb{Z}_p$	2G+2$\mathbb{Z}_p$	3G+5$\mathbb{Z}_p$	1 + 7
Our Scheme II (LE)	6G+3$\mathbb{Z}_p$	2G+2$\mathbb{Z}_p$	6G+5$\mathbb{Z}_p$	1 + 9
	Unforgeability	Invisibility	Ambiguity	No Random Oracle
ZY09 [25]	CDH	3-DDH	√	×
SH11 [21]	CDH, DLP, DLIN	DLIN	√	√
LW12 [19]	q-SDH	DDH	×	√
Our Scheme I	q-SDH, HSDH	DHSDH	×	√
Our Scheme II (EGE)	q-SDH	XDH	√	√
Our Scheme II (LE)	q-SDH	DLIN	√	√

[a] No. of modular exponentiation operations in signature generation (Sign + Receive)
[b] n: No. of bits of each message to be signed
[c] m: The public generators of group G included in the key of the programmable hash function (PHF)

Table 2. Comparison with Existing One-Move NS Schemes

Our first efficient scheme is 33% more efficient, in terms of the number of modular exponentiation operations during signature generation, and 17% shorter in signature size than those in Liu et al's scheme [19], which is known to be the most efficient NS scheme proven secure without random oracles to date. The schemes [25] and [21] also satisfy the Ambiguity property, while the scheme in [25] relies on the random oracle assumption and in [21], the number of components in signer A's key is linear to the security parameter. Our Scheme II (EGE), the ElGamal encryption based variant of Scheme II, has constant size key and is proven without random oracles. In Scheme II (LE), the linear encryption based variant, though the performance is slightly lower than Scheme II (EGE), it relies on a weaker assumption (DLIN) and the scheme will remain secure even in groups where a DDH-deciding algorithm exists.

6 Conclusion

We proposed a new security notion called Ambiguity to Nominative Signature. This new notion ensures that a nominative signature will remain anonymous in the sense that no one (including the signer) but the user can tell whether a particular signer or user has involved in the generation of an alleged nominative signature. We no longer restrict ourselves to look into a nominative signature *as a whole*, but also require that any individual components of a nominative signature should not leak any information regarding the involvement of any particular party with respect to the generation of an alleged nominative signature. We formalized the Ambiguity notion, showed that it is possible to build a highly efficient nominative signature

secure in the existing model but not satisfying the ambiguity requirement. We also proposed a new and secure nominative signature scheme which also satisfies the ambiguity requirement. The new scheme is proven secure without the random oracle assumption.

References

1. G. Ateniese, J. Camenisch, and B. de Medeiros. Untraceable rfid tags via insubvertible encryption. In *ACM Conference on Computer and Communications Security*, pages 92–101. ACM, 2005.
2. B. Barak, R. Canetti, J. B. Nielsen, and R. Pass. Universally composable protocols with relaxed set-up assumptions. In *FOCS 2004*, pages 186–195. IEEE Computer Society, 2004.
3. M. Bellare, A. Boldyreva, A. Desai, and D. Pointcheval. Key-privacy in public-key encryption. In *ASIACRYPT 2001*, LNCS 2248, pages 566–582. Springer, 2001.
4. D. Boneh and X. Boyen. Short signatures without random oracles and the SDH assumption in bilinear groups. *J. Cryptology*, 21(2):149–177, 2008.
5. D. Boneh, X. Boyen, and H. Shacham. Short group signatures. In *CRYPTO 2004*, LNCS 3152, pages 41–55. Springer, 2004.
6. X. Boyen and B. Waters. Full-domain subgroup hiding and constant-size group signatures. In *PKC07*, volume 4450 of *LNCS*, pages 1–15. Springer, 2007.
7. E. Bresson and J. Stern. Proofs of knowledge for non-monotone discrete-log formulae and applications. In *ISC02*, LNCS 2433, pages 272–288. Springer, 2002.
8. J. Camenisch and V. Shoup. Practical verifiable encryption and decryption of discrete logarithms. In D. Boneh, editor, *CRYPTO03*, LNCS 2729, pages 126–144. Springer, 2003.
9. R. Cramer, I. Damgård, and P. MacKenzie. Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *PKC 2000*, LNCS 1751, pages 354–373. Springer, 2000.
10. L. Guo, G. Wang, D. S. Wong, and L. Hu. Further discussions on the security of a nominative signature scheme. In *SAM 2007*, pages 566–572. CSREA Press, June 2007.
11. D. Hofheinz and E. Kiltz. Programmable hash functions and their applications. In *CRYPTO08*, volume 5157 of *LNCS*, pages 21–38. Springer, 2008.
12. Q. Huang, D. Y. W. Liu, and D. S. Wong. An efficient one-move nominative signature scheme. *IJACT*, 1(2):133–143, 2008.
13. Q. Huang and D. S. Wong. Short and efficient convertible undeniable signature schemes without random oracles. *Theor. Comput. Sci.*, 476:67–83, 2013.
14. Q. Huang, G. Yang, D. S. Wong, and W. Susilo. Ambiguous optimistic fair exchange. In *ASIACRYPT 2008*, volume 5350 of *Lecture Notes in Computer Science*, pages 74–89, 2008.
15. Z. Huang and Y. Wang. Convertible nominative signatures. In *ACISP 2004*, LNCS 3108, pages 348–357. Springer, 2004.
16. S. J. Kim, S. J. Park, and D. H. Won. Zero-knowledge nominative signatures. In *PragoCrypt'96*, pages 380–392, 1996.
17. D. Y. W. Liu, S. Chang, and D. S. Wong. A more efficient convertible nominative signature. In *SECRYPT 2007*, pages 214–221. INSTICC Press, 2007.
18. D. Y. W. Liu, S. Chang, D. S. Wong, and Y. Mu. Nominative signature from ring signature. In *IWSEC 2007*, LNCS 4752, pages 396–411. Springer, 2007.
19. D. Y. W. Liu and D. S. Wong. One-move convertible nominative signature in the standard model. In *ProvSec 2012*, LNCS 7496, pages 2–20. Springer, 2012.
20. D. Y. W. Liu, D. S. Wong, X. Huang, G. Wang, Q. Huang, Y. Mu, and W. Susilo. Formal definition and construction of nominative signature. In *ICICS 2007*, LNCS 4861, pages 57–68. Springer, 2007.
21. J. C. N. Schuldt and G. Hanaoka. Non-transferable user certification secure against authority information leaks and impersonation attacks. In *ACNS 2011*, LNCS 6715, pages 413–430, 2011.
22. W. Susilo and Y. Mu. On the security of nominative signatures. In *ACISP 2005*, LNCS 3547, pages 329–335. Springer, 2005.
23. G. Wang and F. Bao. Security remarks on a convertible nominative signature scheme. In *SEC 2007*, IFIP 232. Springer, 2007.
24. W. Zhao, C. Lin, and D. Ye. Provably secure convertible nominative signature scheme. In *INSCRYPT 2008*, LNCS 5487, pages 23–40, 2008.
25. W. Zhao and D. Ye. Pairing-based nominative signatures with selective and universal convertibility. In *INSCRYPT 2009*, LNCS 6151, pages 60–74. Springer, 2011.

A Related Work

Nominative Signature (NS) was introduced by Kim et al. [16]. In their seminal paper, they also proposed the first NS, which was later found insecure [15]. The notion convertible nominative signature was also introduced in [15] and the first construction of a convertible nominative signature was proposed. In [22],

an attack against the scheme in [15] was described. Though the attack was later found invalid [10], new attacks against the scheme in [15] was found in [10,23].

In [20], Liu et al. proposed the first set of formal definitions and security models for Convertible NS (for simplicity, we use NS to represent convertible NS as well). They also proposed the first provably secure NS scheme under the models they defined. Their scheme requires at least four rounds of communication between the signer and the user during signature generation. More efficient nominative signature schemes were later proposed [17,18]. The scheme in [17] requires two rounds during nominative signature generation and the scheme in [18] is the first *one-move* (non-interactive) NS in the literature. Later, more one-move NS schemes [12,24,25] were proposed and proven secure in random oracle model. In [21], Schuldt et al. proposed a new NS scheme without random oracles, and recently in [19], a more efficient NS scheme than that in [21] was proposed. The new scheme achieved constant-size keys.

B Computational Assumptions

B.1 *q*-Strong Diffie-Hellman (*q*-SDH) Assumption [4,5]

Let G_1, G_2 be two cyclic groups of prime order p, respectively generated by g_1 and g_2.

The q-SDH assumption in (G_1, G_2) is defined as follows: given a $(q+3)$-tuple as input $(g_1, g_1^x, g_1^{x^2}, \cdots, g_1^{x^q}, g_2, g_2^x) \in G_1^{q+1} \times G_2^2$, output a pair $(c, g_1^{1/(x+c)})$ where $c \in \mathbb{Z}_p$ for a freely chosen value $c \in \mathbb{Z}_p \setminus \{-x\}$. We say that the (q, t, ϵ)-SDH assumption holds in (G_1, G_2) if there is no algorithm A which runs in time at most t, and satisfies the following condition:

$$\mathsf{P}[\mathcal{A}(g_1, g_1^x, g_1^{x^2}, \cdots, g_1^{x^q}, g_2, g_2^x) = (c, g_1^{1/(x+c)})] \geq \epsilon$$

where the probability is taken over the random choices of $g_1 \in G_1$ and $g_2 \in G_2$, the random choice of x in $\mathbb{Z}_p^*$, and random bits consumed by $\mathcal{A}$.

Remark: The q-SDH assumption applies to all G_1 and G_2 with an efficient bilinear map e, in which $G_1 = G_2$ or $G_1 \neq G_2$, and is proven in generic group model in [4].

B.2 HSDH Assumption [6]

The *q-Hidden Strong Diffie-Hellman* (*q-HSDH*) assumption (t, ϵ)-holds in G if there is no algorithm A which runs in time at most t, and satisfies the following condition:

$$\mathsf{P}[A(g, g^x, g^\beta, \{g^{1/(x+s_i)}, g^{s_i}, g^{\beta s_i}\}_{i=1}^q) = (g^{1/(x+s)}, g^s, g^{\beta s})] \geq \epsilon$$

where $s \in Z_p$ and $s \notin \{s_1, \cdots, s_q\}$, the probability is taken over the random choices of $x, \beta, s_1, \cdots, s_q \in Z_p$ and the random coins used by A.

B.3 DHSDH Assumption [13]

The *q-Decisional Hidden Strong Diffie-Hellman* (*q-DHSDH*) assumption (t, ϵ)-holds in G if there is no algorithm A which runs in time at most t, and satisfies the following condition:

$$\mathsf{P}[A(g, g^x, g^\beta, \mathbf{Q}, g^{\beta s}, g^{1/(x+s)}) = 1] - \mathsf{P}[A(g, g^x, g^\beta, \mathbf{Q}g^{\beta s}, Z) = 1]| \geq \epsilon$$

where $\mathbf{Q} = \{g^{1/(x+s_i)}, g^{s_i}, g^{\beta s_i}\}_{i=1}^q$, and the probability is taken over the random choices of $x, \beta, s_1, \cdots, s_q, s \in Z_p$ and $Z \in G$, and the random coins used by A.

B.4 External Diffie-Hellman (XDH) Assumption [1]

Let G_1, G_2 and G_T be cyclic groups of prime order p. Let g_1 be the generator of G_1 and g_2 the generator of G_2. Let $e : G_1 \times G_2 \to G_T$ be an efficiently computable map, which is asymmetric with only efficiently-computable isomorphism $\psi : G_2 \to G_1$. The *External Diffie-Hellman* (*XDH*) assumption (t, ϵ)-holds if there is no algorithm $\mathcal{A}$ which runs in time at most t, and satisfies the following condition:

$$P[\mathcal{A}(g_1, g_1^a, g_1^b, g_1^{c_b}, g_2, e) = b \mid c_0 = ab, c_1 \in_R \mathbb{Z}_p^*, b \in_R \{0,1\}] \geq \frac{1}{2} + \epsilon$$

where the probability is taken over the random choices of $g_1 \in G_1$, $g_2 \in G_2$, $a, b, c \in \mathbb{Z}_p^*$, and the random coins consumed by $\mathcal{A}$.

B.5 Decision Linear Problem (DLIN) Assumption [5]

Let G be a cyclic group of prime order p, and g_1, g_2 and g_3 the generators of G. The *Decision Linear Problem* (*DLIN*) assumption (t, ϵ)-holds if there is no algorithm A which runs in time at most t, and satisfies the following condition:

$$P[\mathcal{A}(g_1, g_2, g_3, g_2^{x_1}, g_3^{x_2}, g_1^{z_b} = b \mid z_0 \equiv x_1 + x_2, z_1 \in_R \mathbb{Z}_p^*, b \in_R \{0,1\}] \geq \frac{1}{2} + \epsilon$$

where the probability is taken over the random choices of $g_1, g_2, g_3 \in G$, $x_1, x_2, z \in \mathbb{Z}_p^*$, and the random coins consumed by $\mathcal{A}$.

C The Security Analysis of Our Scheme I

Informally speaking, the unforgeability against malicious users of our NS scheme relies on the unforgeability property of BB's [4] standard signature scheme. The unforgeability against malicious signers relies on the unforgeability property of the HW's undeniable signature scheme. The Invisibility property relies on the invisibility property of HW's undeniable signature scheme.

We now first show that our construction is secure against malicious users with respect to Def. 1.

Theorem 3. *Let $k \in \mathbb{N}$ be a security parameter. For the convertible nominative signature mentioned in Sec. 3.1, suppose there exists a (t, ϵ, Q)-user who is able to forge a valid σ with probability at least ϵ with running time at most t and number of queries at most Q in Game Unforgeability Against Users, then a (t', ϵ')-adversary can be constructed which can forge a valid standard signature in Boneh-Boyen full signature scheme [4] with probability at least $\epsilon' = \epsilon$. The total running time t' is at most $Qt_q + t + c$ where t_q denotes the maximum time required for responding a query and c is some constant time for setting up the system and generating the public/private keys.*

Proof. Suppose there is a (t, ϵ, Q)-forger F owning user B's private key $x_B = (x_{B_1}, x_{B_2})$ (using OCorrupt). We show that F can be transformed into a (t', ϵ')-algorithm S which is able to forge a valid standard signature in Boneh-Boyen full signature scheme [4]. Assume a public key PK_1 and PK_2 is given by the challenger *ch* of the Strong Existential Unforgeability Game defined in [4].

Game Simulation: S outputs param by invoking SystemSetup. For signer A, set, $y_{A_1} = PK_1$ and $y_{A_2} = PK_2$. Let (y_{A_1}, y_{A_2}) be the public keys and (x_{A_1}, x_{A_2}) be the private keys of signer A.

For a OSign query on input (m, y_A, y_2), where $y_A = (y_{A_1}, y_{A_2})$, and $y_2 = (y_{b_1}, y_{b_2}, \eta, \kappa)$, the simulation can be carried out perfectly by requesting a signature $\sigma'_q = (\sigma_q^{BB}, r_A)$ on m from *ch* and routes σ'_q to F.

Reduction: Eventually, F outputs a forged NS signature (m^*, σ^*). A forged standard BB signature $\sigma' = (\sigma^{BB}, r_A)$ on message $m||y_B$ can be extracted by S. Therefore, the probability $\epsilon' = \epsilon$ and the running time t' is at most $Qt_q + t + c$.

Theorem 4. *Let $k \in \mathbb{N}$ be a security parameter. For the convertible nominative signature mentioned in Sec. 3.1, if a (t, ϵ, Q)-signer can forge a valid nominative signature in Game Unforgeability Against Signers 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 forge a valid HW undeniable signature with probability at least $\epsilon' = \epsilon$ after running at most time $t' \leq t + Qt_q + c$ where t_q is the maximum time for simulating a query and c denotes some constant time for system setup and key generation.*

Proof. We show how to construct a (t', ϵ')-algorithm S to forge a HW undeniable signature from a (t, ϵ, Q)-forger F of a nominative signature who has signer A's private keys (x_{A_1}, x_{A_2}) (obtained by querying OCorrupt). Set, $pk_B = PK$ which is obtained from challenger *ch* of the Unforgeability Game of the underlying HW scheme. Let $pk_B = (y_{B_1}, y_{B_2}, \eta, \kappa)$ as the public key of B and $sk_B = (x_{B_1}, x_{B_2})$ as the private key of B.

The simulation of the oracles are shown below:

- OReceive: Upon receiving (m, σ') from F for the generation of nominative signature, S requests an undeniable signature σ_U on σ' from *ch* of the HW scheme and forms the nominative signature (m, σ) where $\sigma = (\sigma', \sigma_U)$.
- OConvert: Given a valid (m, σ) pair with respect to A and B, F is able to convert σ to a standard signature σ^{std}, since F possesses the universal convertor *ucvt*, obtained from the challenger of the underlying HW scheme. F returns $\sigma^{std} = (\sigma', \sigma_U^{std})$.

– OProof: Given a (m, σ) pair with respect to A and B, F is able to route the proof conversation between F and the confirmation/disavowal oracles of the underlying HW scheme.

Reduction: F outputs an NS signature-pair (m^*, σ^*) with $\sigma^* = (\sigma'^*, \sigma_U^*)$ where σ_U^* is a valid forgery of a convertible undeniable signature on σ'^*. For the event that if F forges a valid NS signature successfully, so does S on forging a valid HW undeniable signature.

Theorem 5 (Invisibility). *If the proposed NS construction is unforgeable and the underlying HW undeniable signature scheme satisfies invisibility, it has the property of invisibility with respect to Def. 3.*

Proof. We show that if there exists a distinguisher D, who has signer A's private keys sk_A (obtained by querying OCorrupt), with an advantage ϵ in Game Invisibility, we can construct a D_{CUS} that distinguishes the validity of HW undeniable signatures with advantage ϵ'. D sends a challenge message m^* to D^{CUS} and aims to carry out SigGen with D^{CUS}. Upon receiving (m^*, σ'^*) from D, D^{CUS} sets σ'^* as the challenge "message" of the invisibility game of the HW scheme. The challenger ch of the HW scheme returns a challenge undeniable signature σ_U^* on σ'^*. D_{CUS} then returns a challenge NS $\sigma^* = (\sigma'^*, \sigma_U^*)$ to D.

The simulation of the oracles are shown below:

– OReceive: Upon receiving (m, σ') from D for the generation of nominative signature, D_{NS} requests an NS signature $\sigma_U = \mathsf{USig}(sk_B, \sigma')$ from ch of the HW scheme. D_{CUS} returns an NS signature $\sigma = (\sigma', \sigma_U)$.
– OConvert: If (m, σ) is valid, D_{CUS} is able to convert σ to a standard signature σ^{std}, since D_{CUS} can route (m, σ) to the selective conversion oracle of HW scheme, which returns cvt of σ_U. D_{CUS} returns $\sigma^{std} = (\sigma', \sigma_U^{std})$.
– OProof: Given a (m, σ) pair with respect to A and B, D_{CUS} is able to route the proof conversation between D and the confirmation/disavowal oracles of the HW scheme.

For the event that if D distinguishes the validity of $\sigma^* = (\sigma'^*, \sigma_U^*)$ successfully, so does D^{CUS} on distinguishing σ_U^* on the "message" σ'^*.

By Theorem 1 and 5 we obtain the following corollary immediately.

Corollary 1 (User-only Conversion). *The proposed NS scheme satisfies the property of user-only conversion.*