A MORE EFFICIENT CONVERTIBLE NOMINATIVE SIGNATURE*

Dennis Y. W. Liu, Shuang Chang and Duncan S. Wong
Dept. of Computer Science, City University of Hong Kong, Hong Kong, China

Keywords: Digital Signature, Nominative Signature, Undeniable Signature.

Abstract: Nominative signature provides an interesting share of power between a nominator and a nominee in which a nominative signature, generated jointly by the nominator and the nominee, can only be verified with the aid of the nominee. In this paper, we propose a new construction of nominative signature which has a higher network efficiency than the existing one (Liu et al., 2007). In addition, our scheme is the first one supporting nominee-only conversion. We also enhance the security model of nominative signature for capturing this new property.

1 INTRODUCTION

Since the introduction of undeniable signature (Chaum and van Antwerpen, 1990; Chaum, 1990; Chaum and van Antwerpen, 1992), there have been many other *non-self-authenticating* notions introduced. One of them is **Nominative Signature (NS)** (Kim et al., 1996; Huang and Wang, 2004; Susilo and Mu, 2005; Guo et al., 2006; Liu et al., 2007). An NS scheme allows a *nominator* A and a *nominee* B to jointly 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 (Kim et al., 1996), it was not until recently that the notion has finally been formalized (Liu et al., 2007). In the past, besides lacking a formal definition, the application of NS has also been questioned. In (Liu et al., 2007), it is shown that NS is a very useful tool for constructing *user certification systems*, which concern about letting a user prove the validity of his own birth certificate, driving licence and academic transcripts, issued by authorities. In such a system, the user (nominee) B does not want a verifier C to disseminate B's certificate s (issued by an authority A – nominator), while B wants to convince C that s is authentic, that is, signed by A. NS is very suitable for this type of applications because NS does not allow A to prove the validity of B's certificate s. This property greatly helps protect the interest of the users.

Related Work. The notion and construction of NS were first proposed in (Kim et al., 1996). However, the construction was later found to be flawed (Huang and Wang, 2004). In (Huang and Wang, 2004), the notion of convertible NS was introduced. This variant of NS allows the nominee to convert an NS to a publicly verifiable one. A new scheme was also proposed. However, it has later been found to be insecure (Susilo and Mu, 2005; Guo et al., 2006).

In (Liu et al., 2007), the first formal security model for NS was defined and a proven secure construction was proposed. This security model is currently the strongest one. However, there is no definition for the nominee-only conversion from a nominative signature to a standard signature. About their construction, the signature generation protocol requires to run a three-move Witness Indistinguishable protocol (Feige and Shamir, 1990; Kurosawa and Heng, 2005).

Our Results. We propose a new construction which does not require the key generation protocol to run a

*The work was supported by a grant from CityU (Project No. 7001844).

three-move Witness Indistinguishable protocol. The key generation can be completed in just two message flows between the nominator and the nominee, and therefore, has a higher network efficiency than the current one (Liu et al., 2007). We also extend the security model for capturing nominee-only conversion.

Paper Organization. We define convertible NS and propose an enhanced security model in Sec. 2. We then propose a new NS construction in Sec. 3. The security analysis is given in Sec. 4. The paper is concluded in Sec. 5.

2 DEFINITIONS AND SECURITY MODELS

We extend the definition of NS from (Liu et al., 2007) to a *convertible* NS. Specifically, in addition to the properties captured in the definition of (Liu et al., 2007), we also allow the nominee, but nobody else, to convert an NS to a standard signature which can be self-authenticated.

A nominative signature (NS) consists of five PPT (probabilistic polynomial-time) algorithms (SystemSetup, KeyGen, $\mathsf{Ver}^{\mathsf{nominee}}$, Convert, $\mathsf{Ver}^{\mathsf{public}}$) and three protocols (SigGen, Confirmation, Disavowal). On input a security parameter 1^k, where $k \in \mathbb{N}$, SystemSetup is first invoked for generating a list of system parameters denoted by param. Then, $(pk, sk) \leftarrow \mathsf{KeyGen}(\mathsf{param})$ is executed for each entity in the system. We use A and B to denote the nominator and the nominee, respectively. Let (pk_A, sk_A) be A's key pair and (pk_B, sk_B) be B's. To generate an NS σ on some message $m \in \{0,1\}^*$, A and B carry out the SigGen protocol.

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

The validity of σ can be determined by B using $\mathsf{Ver}^{\mathsf{nominee}}$ on input (m, σ, pk_A, sk_B). To convince a third party C on the validity/invalidity of σ, B as prover and C as verifier carry out a Confirmation or Disavowal protocol:

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

To convert σ to a standard signature σ^{pub}, B runs $\mathsf{Convert}(m, \sigma, pk_A, sk_B)$. After the conversion, the validity of σ^{pub} can be verified by running $\mathsf{Ver}^{\mathsf{public}}(m, \sigma^{pub}, pk_A, pk_B)$.

Correctness. If all the algorithms mentioned above are executed accordingly, the NS scheme should satisfy the following requirements. (1) $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B)$; (2) C outputs accept at the end of the Confirmation protocol; and (3) $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{public}}(m, \sigma^{pub}, pk_A, pk_B)$.

On the security of NS, (Liu et al., 2007) defines (1) unforgeability, (2) invisibility, (3) security against impersonation and (4) non-repudiation. We will adopt these definitions. Besides, we also define an additional security model for capturing the notion of (5) nominee-only conversion.

Before elaborating the corresponding games, we first describe some oracles that are to be provided to adversaries:

- CreateUser: On input an identity I, it generates a key pair (pk_I, sk_I) using KeyGen and returns pk_I.
- Corrupt: On input a public key pk, if pk is generated by CreateUser or in $\{pk_A, pk_B\}$, the corresponding private key is returned; otherwise, $\perp$ is returned. pk is said to be *corrupted*.
- SignTranscript: On input a message m, two distinct public keys, pk_1 (the nominator) and pk_2 (the nominee), and one parameter called $role \in \{\mathsf{nil}, \mathsf{nominator}, \mathsf{nominee}\}$,
  - if $role = \mathsf{nil}$, $\mathcal{S}$ simulates SigGen and returns $(\sigma, trans_\sigma)$ where σ is a valid nominative signature (i.e. $\mathsf{valid} \leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_1, sk_2)$ where sk_2 is the corresponding private key of pk_2) and $trans_\sigma$ is the transcript of the execution of SigGen.
  - if $role = \mathsf{nominator}$, $\mathcal{S}$ (as nominee with public key pk_2) simulates a run of SigGen with the adversary (which acts as the nominator with pk_1);
  - if $role = \mathsf{nominee}$, $\mathcal{S}$ (as nominator with pk_1) simulates a run of SigGen with the adversary (which acts as the nominee with 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.
  - Passive attack: If $\mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_1, sk_2)$ outputs valid, the oracle returns $\mu = 1$ and a transcript of the Confirmation protocol. Otherwise, $\mu = 0$ and a transcript of the Disavowal protocol are returned.
  - Active/concurrent attack: the oracle checks if σ is valid as in the passive attack. If so, the oracle returns $\mu = 1$ and executes the Confirmation

protocol with the adversary (acting as a verifier). Otherwise, the oracle returns $\mu = 0$ and executes the Disavowal protocol with the adversary. The difference between active and concurrent attack is that the adversary interacts serially with the oracle in the active attack while it interacts with different instances of the oracle concurrently in the concurrent attack.

- OracleConvert: On input (m,σ,pk_1,pk_2) such that valid $\leftarrow$ $\mathsf{Ver}^{\mathsf{nominee}}(m,\sigma,pk_1,sk_2)$, the oracle returns σ^{pub} such that valid $\leftarrow$ $\mathsf{Ver}^{\mathsf{public}}(m,\sigma^{pub},pk_1,pk_2)$.

2.1 Unforgeability

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

1. (*Initialization*) First, param $\leftarrow$ SystemSetup(1^k) is executed and key pairs (pk_A,sk_A) and (pk_B,sk_B) for nominator A and nominee B, respectively, are generated using KeyGen. Then $\mathcal{F}$ is invoked on input $(\mathsf{param},pk_A,pk_B)$.
2. (*Attacking Phase*) $\mathcal{F}$ can make queries to the oracles mentioned above.
3. (*Output Phase*) $\mathcal{F}$ outputs a pair (m^*,σ^*) as a forgery of A's nominative signature on message m^* with B as the nominee.

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

Definition 1 *(Liu et al., 2007) An NS scheme is said to be unforgeable if no PPT forger $\mathcal{F}$ has a non-negligible advantage in Game Unforgeability.*

2.2 Invisibility

Game Invisibility: The initialization phase is the same as that of Game Unforgeability. Let $\mathcal{D}$ be a distinguisher that can query any of the oracles mentioned. At some point in the attacking phase, $\mathcal{D}$ outputs a message m^* and requests for a challenge nominative signature σ^* on m^*. σ^* is generated based on the outcome of a hidden coin toss b. If $b = 1$, σ^* is generated using SigGen. If $b = 0$, σ^* is chosen randomly from the signature space with respect to pk_A and pk_B. At the end of the game, $\mathcal{D}$ outputs a guess b'.

$\mathcal{D}$ *wins* if $b' = b$ and (1) $\mathcal{D}$ does not corrupt sk_B; (2) $(m^*,pk_A,pk_B,role)$ has never been queried to SignTranscript; (3) (m^*,σ^*,pk_A,pk_B) has never been queried to Confirmation/disavowal. $\mathcal{D}$'s advantage in this game is defined as $|\Pr[b' = b] - \frac{1}{2}|$.

Definition 2 *(Liu et al., 2007) An NS scheme satisfies invisibility if no PPT distinguisher $\mathcal{D}$ has a non-negligible advantage in Game Invisibility.*

2.3 Security Against Impersonation

Game Impersonation: Let $\mathcal{I}$ be an impersonator. The initialization phase is the same as that of Game Unforgeability. The two other phases are as follows.

- (*Preparation Phase*) $\mathcal{I}$ may query any of the oracles. $\mathcal{I}$ prepares (m^*,σ^*,μ) where m^* is some message, σ^* is in the signature space with respect to pk_A and pk_B and μ is a bit.
- (*Impersonation Phase*) If $\mu = 1$, $\mathcal{I}$ (as nominee) executes Confirmation protocol with the simulator (as a verifier). If $\mu = 0$, $\mathcal{I}$ executes Disavowal protocol instead.

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

Definition 3 *(Liu et al., 2007) An NS scheme is secure against impersonation if no PPT impersonator $\mathcal{I}$ has a non-negligible advantage in Game Impersonation.*

2.4 Non-repudiation

Game Non-repudiation: Let $\mathcal{B}$ be a *cheating* nominee which can query any of the oracles. The initialization phase is the same as that of Game Unforgeability. The two other phases are: (1) (*Preparation Phase*) $\mathcal{B}$ prepares (m^*,σ^*) where m^* is a message and σ^* is in the signature space with respect to pk_A and pk_B. (2) (*Repudiation Phase*) If $\mathsf{Ver}^{\mathsf{nominee}}(m^*,\sigma^*,pk_A,sk_B) =$ valid, $\mathcal{B}$ executes Disavowal protocol with the simulator (acting as a verifier) on (m^*,σ^*,pk_A,pk_B); otherwise, the Confirmation protocol is carried out.

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

Definition 4 *(Liu et al., 2007) An NS scheme is secure against repudiation if no PPT cheating nominee has a non-negligible advantage in Game Non-repudiation.*

We now propose an additional security requirement. This one is for *convertible* NS.

2.5 Nominee-only Conversion

This security notion requires that it should be infeasible for anyone but the nominee to convert a valid nominative signature to a publicly-verifiable one. We consider the following game.

Game Nominee-only Conversion: The initialization phase is the same as that of Game Unforgeability. An adversary $\mathcal{C}$ can query any of the oracles. At the end of the game, $\mathcal{C}$ outputs $(m^*, \sigma^*, \tilde{\sigma}^{pub})$.

$\mathcal{C}$ *wins* if valid $\leftarrow$ $\mathsf{Ver}^{\mathsf{nominee}}(m^*, \sigma^*, pk_A, sk_B)$, and valid $\leftarrow$ $\mathsf{Ver}^{\mathsf{public}}(m, \tilde{\sigma}^{pub}, pk_A, pk_B)$. The restrictions are (1) $\mathcal{C}$ has never corrupted sk_B; (2) $(m^*, \sigma^*, pk_A, pk_B)$ has never been queried to Oracle-Convert; (3) $(m^*, \sigma, pk_A, pk_B)$ has never been queried to Confirmation/disavowal for any nominative signature σ. $\mathcal{C}$'s advantage is defined as the probability that $\mathcal{C}$ wins.

Definition 5 *An NS satisfies nominee-only conversion if no PPT adversary $\mathcal{C}$ has a non-negligible advantage in Game Nominee-only Conversion.*

3 OUR CONSTRUCTION

In this section, we propose a new construction, which has a higher network efficiency than the one in (Liu et al., 2007) during signature generation and also supports nominee-only conversion.

3.1 Preliminaries

Ring Signature. Our construction makes use of a special structure of the ring signature scheme due to (Rivest et al., 2001) (RST scheme). In the RST scheme, it is assumed that each ring member has a one-way trapdoor permutation f and its inverse f^{-1} (i.e. the trapdoor). There is a random "glue" value z in each RST ring signature and the scheme requires a block cipher $SE : \{0,1\}^k \times \{0,1\}^k \rightarrow \{0,1\}^k$. We denote the output of $SE(K,m)$ by $SE_K(m)$. Let SE^{-1} be the decryption algorithm of the block cipher.

Verifiable Decryption. A verifiable decryption (VD) scheme for a relation $\Re$ (Camenisch and Shoup, 2003) has an encryption/decryption algorithm pair (Enc, Dec) associated with a verification protocol suite which allows a prover who possesses the secret key of a public key pk to convince a verifier that given δ and ciphertext ψ encrypted under pk, ψ is the encryption of ω where $(\omega, \delta) \in \Re$. In other words, the prover is the decryptor who holds the secret key sk.

In our NS scheme, we adopt the proofing protocols for VD of discrete logarithm due to (Camenisch and Shoup, 2003) to implement the Confirmation/Disavowal protocols. The protocols of (Camenisch and Shoup, 2003) are special honest verifier zero-knowledge (SHVZK). In our NS scheme, however, we need concurrent zero-knowledge (CZK) protocols for security proofs. Therefore, we apply the standard transformations (Goldreich and Kahan, 1996; Cramer et al., 2000; Damgård, 2000; Gennaro, 2004) and convert them to CZK variants in the common reference string (CRS) model.

3.2 Our Scheme

SystemSetup: It generates a cyclic group G of k-bit prime order p and a random generator g. Assume that each element of G can be encoded distinctly into a k-bit binary string. Let $H : \{0,1\}^* \rightarrow \{0,1\}^k$ be a hash function. Set $\mathsf{param} = (1^k, SE, G, p, g, H)$.

KeyGen: For nominator A, it generates (f_A, f_A^{-1}), a pair of signing and verification algorithms (Sig_A, Ver_A) and a VD encryption/decryption pair (Enc_A, Dec_A). Set $pk_A = (f_A, Ver_A, Enc_A)$ and $sk_A = (f_A^{-1}, Sig_A, Dec_A)$. Nominee B's key pair is generated similarly.

SigGen Protocol: Let $m \in \{0,1\}^*$ be a message. The protocol is carried out as follows.

1. B picks $r \in_R \mathbb{Z}_p$, computes $R_B = g^r$ and sends R_B to A.
2. (RST scheme) A picks $z \in_R \{0,1\}^k$ and computes $y_B = f_B(R_B)$, $y_A = SE_K^{-1}(z) \oplus SE_K(z \oplus y_B)$, and $R_A = f_A^{-1}(y_A)$, where $K = H(m \| pk_A \| pk_B)$. $\sigma^{ring} = (z, R_A, R_B)$ forms a ring signature on "message" K. A sends σ^{ring} to B.
3. B checks if $z = SE_K(SE_K(z \oplus f_B(R_B)) \oplus f_A(R_A))$ and $R_B = g^r$. If so, B outputs $\sigma = (\sigma^{ring}, Enc_B(r), \sigma^{standard})$, where $\sigma^{standard} = Sig_B(m \| \sigma^{ring} \| Enc_B(r))$.

(*Signature Space.*) $\sigma = (\sigma_1, \sigma_2, \sigma_3)$ is in the signature space with respect to pk_A and pk_B if σ_1 is a valid ring signature on "message" K, σ_2 is properly formed with respect to the VD scheme, i.e., σ_2 can be properly decrypted to some message m, and σ_3 is a valid standard signature of B on "message" $m \| \sigma_1 \| \sigma_2$ (i.e. with respect to Ver_B). Note that if σ is in the signature space, it does not imply that σ is a *valid* NS. The validity can only be verified by B:

$\mathsf{Ver}^{\mathsf{nominee}}$: On input (m, σ, pk_A, sk_B) where $\sigma = (\sigma^{ring}, Enc_B(r), \sigma^{standard})$ is in the signature space, compute $r = Dec_B(Enc_B(r))$ and check if

1. $\sigma^{ring} = (z, R_A, R_B)$ is valid , i.e. $z = SE_K(SE_K(z \oplus f_B(R_B)) \oplus f_A(R_A))$;
2. $Ver_B(m \| \sigma^{ring} \| Enc_B(r),\ \sigma^{standard}) = 1$; and
3. $R_B = g^r$.

If all of them are correct, output valid; otherwise, output invalid.

Confirmation/Disavowal Protocol: On input (m, σ, pk_A, pk_B) where σ is in the signature space, if valid $\leftarrow$ $\mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B)$, B sets $\mu = 1$; otherwise, sets $\mu = 0$.

- If $\mu = 1$, B proves to C that the decryption of $Enc_B(r)$ is a discrete log of R_B using the corresponding VD protocol.
- If $\mu = 0$, B proves to C that the decryption of $Enc_B(r)$ is NOT a discrete log of R_B using the corresponding VD protocol.

Convert: On input (m, σ, pk_A, pk_B) such that valid $\leftarrow$ $\mathsf{Ver}^{\mathsf{nominee}}(m, \sigma, pk_A, sk_B)$, B outputs a standard signature $\sigma^{pub} = (\sigma, r)$.

Verify: On input $(m, \sigma^{pub}, pk_A, pk_B)$, check if all of the followings are valid:

1. $\sigma^{ring} = (z, R_A, R_B)$ is valid, i.e. that is, $z = SE_K(SE_K(z \oplus f_B(R_B)) \oplus f_A(R_A))$;
2. $Ver_B(m \| \sigma^{ring} \| Enc_B(r),\ \sigma^{standard}) = 1$; and
3. if $R_B = g^r$.

Discussion. In the SigGen protocol, there are only two message flows between A and B. When compared with (Liu et al., 2007), our construction does not need a three-move Witness Indistinguishable protocol, and therefore has a higher network efficiency. It remains an open problem if a non-interactive SigGen protocol can be built, namely, there is only one message flow between A and B.

4 SECURITY ANALYSIS

Lemma 1 (Cheating Nominee) *Let $k \in \mathbb{N}$ be a security parameter. If a (t, ε, Q)-nominee can forge a valid NS 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 invert a trapdoor one-way permutation with probability at least $\varepsilon' = Q^{-2}(1 - 2^{-k})\varepsilon$ after running at most time $t' = t + Qt_q + c$ where t_q is the maximum time for simulating one oracle query and c is some constant.*

Lemma 2 (Cheating Nominator) *If a (t, ε, Q)-nominator can forge a valid NS, there exists a (t', ε')-adversary which can existentially forge a standard signature under the model of chosen message attacks (Goldwasser et al., 1988) with probability at least $\varepsilon' = (1 - 2^{-k}Q)\varepsilon$ after running at most time $t' = t + Qt_q + c$, where t_q is the maximum time for simulating one oracle query and c is some constant.*

Theorem 1 (Unforgeability) *The NS scheme proposed above is unforgeable (Def. 1) if there exists trapdoor one-way permutations and existentially unforgeable signature schemes secure against chosen message attacks (Goldwasser et al., 1988).*

This theorem follows directly from Lemma 1 and 2. Proofs of the lemmas are in Appendix A.

Theorem 2 (Invisibility) *If there exists a (t, ε, Q)-distinguisher $\mathcal{D}$ in Game Invisibility and existentially unforgeable signature schemes secure against chosen message attacks (Goldwasser et al., 1988), there exists a (t', ε')-distinguisher $\mathcal{D}^{Enc}$ which has advantage at least $\varepsilon' = \varepsilon$ to launch an adaptive chosen ciphertext attack to the encryption algorithm of VD by 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.*

Theorem 3 (Nominee-only Conversion) *The convertible NS scheme proposed satisfies nominee-only conversion (Def. 5) if there exists trapdoor one-way permutations and existentially unforgeable signature schemes against chosen message attacks (Goldwasser et al., 1988).*

All proofs above are in Appendix A.

Both confirmation and disavowal protocols in this scheme are zero-knowledge. Therefore, the scheme already satisfies the requirements of security against impersonation (Def. 2.3). In addition, by using the technique of Theorem 2, it can be shown that compromising the security against impersonation of this scheme reduces to compromising the underlying zero-knowledge confirmation/disavowal protocols of VD of discrete logarithm in (Camenisch and Shoup, 2003). We skip the details but readers can readily derive the reduction from the proving technique of Theorem 2.

The scheme also satisfies the requirement that nominee cannot repudiate. This follows directly the soundness property of the underlying VD of discrete logarithm protocol (Camenisch and Shoup, 2003).

5 CONCLUSION

We proposed a convertible NS scheme which does not require to run a three-move Witness Indistinguishable protocol for signature generation and only two message flows are required to complete the generation. This gives our construction an advantage in network efficiency over the one in (Liu et al., 2007). We also enhanced the security model of (Liu et al., 2007) for capturing nominee-only conversion. It remains an open problem to construct an NS with a non-interactive signature generation process.

REFERENCES

Camenisch, J. and Shoup, V. (2003). Practical verifiable encryption and decryption of discrete logarithms. In *CRYPTO 2003*, pages 126–144.

Chaum, D. (1990). Zero-knowledge undeniable signatures. In *Proc. EUROCRYPT 90*, pages 458–464. Springer-Verlag. LNCS 473.

Chaum, D. and van Antwerpen, H. (1990). Undeniable signatures. In *Proc. CRYPTO 89*, pages 212–216. Springer-Verlag. LNCS 435.

Chaum, D. and van Antwerpen, H. (1992). Cryptographically strong undeniable signatures, unconditionally secure for the signer. In *Proc. CRYPTO 91*, pages 470–484. Springer-Verlag. LNCS 576.

Cramer, R., Damgård, I., and MacKenzie, P. D. (2000). Efficient zero-knowledge proofs of knowledge without intractability assumptions. In *PKC 00*, pages 354–372.

Damgård, I. (2000). Efficient concurrent zero-knowledge in the auxiliary string model. In *EUROCRYPT00*, pages 418–430.

Feige, U. and Shamir, A. (1990). Witness indistinguishable and witness hiding protocols. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 416–426.

Gennaro, R. (2004). Multi-trapdoor commitments and their applications to proofs of knowledge secure under concurrent man-in-the-middle attacks. In *CRYPTO 04*, pages 220–236.

Goldreich, O. and Kahan, A. (1996). How to construct constant-round zero-knowledge proof systems for np. *J. Cryptology*, 9(3).

Goldwasser, S., Micali, S., and Rivest, R. (1988). A digital signature scheme secure against adaptive chosen-message attack. *SIAM J. Computing*, 17(2):281–308.

Guo, L., Wang, G., and Wong, D. (2006). Further discussions on the security of a nominative signature scheme. Cryptology ePrint Archive, Report 2006/007.

Huang, Z. and Wang, Y. (2004). Convertible nominative signatures. In *Proc. of Information Security and Privacy (ACISP'04)*, pages 348–357. Springer-Verlag. LNCS 3108.

Kim, S. J., Park, S. J., and Won, D. H. (1996). Zero-knowledge nominative signatures. In *PragoCrypt'96, International Conference on the Theory and Applications of Cryptology*, pages 380–392.

Kurosawa, K. and Heng, S. (2005). 3-move undeniable signature scheme. In *Proc. EUROCRYPT 2005*, pages 181–197. LNCS 3494.

Liu, D. Y. W., Wong, D. S., Huang, X., Wang, G., Huang, Q., Mu, Y., and Susilo, W. (2007). Nominative signature: Application, security model and construction. Cryptology ePrint Archive, Report 2007/069. http://eprint.iacr.org/2007/069.

Rivest, R., Shamir, A., and Tauman, Y. (2001). How to leak a secret. In *Proc. ASIACRYPT 2001*, pages 552–565. Springer-Verlag. LNCS 2248.

Susilo, W. and Mu, Y. (2005). On the security of nominative signatures. In *Proc. of Information Security and Privacy (ACISP'05)*, pages 329–335. Springer-Verlag. LNCS 3547.

A APPENDIX

A.1 Proof of Lemma 1

Proof. If a (t,ε,Q)-forger $\mathcal{F}$ after obtaining $sk_B = (f_B^{-1}, Dec_B, Sig_B)$ via Corrupt can win Game Unforgeability with at least probability ε by producing a valid nominative signature $\sigma^* = (\sigma^{ring*}, Enc_B(r^*), \sigma^{standard*})$ on some message m^* after running at most time t and making at most Q queries (all kinds of oracle queries which include game specific oracles and random oracles), we construct a (t',ε')-algorithm $\mathcal{S}$ which inverts a trapdoor one-way permutation $\hat{f} : \{0,1\}^k \to \{0,1\}^k$ on some random input $\hat{y} \in_R \{0,1\}^k$ with at least probability ε' after running at most time t'. We will derive the values of ε' and t' in this proof. Let the ring signature σ^{ring*} on "message" K^* be (z^*, R_A^*, R_B^*). Assume that all hash evaluations and SE/SE^{-1} evaluations made by $\mathcal{F}$ are obtained from oracle access.

Game Simulation: $\mathcal{S}$ first generates param according to SystemSetup, and sets nominator A's public key to $pk_A = (\hat{f}, Ver_A, Enc_A)$ and private key to $sk_A = (\perp, Sig_A, Dec_A)$ where $\perp$ denotes an empty string as the trapdoor information of $\hat{f}$ is unavailable to $\mathcal{S}$. For nominee B, the public and private keys are all generated according to KeyGen. Then $\mathcal{F}$ is invoked on $(1^k, pk_A, pk_B)$. Oracles are also simulated.

For oracle CreateUser, a new key pair is generated using KeyGen and the public key is returned. For oracle Corrupt, for example, if B is queried, sk_B is returned. As restricted by the game and the statement of this lemma, A's private key cannot be compromised by $\mathcal{F}$. For a SignTranscript query, there are three cases:

- Case (1): If $role =$ nil, a nominative signature is simulated by following SigGen. There is one exception: if A is indicated as the nominator (i.e. $pk_1 = pk_A$ in Game Unforgeability), $\mathcal{S}$ is unable to follow the protocol to compute an inversion of $\hat{f}$. But thanks to random oracle, $\mathcal{S}$ can do the evaluation of $\hat{f}$ and assign the appropriate SE/SE^{-1} evaluations with a randomly generated 'glue' value $z \in_R \{0,1\}^k$. This simulation is computationally indistinguishable from a real simulation due to the idealness of random oracles.

- Case (2): If $role =$ nominator, $\mathcal{S}$ simulates an execution of SigGen protocol with $\mathcal{F}$. $\mathcal{S}$ acts as the nominee. Similar to Case (1), $\mathcal{S}$ can simply follow the exact execution of SigGen protocol even if the nominee is A. This is because when A is the nominee, A does not require to invert $\hat{f}$.
- Case (3): If $role =$ nominee, $\mathcal{S}$ acts as nominator and simulates an execution of SigGen protocol with $\mathcal{F}$. During the simulation, $\mathcal{S}$ follows the execution of SigGen protocol except when the nominator is indicated as A. In this case, we use the strategy described in Case (1) by assigning appropriate SE/SE^{-1} evaluations such that $\mathcal{S}$ only needs to evaluate the forward direction of $\hat{f}$. Note that by following the specification of SigGen protocol, $\mathcal{S}$ acting as A only needs to compute the ring signature component after receiving the first message $R_{\tilde{B}}$ from $\mathcal{F}$ which is acting as nominee $\tilde{B}$. Hence by randomly generate $z \in_R \{0,1\}^\ell$ and properly adjust the SE/SE^{-1} evaluations on $(z \oplus f_{\tilde{B}}(R_{\tilde{B}}))$ and z, $\mathcal{S}$ does not need to invert $\hat{f}$.

For Confirmation/disavowal and OracleConvert queries, since $\mathcal{S}$ has all parties' private key component Dec, $\mathcal{S}$ can always carry out the confirmation/disavowal protocols and perform the standard signature conversion.

Reduction: We follow the argument of the "gap" technique used in the soundness proof of the ring signature of (Rivest et al., 2001). The "gap" technique is based on an observation that the valid ring signature σ^{ring*} forged by $\mathcal{F}$ must have a gap somewhere between two cyclically consecutive occurrences of SE, and $\mathcal{F}$ must be forced to fill in this gap by computing the inverse of the corresponding trapdoor one-way permutation. Since $\mathcal{F}$ has to query $\mathcal{S}$ for the results of SE and SE^{-1} evaluations, $\mathcal{S}$ can make use of the queries of the two SE/SE^{-1} evaluations, which form the gap, to assign the desired $\hat{y}$. If $\mathcal{F}$ makes at most Q queries, the probability that $\mathcal{S}$ guesses correctly the two SE/SE^{-1} queries is at least Q^{-2}. In σ^{ring*}, there are only two possible gaps. One is at $y_2 = f_B(R_B^*)$ and the other one at $y_1 = \hat{f}(R_A^*)$. If the gap is at y_2, then with at most 2^{-k} probability that $f_B^{-1}(y_2)$ is of the form g^{r^*} where $r^* \in_R \mathbb{Z}_p$ since y_2 is uniformly distributed over $\{0,1\}^k$. Therefore, with probability $(1-2^{-k})$, the gap is at y_1. $\mathcal{S}$'s goal is to set y_1 to $\hat{y}$. As described above, $\mathcal{S}$ randomly picks two SE/SE^{-1} queries as the guess of the two SE/SE^{-1} queries for forming the gap. Once $\mathcal{F}$ outputs $\sigma^{ring*} = (z^*, R_A^*, R_B^*)$, $\mathcal{S}$ outputs R_A^* as the result of $\hat{f}^{-1}(\hat{y})$.

Hence if the advantage of $\mathcal{F}$ in Game Unforgeability is ε, the probability that $\mathcal{S}$ inverts the trapdoor one-way permutation is at least $Q^{-2}(1-2^{-k})\varepsilon$. If each random oracle query takes at most time t_q to finish, the simulation time of the game for $\mathcal{F}$ is at most $t + Qt_q + c$ where c denotes some constant time for system setup and key generation. $\Box$

A.2 Proof of Lemma 2

Proof. If a (t,ε,Q)-forger $\mathcal{F}$ after obtaining via oracle Corrupt the nominator A's private key $sk_A = (f_A^{-1}, Sig_A, Enc_A)$ and is able to win Game Unforgeability with probability at least ε by producing a valid nominative signature $\sigma^* = (\sigma^{ring*}, Enc_B(r^*), \sigma^{standard*})$ on some message m^* after running at most time t and making at most Q queries, where $\sigma^{standard*}$ is a standard signature of nominee B on "message" $m^* \| \sigma^{ring*} \| Enc_B(r^*)$, we construct a (t',ε')-algorithm $\mathcal{S}$ to forge a signature with respect to a standard signature scheme (Sig^*, Ver^*) with probability at least ε', in the model of existential forgery against chosen message attacks (Goldwasser et al., 1988) after running at most time t'. By forging a standard signature, $\mathcal{S}$ is given a problem instance Ver^* but not Sig^* and $\mathcal{S}$ is to output a pair $(\tilde{m},\tilde{\sigma})$ such that $Ver^*(\tilde{m},\tilde{\sigma}) = 1$ after adaptively querying a signing oracle. The restriction is that $\tilde{m}$ has never been queried to the signing oracle.

In the simulation of Game Unforgeability, $\mathcal{S}$ sets the public key of nominee B to $pk_B = (f_B, Ver^*, Enc_B)$ and private key to $sk_B = (f_B^{-1}, \perp, Dec_B)$. The simulation is similar to that in the proof of Lemma 1 with the exception that for each query of B's standard signature, the query will be forwarded to the signing oracle of Sig^* by $\mathcal{S}$ and the answer is relayed back.

First, we show that with probability at most $2^{-k}Q$, the ring signature σ^{ring*} in σ^* is an output of oracle SignTranscript. As restricted by Game Unforgeability, $(m^*, pk_A, pk_B, role)$ should have never been queried to oracle SignTranscript. Hence if oracle SignTranscript has output a nominative signature which contains the ring signature σ^{ring*}, it should be a valid ring signature for some message, say $\hat{K}$, with respect to ring members identified by pk_1 and pk_2. Since $\mathcal{S}$ simulates all the hash functions and SE/SE^{-1} evaluations by picking returning values uniformly at random from the corresponding spaces, the chance that at least there is one valid output of SignTranscript that contains σ^{ring*} is at most $2^{-k}Q$.

Hence when $\mathcal{F}$ outputs a forgery, $\sigma^{standard*}$ must be a forgery with respect to (Sig^*, Ver^*) on message $\tilde{m} = m^* \| \sigma^{ring*} \| Enc_B(r)^*$ with exceptional probability of at most $2^{-k}Q$. If the advantage of $\mathcal{F}$ in Game Unforgeability is ε, the probability that $\mathcal{S}$ existentially forges a signature with respect to (Sig^*, Ver^*) is at least $\varepsilon' = (1-2^{-k}Q)\varepsilon$. Similar to the proof of Lemma 1, the running time of $\mathcal{S}$ is at most $t' = t + Qt_q + c$. $\Box$

A.3 Proof of Theorem 2

Proof. We show that if there exists a distinguisher $\mathcal{D}$ with advantage ε in Game Invisibility, then we can construct a distinguisher $\mathcal{D}^{Enc}$ for the encryption algorithm (Enc, Dec) of the VD scheme with advantage ε' which is a polynomial in ε.

To simulate Game Invisibility, $\mathcal{D}^{Enc}$ carries out similar simulations to that described in the proof of Lemma 1. $\mathcal{S}$ sets the public key of nominee B to $pk_B = (f_B, Ver^*, Enc)$ and private key to $sk_B = (f_B^{-1}, Sig^*, \perp)$.

For a Confirmation/disavowal query with B as the nominee, although $\mathcal{D}^{Enc}$ does not have Dec_B, $\mathcal{D}^{Enc}$ can carry out the confirmation/disavowal protocols as $\mathcal{D}^{Enc}$ is always the one who generates the querying nominative signature (regardless its validity). This is because of the security of the underlying signature scheme. Since $\mathcal{D}$ does not get access to Sig^*, under the security of the signature scheme, the challenging nominative signature must have the third component generated by $\mathcal{D}^{Enc}$. In this case, it is also $\mathcal{D}^{Enc}$ who prepares the second component. Therefore, $\mathcal{D}^{Enc}$ can always carry out the confirmation/disavowal protocols.

For an OracleConvert query on input (m, σ, pk_1, pk_2), $\mathcal{D}^{Enc}$ simulates it according to Convert but with one exception. If $pk_2 = pk_B$, that is, the nominee of the query is indicated as B, $\mathcal{D}^{Enc}$ does not know Dec. Similar to the above, it must be $\mathcal{D}^{Enc}$ who generates σ, due to the unforgeability of Sig^*. The simulator maintains a list L containing pairs of (σ, r) where $R_B = g^r$, $r \in_R \mathbb{Z}_p$. When $\mathcal{D}^{Enc}$ receives a Convert query, it searches L and locates the corresponding r. The output will then be $\sigma^{pub} = (\sigma, r)$.

At some point in the attacking phase, $\mathcal{D}$ outputs a message m^* and requests a challenge nominative signature σ^* on m^*. Let r_0, r_1 selected by $\mathcal{D}^{Enc}$ be the challenge messages and $Enc_B(r_b)$ for $b \in \{1,0\}$ is the return value of the encryption oracle for $\mathcal{D}^{Enc}$. The challenge σ^* is generated based on the outcome of a hidden coin toss b'. If $b' = 1$, σ^* is generated by running SigGen using $Enc_B(r_b)$ and r_1. If $b' = 0$, σ^* is generated by running SigGen using $Enc_B(r_b)$ and r_0.

At the end of the simulation, there are two cases:

- If $b' = 0$, if $\mathcal{D}$ outputs 0, then $\mathcal{D}^{Enc}$ outputs 0, otherwise $\mathcal{D}^{Enc}$ outputs 1.
- If $b' = 1$, if $\mathcal{D}$ outputs 1, then $\mathcal{D}^{Enc}$ outputs 1 also, otherwise $\mathcal{D}^{Enc}$ outputs 0.

If $\mathcal{D}$ has advantage ε, then $\mathcal{D}^{Enc}$ will have advantage $\varepsilon' = \varepsilon$. Similar to Lemma 1, the running time of $\mathcal{D}^{Enc}$ will be at most $t' = t + Qt_q + c$. $\Box$

A.4 Proof of Theorem 3

Proof. By Theorem 1, the scheme is unforgeable with respect to Def. 1 if there exist trapdoor one-way permutation and standard signature scheme which is existentially unforgeable against chosen message attacks. In Game Nominee-only Conversion, adversary $\mathcal{C}$ can corrupt A's private key but not B's private key. Hence if $\mathcal{C}$ wins and outputs a triple $(m^*, \sigma^*, \tilde{\sigma}^{pub})$ such that valid $\leftarrow \mathsf{Ver}^{\mathsf{nominee}}(m^*, \sigma^*, pk_A, sk_B)$ and valid $\leftarrow \mathsf{Ver}^{\mathsf{public}}(m^*, \tilde{\sigma}^{pub}, pk_A, pk_B)$, σ^* must be generated by the game simulator via a SignTranscript query rather than by $\mathcal{C}$ with negligible exceptional probability. The game simulation is the same as that in the proof of Theorem 2.

We now show that if there exists a (t, ε, Q)-adversary $\mathcal{C}$ in Game Nominee-Only conversion, then there exists a (t', ε')-distinguisher $\mathcal{D}^{Enc}$ which has advantage at least $\varepsilon' = \varepsilon$ to launch an adaptive chosen ciphertext attack to the underlying encryption scheme by 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.

Let r_0, r_1 be the challenge message selected by $\mathcal{D}^{Enc}$ and $Enc_B(r_b)$, for $b \in \{1,0\}$, is the return value of the encryption oracle. $\mathcal{D}^{Enc}$ randomly picks a query to SignTranscript and uses r_i where $i \in_R \{1,0\}$ and $Enc_B(r_b)$ for generating $\tilde{\sigma}$. Let $\mathbf{E}$ be the event that $\mathcal{D}^{Enc}$ does not abort when $\mathcal{C}$ outputs $(m^*, \sigma^*, \tilde{\sigma}^{pub})$ where $\tilde{\sigma} = \sigma^*$. Obviously, $\Pr[\mathbf{E}]$ is at least $1/Q$. For event $\mathbf{E}$, if the probability that $\mathcal{C}$ wins in Game Nominee-only conversion is ε, $\mathcal{D}^{Enc}$ will win with probability $\frac{\varepsilon}{2}$. For event $\overline{\mathbf{E}}$, the probability that $\mathcal{D}^{Enc}$ wins is $\frac{1}{2}$ only since $\mathcal{D}^{Enc}$ has to make the guess. Therefore, the probability that $\mathcal{D}^{Enc}$ wins is equal to $\Pr[\mathbf{E}](\frac{\varepsilon}{2}) + \Pr[\overline{\mathbf{E}}]\frac{1}{2}$. Since $\Pr[\mathbf{E}]$ is *at least* $1/Q$, the winning probability of $\mathcal{D}^{Enc}$ is at least $\frac{\varepsilon}{2Q} + \frac{1}{2}$. Similar to Lemma 2, the running time of $\mathcal{D}^{Enc}$ is at most $t + Qt_q + c$. $\Box$