Construction for a Nominative Signature Scheme from Lattice with Enhanced Security

Meenakshi Kansal(✉), Ratna Dutta, and Sourav Mukhopadhyay

Department of Mathematics, Indian Institute of Technology Kharagpur, Kharagpur, India
{kansal,ratna,sourav}@maths.iitkgp.ernet.in

Abstract. The existing secure nominative signature schemes are all based on bilinear pairings and are secure only on classical machines. In this paper, we present the *first* lattice based nominative signature scheme. The security of our scheme relies on the hardness of short integer solution (SIS) and learning with error (LWE) problems for which no polynomial time quantum algorithms exist till now. Consequently, our scheme is the *first* nominative signature scheme that withstand quantum attacks. Furthermore, we propose *stronger* security models for unforgeability and invisibility and prove our construction achieve these enhanced security. Besides, our scheme exhibits impersonation and non-repudiation following standard security model. We emphasis that the security analysis against all the security attributes for our scheme are in standard model except the security against malicious nominator which uses random oracle.

Keywords: Lattice based cryptography · Nominative signature · Unforgeability · Invisibility · Non-repudiation

1 Introduction

A nominative signature scheme, introduced by Kim et al. [6], is an important cryptographic primitive which enables a nominator to select a nominee and produce a nominative signature corresponding to the nominee. Moreover, *only* the nominee can prove (convince) the validity of a nominative signature to a verifier. A nominative signature should satisfy the following security attributes – unforgeability, invisibility, non-impersonation and non-repudiation. Unforgeability ensures that a nominator or a nominee cannot produce a valid nominative signature alone while invisibility features that the verification of a nominative signature can be performed by nominee only. Non-impersonation guarantees that only the nominee can prove (convince) the validity of a nominative signature to a verifier. Non-repudiation holds certain control on the nominee. It ensures that inspite of having the ability of verification and checking validity of a nominative

C. Carlet et al. (Eds.): C2SI 2019, LNCS 11445, pp. 72–91, 2019.
https://doi.org/10.1007/978-3-030-16458-4_6

signature, the nominee cannot deceive a verifier by proving the validity of an invalid nominative signature or invalidity of a valid nominative signature.

Nominative signature has several practical applications in user identification system, banking, insurance, mobile communication etc. For instance, suppose a government body (nominator) certifies and issues signature on passport of a countryman (nominee) who requests for it. Nominative signature scheme can be an ideal cryptographic primitive to handle this situation by producing a mutual agreement (nominative signature) between the government body and the countryman. The government body cannot make false claims on the countryman and vice versa if the scheme is unforgeable. The invisibility property of the scheme permits only the countryman to verify whether the issued passport contains all the correct details or not. Impersonation allows only the countryman to prove (convince) to the airport authority (verifier) that the passport belongs to him/her. Moreover, if the countryman has a fake (or an original) passport then he should not be able to mislead the airport authority by proving the fake passport to be an original (or an original passport to be a fake) passport. This feature is ensured by non-repudiation.

Related Work: Nominative signature was introduced by Kim et al. [6] in 1996 based on Schnorr's signature and claimed to be secure under the hardness of discrete logarithm problem. However, the scheme is found flawed by Huang and Wang [5] in 2004. The concept of convertible nominative signature was introduced in [5]. They also proposed a construction for convertible nominative signature which is proven to be insecure in [15].

The formal definition and security model for a nominative signature was introduced by Liu et al. [10] in 2007 along with a nominative signature scheme. This construction is based on Chaum's undeniable signature and is secure under the hardness of computational Diffie-Hellman problem, decisional Diffie-Hellman problem and discrete logarithm problem. The scheme requires multi-round of communication between a nominator and a nominee for the signature generation. A more efficient design for nominative signature was proposed by Liu et al. [9] using ring signature with one round of communication between a nominator and a nominee. This construction is proven to be secure under the discrete logarithm assumption and computational Diffie-Hellman assumption. However, the schemes [9,10] exhibit the weak invisibility in the sense that the nominator does not take part in generating of some valid signatures.

Huang et al. [4] proposed a stronger security model by introducing stronger invisibility with an extra feature of considering nominator as an adversary. They designed a one-round nominative signature scheme which achieves security in this stronger security model. They proposed a security model stronger that of [4] by proposing a stronger unforgeability where adversary generates the challenge public nominee key. Together with the model of stronger unforgeability they have constructed a nominative signature scheme which is proven to be secure in this stronger security model.

The works of [4,14] are the only secure nominative signature schemes so far on the classical machine. Both these schemes use bilinear pairing. The scheme in [4]

uses witness indistinguishable and is proven secure in the random oracle model under the hardness of weak discrete logarithm problem, weak Diffie-Hellman problem, bilinear Diffie-Hellman exponent problem, weak computational Diffie-Hellman (WCDH)-I problem and WCDH-II problem. It is an efficient scheme as it requires only one-round of communication between a nominator and a nominee to generate a nominative signature. The number of bilinear pairings used in the generation of the nominative signature is 3. The nominator's public-secret keys, nominee's public-secret keys all have size $|\mathbb{G}|$, nominative signature is of size $4|\mathbb{G}|$, communication cost is $|\mathbb{G}|$. On the other hand, the scheme in [14] uses zero knowledge proof of knowledge and is proven secure in the standard model under the hardness of discrete logarithm problem and discrete linear problem. It also requires only one-round of communication between a nominator and a nominee to generate a nominative signature and it uses 3 bilinear pairings in the generation of the nominative signature. The nominator's public-secret keys, nominee's public-secret keys are of size $(n+3)|\mathbb{G}|$, $|\mathbb{G}|$, $(n+6)|\mathbb{G}|$ and $(n+3)|\mathbb{G}|$ respectively, nominative signature is of size $4|\mathbb{G}|$, communication cost is $2|\mathbb{G}|$ where $|\mathbb{G}|$ is the bit size of an element of the group $\mathbb{G}$.

As there is a threat on the reality of quantum machine, a modern public key cryptosystem is required to withstand quantum attacks. Cryptosystems based on hash functions, lattices, codes, multivariate polynomials, isogenies etc are secure on the quantum machine. Lattice based cryptography is one of the most promising tools for the post quantum era as it offers security under worst-case intractability assumptions, efficient parallel computations and homomorphic computation in addition to the apparent resistance to quantum attacks. Although a number of cryptographic primitives have been designed using lattice, till now there are no lattice-based construction for nominative signature.

Our Contribution: In this paper, we propose a security model for the nominative signature scheme which is stronger than the models proposed in [4,14]. Further, we construct the *first lattice based* nominative signature scheme which achieves security in this stronger security model under the hardness of short integer solution (SIS) problem [1] and learning with error (LWE) problem [13]. More precisely, we note the following:

– At a high level, we design a nominative signature by employing the decomposition extension technique of Ling et al. [8] and integrate the non-interactive zero knowledge argument system of Libert et al. [7]. In our construction, the public key of a nominator or a nominee is a matrix $\mathbf{S} \in \mathbb{Z}_q^{n\times m}$ and the secret key $\mathsf{T}_\mathbf{S} \in \mathbb{Z}^{m\times m}$ is a short basis of the lattice $\Lambda_q^{\perp}(\mathbf{S}) = \{\mathbf{x} : \mathbf{S}\mathbf{x} = \mathbf{0} \bmod q\}$ where q, n, m are integers and $m = poly(n)$. The nominator can choose a nominee. The nominee, in turn, proves its identity to the nominator by issuing a signature Sig to the nominator which contains a non-interactive zero knowledge proof Π. The proof Π proves to the nominator that the nominee has the knowledge of a vector $\mathbf{y} \in \mathbb{Z}_q^m$ satisfying an equation of the form $\mathbf{P}\mathbf{y} = \mathbf{v} \bmod q$. Here $\mathbf{P} \in \mathbb{Z}_q^{n\times m}$ and $\mathbf{v} \in \mathbb{Z}_q^n$ are suitably formed using the decomposition-extension technique and are publicly computable. After suitably verifying Sig, the nominator issues the nominative signature nsig which

consists of a short solution $\mathbf{x}$ of an equation of the form $\mathbf{Ax} = \mathbf{b} \bmod q$ where $\mathbf{A}$, $\mathbf{b}$ are publicly available. The nominative signature nsig can be verified by the nominee and the validity (or invalidity) can be proven to the verifier only by the nominee using the confirmation protocol (or disavowal protocol).

- We propose a security model which is stronger than the security of [4,14] by exhibiting stronger unforgeability against malicious nominee, stronger unforgeability against malicious nominator together with stronger invisibility. Similar to [14], the security against impersonation in our model is included in the unforgeability against malicious nominator while non-repudiation follows the model of [4]. The unforgeability in our model is stronger in the following sense.
  (i) The forger is allowed to query for the signature on the forged tuple (M^*, NE, NR) only ones. Here M^* is the message on which the forged nominative signature nsig^* is produced, NE is the malicious (or uncorrupted) nominee and NR is the uncorrupted (or malicious) nominator corresponding to nsig^* for unforgeability against malicious nominee (or against malicious nominator). This query is not permitted in the security model of [4,14].
  (ii) Besides, the forger is provided the flexibility to choose the honest nominator NR (or the honest nominee NE) from all the uncorrupted nominators (or nominees) to achieve unforgeability against malicious nominees (or malicious nominators). In [4,14], honest nominee or honest nominator are chosen by the challenger.
  (iii) Additionally, the forger can corrupt nominator and nominees by querying their secret keys which is not permitted in the security model of [14].
  (iv) Furthermore, similar to [14], the forger is allowed to query for a proof for the validity or invalidity of the signature Sig (or nsig) issued by a honest nominee (or a honest nominator).
- Like [4,14] our scheme also offers non-transferability which ensures that the verifier cannot convince (or disavow) a third party that the verifier received a valid (or invalid) signature on a given message from the nominee. It follows from the combination of invisibility and zero knowledge argument system.
- We also achieve a stronger invisibility as our model gives the choice to the adversary to choose the honest nominee for the challenge query which is not permitted in [4,14].
- Our scheme is proven to be secure in this stronger security model. We achieve unforgeability against malicious nominee under the hardness of SIS search problem. The invisibility is obtained under the hardness of SIS decision problem and LWE problem. Non-repudiation follows from the completeness and soundness properties of the non-interactive zero knowledge argument system of [7]. Our security analysis is in the standard model without using any random oracles. However, we attain unforgeability against malicious nominator in the random oracle model under the hardness of SIS search problem. As mentioned earlier, we cover non-impersonation in the unforgeability against malicious nominator.

– In our scheme, the public key of a user (nominator or nominee) is a matrix $\mathbf{S} \in \mathbb{Z}_q^{n \times m}$ and the secret key $\mathsf{T_S} \in \mathbb{Z}^{m \times m}$ is a short basis of the lattice $\Lambda_q^{\perp}(\mathbf{S}) = \{\mathbf{x} : \mathbf{Sx} = \mathbf{0} \bmod q\}$ where q, n, m are integers and $m = poly(n)$. Consequently, the size of user's public key and secret key is $\widetilde{\mathcal{O}}(n^2)$ each. On the other hand, the nominative signature in our scheme is $\mathsf{nsig_{NR}} = (\mathbf{z}, \mathbf{y}_1)$ where $\mathbf{z} \in \mathbb{Z}_q^m$ and $\mathbf{y}_1 \in \mathbb{Z}_q^n$. Therefore the size of the nominative signature is $\widetilde{\mathcal{O}}(n)$. The Sig issued by the nominee to prove his identity to the nominator is $\mathsf{Sig} = (\Pi,\ \mathbf{y}_1)$ where $\Pi = (\{\mathsf{COM}\}_{\gamma=1}^{s}, \mathsf{Ch}, \{\mathsf{RSP}\}_{\gamma=1}^{s})$ is the proof of knowledge of a vector $\mathbf{y} \in \mathbb{Z}_q^m$ satisfying an equation of the form $\mathbf{Py} = \mathbf{v} \bmod q$. This implies that the communication cost for the non-interactive zero knowledge proof is $s \cdot |\mathsf{COM}| + s \cdot |\mathsf{RSP}| + s$. Here COM is the commitment function used by the nominee to produce a commitment about the knowledge of $\mathbf{y}$ to the nominator and RSP is the response on this commitment COM depending on the challenge Ch and $|\mathsf{RSP}| = \mathcal{O}(L)$, $L = 6(m+1)p$, $p = \lfloor \log_2 \beta \rfloor + 1$, $\beta = 2\sigma\sqrt{m}$ and σ is the standard deviation of the discrete Gaussian distribution.

2 Preliminaries

Notations. Here we define some basic terminology for our work. Throughout this paper, a vector $\mathbf{a} \in \mathbb{S}^n$ denotes a column vector of dimension $n \times 1$ with entries from the set $\mathbb{S}$. For $\mathbf{u} = (u_1, u_2, \ldots, u_n) \in \mathbb{R}^n$, $||\mathbf{u}||_\infty = \max_i |u_i|$ denotes the maximum norm and $||\mathbf{u}|| = \sqrt{u_1^2 + u_2^2 + \ldots + u_n^2}$ stands for the Euclidean norm. Let $\mathbf{A} = (\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_m)$ be a matrix with m columns in $\mathbb{R}^n$ then $||\mathbf{A}|| = \max_{1 \le i \le m} ||\mathbf{a}_i||$. The notation $\mathbf{A} \hookleftarrow \Delta$ implies $\mathbf{A}$ is a matrix following the distribution Δ and $\mathbf{A}^t$ represents the transpose of the matrix $\mathbf{A}$. We refer $||$ for the concatenation of matrices and also for the concatenation of vectors. We say that a function f is negligible in λ if $f = \lambda^{-\omega(1)}$.

Definition 1 (Lattice). *For any $m \ge n$, let $\boldsymbol{B} = \{\boldsymbol{b}_1, \boldsymbol{b}_2, \cdots, \boldsymbol{b}_m\}$ be any linearly independent set of vectors in $\mathbb{R}^n$. A lattice generated by the set $\boldsymbol{B}$ is defined as $\Lambda(\boldsymbol{B}) = \{\sum_{\boldsymbol{b}_i \in \boldsymbol{B}} c_i \boldsymbol{b}_i : c_i \in \mathbb{Z}\}$ with basis $\boldsymbol{B}$.*

For $q \in \mathbb{N}$, matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$ and vector $\mathbf{u} \in \mathbb{Z}_q^n$, we define the following three q-ary lattices generated by $\mathbf{A}$: $\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{Ax} = \mathbf{0} \bmod q\}$, $\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{Ax} = \mathbf{u} \bmod q\}$, $\Lambda_q(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A}^t\mathbf{s} = \mathbf{x} \bmod q, \text{for some } \mathbf{s} \in \mathbb{Z}_q^n\}$, where m, n are integers with $m \ge n \ge 1$ and $\mathbf{0}$ is a zero vector of size $n \times 1$.

Definition 2 (Gaussian distribution over a lattice). *For a lattice Λ and a real number $\sigma > 0$, discrete Gaussian distribution over Λ centered at $\boldsymbol{0}$, denoted by $D_{\Lambda,\sigma}$, is defined as: $\forall \boldsymbol{y} \in \Lambda$, $D_{\Lambda,\sigma}[\boldsymbol{y}] \sim exp(-\pi||\boldsymbol{y}||^2/\sigma^2)$, i.e. $D_{\Lambda,\sigma}[\boldsymbol{y}]$ is proportional to $exp(-\pi||\boldsymbol{y}||^2/\sigma^2)$ where $D_{\Lambda,\sigma}[\boldsymbol{y}]$ means the vector $\boldsymbol{y} \hookleftarrow D_{\Lambda,\sigma}$. We say that $D_{\Lambda,\sigma}$ is a distribution with standard deviation σ.*

Lemma 1. *For any n-dimensional lattice Λ and for any real number $\sigma > 0$, we have the following results and probabilistic polynomial time (PPT) algorithms:*

(i) $Pr_{\boldsymbol{b} \hookleftarrow D_{\Lambda,\sigma}}[||\boldsymbol{b}|| \leq \sigma\sqrt{n}] \geq 1 - 2^{-\Omega(n)}$, *i.e. if* $\boldsymbol{b} \hookleftarrow D_{\Lambda,\sigma}$ *then* $||\boldsymbol{b}|| \leq \sigma\sqrt{n}$ *with overwhelming probability.*

(ii) $\mathsf{TrapGen}(n, m, q) \longrightarrow (\boldsymbol{A}, \mathsf{T}_{\boldsymbol{A}})$ *[2]. This randomized algorithm outputs a matrix* $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ *and a short basis* $\mathsf{T}_{\boldsymbol{A}} \in \mathbb{Z}^{m\times m}$ *of* $\Lambda_q^{\perp}(\boldsymbol{A})$ *such that* $\boldsymbol{A}$ *is within the statistical distance* $2^{-\Omega(n)}$ *to* $U(\mathbb{Z}_q^{n\times m})$ *and* $||\widetilde{\mathsf{T}}_{\boldsymbol{A}}|| \leq \mathcal{O}(\sqrt{n \log q})$. *Here* $U(\mathbb{Z}_q^{n\times m})$ *is the uniform distribution of integer matrices over* $\mathbb{Z}_q$ *of order* $n \times m$ *and* $\widetilde{\mathsf{T}}_{\boldsymbol{A}}$ *is the Gram-Schmidt orthogonalization of* $\mathsf{T}_{\boldsymbol{A}}$.

(iii) $\mathsf{SampleD}(\mathsf{T}_{\boldsymbol{A}}, \boldsymbol{A}, \boldsymbol{u}, \sigma) \longrightarrow (\boldsymbol{x})$ *[12]. Given a matrix* $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ *whose columns span* $\mathbb{Z}_q^n$, *a basis* $\mathsf{T}_{\boldsymbol{A}} \in \mathbb{Z}^{m\times m}$ *of* $\Lambda_q^u(\boldsymbol{A})$, *a vector* $\boldsymbol{u} \in \mathbb{Z}_q^n$ *and a real number* σ, *this randomized algorithm returns a vector* $\boldsymbol{x} \in \mathbb{Z}^m$ *from the distribution* $D_{\mathbb{Z}^m,\sigma}$ (i.e., $||\mathbf{x}|| \leq \sigma\sqrt{m}$ by (i)) *satisfying* $\boldsymbol{A} \cdot \boldsymbol{x} = \boldsymbol{u} \bmod q$.

2.1 Computational and Decisional Problems

Definition 3 (Inhomogeneous short integer solution (ISIS) search problem) [1]**.** *Given an integer q, a real number* β, *a matrix* $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ *and a vector* $\boldsymbol{u} \in \mathbb{Z}_q^n$, *the* ISIS *problem is to find an integer vector* $\boldsymbol{e} \in \mathbb{Z}^m$ *such that* $\boldsymbol{A}\boldsymbol{e} = \boldsymbol{u} \bmod q$ *with* $||\boldsymbol{e}|| \leq \beta$ *with non-negligible probability. If* $\boldsymbol{u} = \boldsymbol{0} \in \mathbb{Z}_q^n$, *then it is known as* short integer solution (SIS) *problem.*

Definition 4 (Short integer solution (SIS) decision problem) [11]**.** *Let* χ *be a distribution over* $\mathbb{Z}_q$ *having samples of the form* $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s}) \in \mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^{n\times 1}$ *with standard deviation* σ *where* $\boldsymbol{A} \in \mathbb{Z}_q^{n\times m}$ *is a matrix and* $\boldsymbol{s} \in \mathbb{Z}_q^{m\times 1}$ *is a vector with* $||\boldsymbol{s}|| \leq \sigma\sqrt{m}$. *The* decisional SIS *is to decide whether* $(\boldsymbol{A}, \boldsymbol{A}\boldsymbol{s})$ *follows* χ *distribution or uniform distribution* $U(\mathbb{Z}_q^{n\times m} \times \mathbb{Z}_q^n)$ *with non-negligible probability.*

Definition 5 (Learning with errors (LWE) problem) [13]**.** *Let* $n \geq 1$ *be any integer,* $p \geq 2$ *be any prime and* χ *be a distribution on* $\mathbb{Z}$. *For any fixed vector* $\boldsymbol{s} \in \mathbb{Z}_p^n$, *given arbitrarily many samples of the form* $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s}\rangle + e)$ *with* $\boldsymbol{a}$ *uniform in* $\mathbb{Z}_p^n$ *and e sampled from* χ, *the problem of finding* $\boldsymbol{s}$ *is called the* search LWE *and the problem of distinguishing the distribution of* $(\boldsymbol{a}, \langle \boldsymbol{a}, \boldsymbol{s}\rangle + e)$ *from the uniform distribution* $U(\mathbb{Z}_p^n \times \mathbb{Z}_p)$ *is called the* decisional LWE. *Here* $\langle \boldsymbol{a}, \boldsymbol{s}\rangle = \boldsymbol{a}^t\boldsymbol{s}$.

2.2 Zero Knowledge Argument System [7]

This section deals with the zero knowledge argument system when the prover wants to prove the knowledge of the witness $\mathbf{x}$ satisfying the relation $\mathbf{P}\mathbf{x} = \mathbf{v}$ without giving $\mathbf{x}$ to the verifier. Here $\mathbf{P}$ is any matrix and $\mathbf{v}$ is a vector (or matrix), both publicly available and $\mathbf{x}$ is prover's secret vector (or matrix) with some conditions to be proven in zero knowledge to the verifier.

Let $q \geq 2$ be any integer and D, L be two positive integers. We consider a set $\mathsf{VALID} \subseteq \{-1, 0, 1\}^L$. Similar to Libert et al. [7], let S be any finite set of permutations such that for any $\pi \in S$, one can associate a permutation T_π of L elements satisfying

(i) $\mathbf{x} \in \mathsf{VALID} \Leftrightarrow T_\pi(\mathbf{x}) \in \mathsf{VALID}$.
(ii) If $\mathbf{x} \in \mathsf{VALID}$ and π is uniform in S then $T_\pi(\mathbf{x})$ is uniform in VALID.

A zero knowledge argument of knowledge (ZKAoK) for the relation $\mathcal{R} = \{(\mathbf{P}, \mathbf{v}) \in \mathbb{Z}_q^{D \times L} \times \mathbb{Z}_q^D, \mathbf{x} \in \mathsf{VALID} : \mathbf{Px} = \mathbf{v} \bmod q\}$ written as $((\mathbf{P}, \mathbf{v}), \mathbf{x}) \in \mathcal{R}$ is a 3-round protocol $\mathsf{ZKAoK} = (\mathsf{Commitment}, \mathsf{Challenge}, \mathsf{Response}, \mathsf{Verification})$ between a prover and a verifier, both having access to $\mathbf{P}$ and $\mathbf{v}$ where $\mathsf{ZKAoK.Commitment}$, $\mathsf{ZKAoK.Challenge}$, $\mathsf{ZKAoK.Response}$ are PPT algorithms and $\mathsf{ZKAoK.Verification}$ is a deterministic algorithm with the following requirements:

1. $\mathsf{ZKAoK.Commitment}(\mathbf{P}, \mathbf{v}, \mathbf{x}) \longrightarrow (\mathsf{COM} = (C_1, C_2, C_3))$. The prover does the following:
   (a) It samples randomness ρ_1, ρ_2, ρ_3 for generating commitments and selects $\mathbf{r} \hookleftarrow U(\mathbb{Z}_q^L), \pi \hookleftarrow U(S)$ where S is a finite set of permutation.
   (b) It computes the commitment $\mathsf{COM} = (C_1, C_2, C_3)$ where $C_1 = \mathsf{CMT}_1(\pi, \mathbf{Pr}; \rho_1)$, $C_2 = \mathsf{CMT}_2(T_\pi(\mathbf{r}); \rho_2)$, $C_3 = \mathsf{CMT}_3(T_\pi(\mathbf{x} + \mathbf{r}); \rho_3)$ are generated using randomness ρ_1, ρ_2, ρ_3 respectively and the permutation T_π corresponding to π. Here CMT_i, $i = 1, 2, 3$, is statistically hiding and computationally binding commitment scheme such that the hiding property holds even against all-powerful receivers while the binding property holds only for polynomially bounded senders.
   (c) Finally, the prover sends the commitment COM to the verifier.
2. $\mathsf{ZKAoK.Challenge}(\mathbf{P}, \mathbf{v}) \longrightarrow (\mathsf{Ch} \hookleftarrow U(\{1, 2, 3\}))$. The verifier sends a challenge $\mathsf{Ch} \hookleftarrow U(\{1, 2, 3\})$ to the prover.
3. $\mathsf{ZKAoK.Response}(\mathsf{Ch}, \rho_1, \rho_2, \rho_3, \pi, \mathbf{r}, \mathbf{x}) \longrightarrow (\mathsf{RSP})$. The prover sends a response RSP computed as follows:
   (a) If $\mathsf{Ch} = 1$ then the prover sets $\mathbf{t_x} = T_\pi(\mathbf{x})$, $\mathbf{t_r} = T_\pi(\mathbf{r})$ and $\mathsf{RSP} = (\mathbf{t_x}, \mathbf{t_r}, \rho_2, \rho_3)$ using T_π associated with π.
   (b) If $\mathsf{Ch} = 2$ then the prover sets $\pi_2 = \pi$, $\mathbf{y} = \mathbf{x} + \mathbf{r}$ and $\mathsf{RSP} = (\pi_2, \mathbf{y}, \rho_1, \rho_3)$.
   (c) If $\mathsf{Ch} = 3$ then the prover sets $\pi_3 = \pi$, $\mathbf{r}_3 = \mathbf{r}$ and $\mathsf{RSP} = (\pi_3, \mathbf{r}_3, \rho_1, \rho_2)$.
4. $\mathsf{ZKAoK.Verification}(\mathbf{P}, \mathbf{v}, \mathsf{RSP}, \mathsf{Ch}, \mathsf{COM}) \longrightarrow (\mathsf{VRF})$. On receiving the response RSP from the prover, the verifier uses the commitment scheme CMT_i, $i = 1, 2, 3$ and proceeds as follows:
   (a) If $\mathsf{Ch} = 1$ then the verifier checks whether $\mathbf{t_x} \in \mathsf{VALID}$ and $C_2 = \mathsf{CMT}_2(\mathbf{t_r}; \rho_2)$, $C_3 = \mathsf{CMT}_3(\mathbf{t_x} + \mathbf{t_r}; \rho_3)$ using $\mathsf{RSP} = (\mathbf{t_x}, \mathbf{t_r}, \rho_2, \rho_3)$ and extracting C_2, C_3 from COM.
   (b) If $\mathsf{Ch} = 2$ then the verifier checks whether $C_1 = \mathsf{CMT}_1(\pi_2, \mathbf{Py}\text{-}\mathbf{v}; \rho_1)$ and $C_3 = \mathsf{CMT}_3(T_{\pi_2}(\mathbf{y}); \rho_3)$ extracting C_1, C_3 from COM and using $\mathsf{RSP} = (\pi_2, \mathbf{y}, \rho_1, \rho_3)$ together with the permutation T_{π_2} associated with π_2.
   (c) If $\mathsf{Ch} = 3$ then the verifier checks whether $C_1 = \mathsf{CMT}_1(\pi_3, \mathbf{Pr}_3; \rho_1), C_2 = \mathsf{CMT}_2(T_{\pi_3}(\mathbf{r}_3); \rho_2)$ using C_1, C_2 obtained from COM, $\mathsf{RSP} = (\pi_3, \mathbf{r}_3, \rho_1, \rho_3)$ and permutation T_{π_3} corresponding to π_3.

   In each case, the verifier outputs $\mathsf{VRF} = 1$ if the verification succeeds; otherwise $\mathsf{VRF} = 0$.

The above zero knowledge argument protocol satisfies the following three properties [7]:

Let $\mathsf{VRF} \leftarrow \mathsf{ZKAoK.Verification}\ (\mathbf{P}, \mathbf{v}, \mathsf{RSP}, \mathsf{COM})$, $\mathsf{RSP} \leftarrow \mathsf{ZKAoK.Response}\ (\mathsf{Ch}, \rho_1, \rho_2, \rho_3, \pi, \mathbf{r}, \mathbf{x})$, $\mathsf{COM} \leftarrow \mathsf{ZKAoK.Commitment}\ (\mathbf{P}, \mathbf{v}, \mathbf{x})$ and $\mathsf{Ch} \leftarrow \mathsf{ZKAoK.Challenge}\ (\mathbf{P}, \mathbf{v})$ where $\rho_1, \rho_2, \rho_3, \mathbf{r}, \pi$ are as selected in algorithm $\mathsf{ZKAoK.Commitment}(\mathbf{P}, \mathbf{v}, \mathbf{x})$ by the prover.

Correctness: If $((\mathbf{P}, \mathbf{v}), \mathbf{x}) \in \mathcal{R}$ then $Pr[\mathsf{VRF} = 1] = 1$.

Soundness: If $((\mathbf{P}, \mathbf{v}), \mathbf{x}) \notin \mathcal{R}$ then $Pr[\mathsf{VRF} = 1] \leq \mathsf{negl}(\lambda)$ where $\mathsf{negl}(\lambda)$ is a negligible function in λ.

Zero Knowledge: If the statement proven by the prover is true then the cheating verifier learns only the fact that the statement is true.

Remark 1. The above protocol is repeated $s = \omega(\log n)$ times to achieve negligible soundness error and can be made non-interactive using Fiat-Shamir heuristic [3] as a triple $\Pi = (\{\mathsf{COM}_\gamma\}_{\gamma=1}^{s}, \mathsf{Ch}, \{\mathsf{RSP}_\gamma\}_{\gamma=1}^{s})$ where $\mathsf{Ch} = H(M, \{\mathsf{COM}_\gamma\}_{\gamma=1}^{s}, \mathsf{aux}) \in \{1,2,3\}^s$, M is a message, aux is some auxiliary information and $H : \{0,1\}^* \rightarrow \{1,2,3\}^s$ is a cryptographically secure hash function. The prover sends s commitments COM_γ, $\gamma = 1, 2, \ldots, s$ to the verifier who in turn sends the challenge $\mathsf{Ch} = H(M, \{\mathsf{COM}_\gamma\}_{\gamma=1}^{s}, \mathsf{aux}) \in \{1,2,3\}^s$ to the prover treating the hash function H as a random oracle. At the end, the prover outputs response RSP_γ generated by executing $\mathsf{ZKAoK.Response}(\mathsf{Ch}[\gamma], \rho_1^{(\gamma)}, \rho_2^{(\gamma)}, \rho_3^{(\gamma)}, \pi^{(\gamma)}, \mathbf{r}^{(\gamma)}, \mathbf{x})$ where $\mathsf{Ch}[\gamma]$ is the γ-th digit of $\mathsf{Ch} \in \{1,2,3\}^s$ and $\rho_1^{(\gamma)}, \rho_2^{(\gamma)}, \rho_3^{(\gamma)}, \mathbf{r}^{(\gamma)}, \pi^{(\gamma)}$ are as selected by the prover in the γ-th run of the algorithm $\mathsf{ZKAoK.Commitment}(\mathbf{P}, \mathbf{v}, \mathbf{x})$ for $\gamma = 1, 2, \ldots, s$. For the verification, the response RSP_γ corresponding to the γ-th digit of $\mathsf{Ch} \in \{1,2,3\}^s$ is verified following the algorithm $\mathsf{ZKAoK.Verification}(\mathbf{P}, \mathbf{v}, \mathsf{RSP}_\gamma, \mathsf{Ch}[\gamma], \mathsf{COM}_\gamma)$ that generates VRF_γ. If $\mathsf{VRF}_\gamma = 1$ for all $\gamma = 1, 2, \ldots, s$ then this treats Π as a confirmation proof of the above zero knowledge argument system. On the other hand, $\mathsf{VRF}_\gamma = 0$ for atleast one $\gamma = 1, 2, \ldots, s$ considers Π as a disavowal proof for the above zero knowledge argument system.

Theorem 1 [7]**.** *The protocol described above is a statistical* ZKAoK *for the relation $\mathcal{R}$ with soundness error 2/3 and perfect completeness having the communication cost $\mathcal{O}(L \log q)$.*

3 Our Nominative Signature Scheme

Communication Model: Informally speaking, our scheme involves a trusted authority together with nominees and nominators. The trusted authority generates the system parameters, public-secret key pairs of nominees and nominators. System parameters and public keys are made public and secret keys are sent secretly to the concerned parties by the trusted authority.

A nominee issues a signature Sig to the nominator. To generate Sig, the nominee firstly transforms the system of equations involving two equations into an

equation of the form $\mathbf{D}_0\mathbf{x}_0+\mathbf{D}_1\mathbf{x}_1 = \mathbf{v} \bmod q$. Then by using the decomposition-extension technique, the equation $\mathbf{D}_0\mathbf{x}_0 + \mathbf{D}_1\mathbf{x}_1 = \mathbf{v} \bmod q$ is transformed into an equation $\mathbf{P}\mathbf{x} = \mathbf{v} \bmod q$. The decomposition-extension technique helps in converting $[\mathbf{x}_0||\mathbf{x}_1]$ to $\mathbf{x}$ such that $\mathbf{x} \in \mathsf{VALID}$. Further, the nominee proves to the nominator in zero-knowledge the possession of $\mathbf{x} \in \mathsf{VALID}$ satisfying $\mathbf{P}\mathbf{x} = \mathbf{v} \bmod q$.

After receiving the Sig from the nominee, the nominator verifies the validity of Sig and issues the nominative signature nsig. The verification of nsig can be done *only* by the nominee. Our scheme also involves a confirmation or disavowal protocol in which the nominee proves to the verifier in zero-knowledge the validity or the invalidity of the nominative signature nsig issued by the nominator.

Formally, our nominative signature $\mathsf{NS} = \{\mathsf{Setup},\ \mathsf{KeygenNR},\ \mathsf{KeygenNE},\ \mathsf{SignNE},\ \mathsf{SignNR},\ \mathsf{Verify},\ \mathsf{ConfOrDisav} = (\mathsf{TMnominee},\ \mathsf{TMverifer})\}$ works as follows:

NS.Setup$(\lambda) \rightarrow \mathsf{param}$**.** Given a security parameter $\lambda > 0$, the key generation center (KGC) generates an integer n of size $\mathcal{O}(\lambda)$, a prime modulus q of size $\mathcal{O}(n^3)$ and an integer m such that $m = 2n + 8n\lceil \log q \rceil > n\lceil \log q \rceil$. The KGC also chooses a real number σ of size $\Omega(\sqrt{l \log q} \log n)$, an error bound $\beta = 2\sigma\sqrt{m}$ and two cryptographically secure hash functions $H_1 : \{0,1\}^* \rightarrow \mathbb{Z}_q^n$, $H : \{0,1\}^* \rightarrow \{1,2,3\}^s$ where s is of size $\omega(\log n)$. Observe that the size of β is $\sigma\omega(\log m)$. The KGC publishes the system parameters $\mathsf{param} = (n, q, m, \sigma, \beta, H, H_1)$. We use σ for the standard deviation of the discrete Gaussian distribution.

NS.KeygenNR$(\mathsf{param},\ u) \rightarrow (\mathsf{PK}_u,\ \mathsf{SK}_u)$**.** To generate the public-secret key pair of a nominator u, the KGC invokes $\mathsf{TrapGen}(n,\ m,\ q) \rightarrow (\mathbf{A}_u,\ \mathsf{T}_{\mathbf{A}_u})$ described in Lemma 1 in Sect. 2 and sets the public and secret key

$$\mathsf{PK}_u = \mathbf{A}_u,\ \mathsf{SK}_u = \mathsf{T}_{\mathbf{A}_u}$$

for u where $\mathbf{A}_u \in \mathbb{Z}_q^{n \times m}$ and $\mathsf{T}_{\mathbf{A}_u} \in \mathbb{Z}^{m \times m}$. The public key PK_u is made public while the secret key SK_u is sent secretly by the KGC to u.

NS.KeygenNE$(\mathsf{param},\ v) \rightarrow (\mathsf{pk}_v,\ \mathsf{sk}_v)$**.** The KGC runs $\mathsf{TrapGen}(n,\ m,\ q) \rightarrow (\mathbf{B}_v,\ \mathsf{T}_{\mathbf{B}_v})$ (see Lemma 1 in Sect. 2) to produce the public-secret key pair of a nominee v. It sets the public key and secret key

$$\mathsf{pk}_v = \mathbf{B}_v,\ \mathsf{sk}_v = \mathsf{T}_{\mathbf{B}_v}$$

for v where $\mathbf{B}_v \in \mathbb{Z}_q^{n \times m}$ and $\mathsf{T}_{\mathbf{B}_v} \in \mathbb{Z}^{m \times m}$. The KGC makes pk_v publicly available and sends sk_v secretly to v.

NS.SignNE$(\mathsf{param},\ \mathsf{sk}_{\mathsf{NE}},\ \mathsf{pk}_{\mathsf{NE}},\ \mathsf{PK}_{\mathsf{NR}},\ M) \rightarrow (\mathsf{Sig}_{M,\mathsf{NE},\mathsf{NR}} = (\Pi,\ \mathbf{y}_1))$**.** Let M be a message to be signed. A nominee NE performs the following steps using $\mathsf{param} = (n,\ q,\ m,\ \sigma,\ \beta,\ H,\ H_1)$, $\mathsf{sk}_{\mathsf{NE}} = \mathsf{T}_{\mathbf{B}_{\mathsf{NE}}}$, $\mathsf{pk}_{\mathsf{NE}} = \mathbf{B}_{\mathsf{NE}}$ and $\mathsf{PK}_{\mathsf{NR}} = \mathbf{A}_{\mathsf{NR}}$ to generate the signature $\mathsf{Sig}_{M,\mathsf{NE},\mathsf{NR}} = (\Pi, \mathbf{y}_1)$ on M.

(i) The nominee NE computes $\mathbf{y} = H_1(M||\mathbf{A}_{\mathsf{NR}}||\mathbf{B}_{\mathsf{NE}}) \in \mathbb{Z}_q^n$ and generates a short vector $\mathbf{v} \in \mathbb{Z}_q^m$ satisfying $\mathbf{B}_{\mathsf{NE}} \cdot \mathbf{v} = \mathbf{y} \bmod q$ with $||\mathbf{v}|| \leq \sigma\sqrt{m}$ by

running the algorithm $\mathsf{SampleD}(\mathsf{T}_{\mathbf{B}_{\mathsf{NE}}}, \mathbf{B}_{\mathsf{NE}}, \mathbf{y}, \sigma) \rightarrow (\mathbf{v})$ using the short basis $\mathsf{sk}_{\mathsf{NE}} = \mathsf{T}_{\mathbf{B}_{\mathsf{NE}}}$ given in Lemma 1 in Sect. 2. Note that $||\mathbf{v}||_\infty \leq ||\mathbf{v}|| \leq \sigma\sqrt{m} \leq \beta$ as $\beta = 2\sigma\sqrt{m}$.

(ii) The nominee chooses a random number $r_1 \in [-\beta, \beta]$ and sets

$$\mathbf{y}_1 = \mathbf{B}_{\mathsf{NE}}^t \cdot (r_1\mathbf{y}) + \mathbf{v} \bmod q \tag{1}$$

where $\mathbf{B}_{\mathsf{NE}}^t$ is the transpose of the matrix $\mathbf{B}_{\mathsf{NE}}$. Note that, given the values of $(\mathbf{y}_1, \mathbf{B}_{\mathsf{NE}}, \mathbf{y})$ then the problem to find $(r_1, \mathbf{v})$ from Eq. 1 is not feasible under the hardness of LWE.

(iii) The nominee rewrites the system of the equations

$$\mathbf{B}_{\mathsf{NE}} \cdot \mathbf{v} = \mathbf{y} \bmod q,$$

$$\mathbf{y}_1 = \mathbf{B}_{\mathsf{NE}}^t \cdot (r_1\mathbf{y}) + \mathbf{v} \bmod q$$

into a single equation

$$\mathbf{D}_0 \cdot \mathbf{x}_0 + \mathbf{D}_1 \cdot \mathbf{x}_1 = \mathbf{b} \bmod q \tag{2}$$

with

$$\mathbf{D}_0 = \begin{bmatrix} (\mathbf{B}_{\mathsf{NE}})_{n\times m} & \mathbf{0}_{n\times 1} \\ \mathbf{0}_{m\times m} & \mathbf{0}_{m\times 1} \end{bmatrix}, \mathbf{D}_1 = \begin{bmatrix} \mathbf{0}_{n\times 1} & \mathbf{0}_{n\times m} \\ (\mathbf{B}_{\mathsf{NE}}^t \cdot \mathbf{y})_{m\times 1} & \mathbf{I}_{m\times m} \end{bmatrix},$$

$$\mathbf{x}_0 = \begin{bmatrix} \mathbf{v}_{m\times 1} \\ 0_{1\times 1} \end{bmatrix}, \mathbf{x}_1 = \begin{bmatrix} (r_1)_{1\times 1} \\ \mathbf{v}_{m\times 1} \end{bmatrix}, \mathbf{b} = \begin{bmatrix} \mathbf{y}_{n\times 1} \\ (\mathbf{y}_1)_{m\times 1} \end{bmatrix}$$

where $\mathbf{I}_{m\times m}$ is an identity matrix of size m.

(iv) Let $p = \lfloor \log_2 \beta \rfloor + 1$. We define the sets B_{mp}^3, $\mathcal{S}_{3mp}$ as:
$B_{mp}^3 = \{\mathbf{x} \in \{-1, 0, 1\}^{3mp} : \mathbf{x}$ has exactly mp co-ordinates equal to j for $j = -1, 0, 1\}$,
$\mathcal{S}_{3mp} = \{\pi : \pi$ is a permutation on $3mp$ length vectors$\}$.
Then $\widehat{\mathbf{w}} \in B_{mp}^3 \Leftrightarrow \pi(\widehat{\mathbf{w}}) \in B_{mp}^3$ for any permutation $\pi \in \mathcal{S}_{3mp}$.

(v) The Eq. 2 is then converted by the nominee into an equation of the form $\mathbf{Px} = \mathbf{b} \bmod q$ as follows using the algorithm $\mathsf{Dec\text{-}Ext}_{m,p}$ described in Fig. 1 which is the decomposition-extension technique of Ling et al. [8].
Note that $\mathbf{x}_0, \mathbf{x}_1 \in [-\beta, \beta]^{m+1}$. The nominee NE generates
$\widehat{\mathbf{x}}_0 \in B_{(m+1)p}^3 \leftarrow \mathsf{Dec\text{-}Ext}(\mathbf{x}_0)$, $\widehat{\mathbf{x}}_1 \in B_{(m+1)p}^3 \leftarrow \mathsf{Dec\text{-}Ext}(\mathbf{x}_1)$ and sets
$\widehat{\mathbf{D}}_0 = \mathbf{D}_0 \cdot \widehat{\mathbf{K}}_{(m+1),\beta} \bmod q \in \mathbb{Z}_q^{(n+m)\times 3(m+1)p}$,
$\widehat{\mathbf{D}}_1 = \mathbf{D}_1 \cdot \widehat{\mathbf{K}}_{(m+1),\beta} \bmod q \in \mathbb{Z}_q^{(n+m)\times 3(m+1)p}$
where $\widehat{\mathbf{K}}_{m+1,\beta} = [\mathbf{K}_{m+1,\beta} || 0^{m+1\times 2(m+1)p}] \in \mathbb{Z}^{(m+1)\times 3(m+1)p}$,
$\mathbf{K}_{m+1,\beta} = \mathbf{I}_{(m+1)\times(m+1)} \otimes [\beta_1, \beta_2, \ldots, \beta_p]$ and $\widehat{\mathbf{x}}_i \in B_{(m+1)p}^3$ satisfies

$$\widehat{\mathbf{K}}_{(m+1),\beta} \cdot \widehat{\mathbf{x}}_i = \mathbf{x}_i \tag{3}$$

for $i = 0, 1$ (see line 6 in Fig. 1). Next, the nominee sets $\mathbf{P} = [\widehat{\mathbf{D}}_0 || \widehat{\mathbf{D}}_1] \in \mathbb{Z}_q^{D\times L}$, $\mathbf{x} = [\widehat{\mathbf{x}}_0 || \widehat{\mathbf{x}}_1]^t \in \mathbb{Z}_q^L$ where $L = 6(m+1)p$ and $D = n + m$.

Input: $\mathbf{w} = (w_1, w_2, \ldots, w_m) \in [-\beta, \beta]^m$
Output: $\widehat{\mathbf{w}} \in B^3_{mp}$ where $p = \lfloor \log_2 \beta \rfloor + 1$

1. Define a super-decreasing sequence $\{\beta\}_{j=1}^{p}$ of integers by setting $\beta_1 = \lceil \frac{\beta}{2} \rceil$ and $\beta_j = \lceil \frac{\beta - (\beta_1 + \beta_2 + \cdots + \beta_{j-1})}{2} \rceil$ for $2 \leq j \leq p$.
2. **for**$(w_i \in [-\beta, \beta]$ and $i \leq m)$ **do**
   Compute $w_i^{(1)}, w_i^{(2)}, \ldots, w_i^{(p)} \in \{-1, 0, 1\}$ such that $\sum_{j=1}^{p} \beta_j w_i^{(j)} = w_i$.
   end do
3. Set $\mathbf{w}' = (w_1^{(1)}, \ldots, w_1^{(p)}, w_2^{(1)}, \ldots, w_2^{(p)}, \ldots, w_m^{(1)}, \ldots, w_m^{(p)}) \in \{-1, 0, 1\}^{mp}$. Then $\mathbf{w}'$ satisfies $\mathbf{K}_{m,\beta} \cdot \mathbf{w}' = \mathbf{w}$ where
   $\mathbf{K}_{m,\beta} = \mathbf{I}_{m \times m} \otimes (\beta_1, \beta_2, ..., \beta_p) = \begin{bmatrix} \beta_1 \ \beta_2 \ldots \beta_p & & & \\ & \beta_1 \ \beta_2 \ldots \beta_p & & \\ & & \cdots & \\ & & & \beta_1 \ \beta_2 \ldots \beta_p \end{bmatrix}$.
4. Set $\widehat{\mathbf{K}}_{m,\beta} = [\mathbf{K}_{m,\beta} || 0^{m \times 2mp}] \in \mathbb{Z}^{m \times 3mp}$.
5. Select a random vector $\widetilde{\mathbf{w}} \in \{-1, 0, 1\}^{2mp}$ having exactly $(mp - \lambda_0)$ many 0's, $(mp - \lambda_1)$ many 1's and $(\lambda_0 + \lambda_1)$ many -1's where λ_0, λ_1 are respectively the number of 0's and 1's in $\mathbf{w}'$.
6. Set $\widehat{\mathbf{w}} = (\mathbf{w}' || \widetilde{\mathbf{w}}) \in B^3_{mp}$. Then $\widehat{\mathbf{K}}_{m,\beta} \cdot \widehat{\mathbf{w}} = \mathbf{K}_{m,\beta} \cdot \mathbf{w}' = \mathbf{w}$ and $\widehat{\mathbf{w}} \in B^3_{mp} \Leftrightarrow \pi(\widehat{\mathbf{w}}) \in B^3_{mp}$ for any permutation π on $3mp$ length vectors.
7. **return** $\mathbf{w} = (\mathbf{w}' || \widetilde{\mathbf{w}})$.

Fig. 1. Algorithm $\mathsf{Dec\text{-}Ext}_{m,p}(\mathbf{w})$ where $p = \lfloor \log_2 \beta \rfloor + 1$ and $\mathbf{w} \in [-\beta, \beta]^m$.

As $\widehat{\mathbf{x}}_0, \widehat{\mathbf{x}}_1 \in \mathbb{Z}_q^{\frac{L}{2}}$, $\widehat{\mathbf{D}}_0, \widehat{\mathbf{D}}_1 \in \mathbb{Z}_q^{D \times \frac{L}{2}}$, we have

$$\begin{aligned} \mathbf{P}\mathbf{x} &= \widehat{\mathbf{D}}_0 \cdot \widehat{\mathbf{x}}_0 + \widehat{\mathbf{D}}_1 \cdot \widehat{\mathbf{x}}_1 = \mathbf{D}_0 \cdot \widehat{\mathbf{K}}_{(m+1),\beta} \cdot \widehat{\mathbf{x}}_0 + \mathbf{D}_1 \cdot \widehat{\mathbf{K}}_{(m+1),\beta} \cdot \widehat{\mathbf{x}}_1 \\ &= \mathbf{D}_0 \cdot \mathbf{x}_0 + \mathbf{D}_1 \cdot \mathbf{x}_1 \quad \text{(by Eq. 3)} \\ &= \mathbf{b} \mod q \quad \text{(by Eq. 2)} \end{aligned}$$

(vi) Let $\mathsf{VALID} = \{\mathbf{u} \in \{-1, 0, 1\}^L : \mathbf{u} = [\mathbf{u}_0 || \mathbf{u}_1]^t$ for some $\mathbf{u}_0, \mathbf{u}_1 \in B^3_{(m+1)p}\}$ and $\mathcal{S} = \mathcal{S}_{3(m+1)p} \times \mathcal{S}_{3(m+1)p}$. Then $\mathbf{x} = [\widehat{\mathbf{x}}_0 || \widehat{\mathbf{x}}_1]^t \in \mathsf{VALID}$ as $\widehat{\mathbf{x}}_0 \in B^3_{(m+1)p}$, $\widehat{\mathbf{x}}_1 \in B^3_{(m+1)p}$. Also for any randomly selected permutation $\pi = (\pi_0, \pi_1) \in \mathcal{S}$ and vector $\mathbf{x} = [\widehat{\mathbf{x}}_0 || \widehat{\mathbf{x}}_1]^t \in \mathsf{VALID}$, the vector $T_\pi(\mathbf{x}) = (\pi_0(\widehat{\mathbf{x}}_0), \pi_1(\widehat{\mathbf{x}}_1)) \in \mathsf{VALID}$ and $T_\pi(\mathbf{x})$ is uniform in VALID whenever $\mathbf{x} = [\widehat{\mathbf{x}}_0 || \widehat{\mathbf{x}}_1]^t$ is uniform in VALID.

(vii) The nominee NE invokes the algorithm ZKAoK described in Sect. 2.2 for the relation $\mathcal{R} = \{(\mathbf{P}, \mathbf{b}) \in \mathbb{Z}_q^{D \times L} \times \mathbb{Z}_q^D, \mathbf{x} \in \mathsf{VALID} : \mathbf{P}\mathbf{x} = \mathbf{b} \bmod q\}$ to prove the knowledge of the witness $\mathbf{x}$ in statistical zero knowledge argument of knowledge and generates a proof

$$\Pi = (\{\mathsf{COM}_\gamma\}_{\gamma=1}^{s}, \ \mathsf{Ch}, \ \{\mathsf{RSP}_\gamma\}_{\gamma=1}^{s})$$

where $\mathsf{COM}_\gamma \leftarrow \mathsf{ZKAoK.Commitment}(\mathbf{P}, \mathbf{b}, \mathbf{x})$, $\mathsf{Ch} = H(M, \{\mathsf{COM}_\gamma\}_{\gamma=1}^{s}, \mathbf{y}_1) \in \{1,2,3\}^s$, $\mathsf{RSP}_\gamma \leftarrow \mathsf{ZKAoK.Response}(\mathsf{Ch}[\gamma], \rho_1^{(\gamma)}, \rho_2^{(\gamma)}, \rho_3^{(\gamma)}, \pi^{(\gamma)}, \mathbf{r}^{(\gamma)}, \mathbf{x})$ where $\mathsf{Ch}[\gamma]$ is the γ-th digit of $\mathsf{Ch} \in \{1,2,3\}^s$, $s = \omega(\log n)$ and $\rho_1^{(\gamma)}$, $\rho_2^{(\gamma)}$, $\rho_3^{(\gamma)}$, $\pi^{(\gamma)}$, $\mathbf{r}^{(\gamma)}$ are as selected by the nominee NE in the γ-th run of the algorithm $\mathsf{ZKAoK.Commitment}(\mathbf{P}, \mathbf{b}, \mathbf{x})$ for $\gamma = 1, 2, \ldots, s$.

(viii) Finally, the nominee NE sends the signature $\mathsf{Sig}_{M,\mathsf{NE},\mathsf{NR}} = (\Pi, \mathbf{y}_1)$ to the nominator NR over a public channel and stores $(r_1, \mathbf{v}, M, \mathsf{NR}, \mathbf{y}_1)$ in its current state $\mathsf{state}_{\mathsf{NE}}$ where $\mathbf{y}_1$ works as the session identity which is session specific.

NS.SignNR$(\mathsf{param}, \mathsf{SK}_{\mathsf{NR}}, \mathsf{PK}_{\mathsf{NR}}, \mathsf{pk}_{\mathsf{NE}}, M, \mathsf{Sig}_{M,\mathsf{NE},\mathsf{NR}}) \rightarrow (\mathsf{nsig}_{M,\mathsf{NE},\mathsf{NR}} = (\mathbf{z}, \mathbf{y}_1))$**.** On receiving the signature $\mathsf{Sig}_{M,\mathsf{NE},\mathsf{NR}} = (\Pi, \mathbf{y}_1)$ from the nominee NE, the nominator NR executes the following steps and issues a nominative signature $\mathsf{nsig}_{M,\mathsf{NE},\mathsf{NR}} = (\mathbf{z}, \mathbf{y}_1)$ using $\mathsf{SK}_{\mathsf{NR}} = \mathsf{T}_{\mathbf{A}_{\mathsf{NR}}}$, $\mathsf{PK}_{\mathsf{NR}} = \mathbf{A}_{\mathsf{NR}}$ and $\mathsf{pk}_{\mathsf{NE}} = \mathbf{B}_{\mathsf{NE}}$.

(i) The NR computes $\mathbf{y} = H_1(M||\mathbf{A}_{\mathsf{NR}}||\mathbf{B}_{\mathsf{NE}})$ and verifies the zero knowledge proof $\Pi = (\{\mathsf{COM}_\gamma\}_{\gamma=1}^{s}, \mathsf{Ch}, \{\mathsf{RSP}_\gamma\}_{\gamma=1}^{s})$ for the equation $\mathbf{Px} = \mathbf{b} \bmod q$ by computing $\mathsf{VRF}_\gamma \leftarrow \mathsf{ZKAoK.Verification}(\mathbf{P}, \mathbf{b}, \mathsf{RSP}_\gamma, \mathsf{Ch}[\gamma], \mathsf{COM}_\gamma)$ and verifying whether $\mathsf{VRF}_\gamma = 1$ for all $\gamma = 1, 2, \ldots, s$ where RSP_γ, $\mathsf{Ch}[\gamma]$, COM_γ are as defined in step (vi) of the algorithm $\mathsf{NS.SignNE}(\mathsf{param}, \mathsf{sk}_{\mathsf{NE}}, \mathsf{pk}_{\mathsf{NE}}, \mathsf{PK}_{\mathsf{NR}}, M)$. Note that $\mathbf{P} = [\widehat{\mathbf{D}}_0||\widehat{\mathbf{D}}_1]$ and $\mathbf{b} = [\mathbf{y}||\mathbf{y}_1]^t$ are publicly computable, $\mathbf{y}_1$ is extracted from $\mathsf{Sig}_{M,\mathsf{NE},\mathsf{NR}}$ and $\mathsf{pk}_{\mathsf{NE}} = \mathbf{B}_{\mathsf{NE}}$ where $\widehat{\mathbf{D}}_0 = \mathbf{D}_0 \cdot \widehat{\mathbf{K}}_{(m+1),\beta}$ and $\widehat{\mathbf{D}}_1 = \mathbf{D}_1 \cdot \widehat{\mathbf{K}}_{(m+1),\beta}$. The witness $\mathbf{x} = [\widehat{\mathbf{x}}_0||\widehat{\mathbf{x}}_1]^t$ is known only to the nominee NE.

(ii) If the verification fails, the nominator NR aborts; otherwise the nominator NR finds a short vector

$$\mathbf{z} \in \mathbb{Z}_q^m \text{ satisfying } \mathbf{A}_{\mathsf{NR}} \cdot \mathbf{z} = \mathbf{y}_1 \bmod q \text{ with } ||\mathbf{z}|| \leq \sigma\sqrt{m}$$

using the short basis $\mathsf{SK}_{\mathsf{NR}} = \mathsf{T}_{\mathbf{A}_{\mathsf{NR}}}$ following the algorithm $\mathsf{SampleD}(\mathsf{T}_{\mathbf{A}_{\mathsf{NR}}}, \mathbf{A}_{\mathsf{NR}}, \mathbf{y}_1, \sigma) \rightarrow \mathbf{z}$ as in Lemma 1 in Sect. 2 and issues the nominative signature $\mathsf{nsig}_{M,\mathsf{NE},\mathsf{NR}} = (\mathbf{z}, \mathbf{y}_1)$.

NS.Verify$(\mathsf{param}, \mathsf{state}_{\mathsf{NE}}, \mathsf{pk}_{\mathsf{NE}}, \mathsf{PK}_{\mathsf{NR}}, M, \mathsf{nsig}_{M,\mathsf{NE},\mathsf{NR}}) \in \{\mathsf{valid}, \mathsf{invalid}\}$**.** This algorithm is executed by the nominee NE with its current internal state $\mathsf{state}_{\mathsf{NE}}$ who on receiving $\mathsf{nsig}_{M,\mathsf{NE},\mathsf{NR}} = (\mathbf{z}, \mathbf{y}_1)$ uses $\mathsf{PK}_{\mathsf{NR}} = \mathbf{A}_{\mathsf{NR}}$ and $\mathsf{pk}_{\mathsf{NE}} = \mathbf{B}_{\mathsf{NE}}$ to compute $\mathbf{y} = H_1(M||\mathbf{A}_{\mathsf{NR}}||\mathbf{B}_{\mathsf{NE}})$ and verify whether

$$\mathbf{y}_1 = \mathbf{B}_{\mathsf{NE}}^t \cdot (r_1\mathbf{y}) + \mathbf{v} \bmod q,\ \mathbf{A}_{\mathsf{NR}} \cdot \mathbf{z} = \mathbf{y}_1 \bmod q \text{ and } ||\mathbf{z}|| \leq \sigma\sqrt{m}$$

where the nominee NE extracts $\mathbf{v}$, r_1 from its internal secret state $\mathsf{state}_{\mathsf{NE}}$ which contains $(r_1, \mathbf{v}, M, \mathsf{NR}, \mathbf{y}_1)$. If the verification succeeds, it outputs valid; otherwise it returns invalid.

NS.ConfOrDisav = (TMnominee, TMverifier)**.** This protocol satisfies the following requirements:

(i) $\mathsf{TMnominee}(\mathsf{param}, \mathsf{state}_{\mathsf{NE}}, \mathsf{pk}_{\mathsf{NE}}, \mathsf{PK}_{\mathsf{NR}}, M, \mathsf{nsig}_{M,\mathsf{NE},\mathsf{NR}}) \rightarrow (\mu, \Pi_{\mathsf{confORdisav}})$. The nominee NE generates a proof

$$\Pi_{\mathsf{confORdisav}} = (\{\mathsf{COM}_\gamma\}_{\gamma=1}^{s}, \mathsf{Ch}, \{\mathsf{RSP}_\gamma\}_{\gamma=1}^{s})$$

for the relations $\mathbf{B}_{\mathsf{NE}} \cdot \mathbf{v} = \mathbf{y} \bmod q$, $\mathbf{B}_{\mathsf{NE}}^{t} \cdot (r_1\mathbf{y}) + \mathbf{v} = \mathbf{y}_1 \bmod q$ by converting this system of equations into an equation of the form $\mathbf{D}_0\mathbf{x}_0 + \mathbf{D}_1\mathbf{x}_1 = \mathbf{b} \bmod q$ which in turn is reduced to an equation of the form $\mathbf{P}\mathbf{x} = \mathbf{b}$ as explained in steps (iii) and (iv) respectively of the algorithm NS.SignNE, and then invoking the algorithm ZKAoK for the relation $\mathcal{R} = \{(\mathbf{P}, \mathbf{b}) \in \mathbb{Z}_q^{D \times L} \times \mathbb{Z}_q^{D}, \mathbf{x} \in \mathsf{VALID} : \mathbf{P}\mathbf{x} = \mathbf{b} \bmod q\}$ as in step (vi) of the algorithm NS.SignNE. Note that $\mathbf{P} = [\widehat{\mathbf{D}}_0 || \widehat{\mathbf{D}}_1]$ and $\mathbf{b} = [\mathbf{y} || \mathbf{y}_1]^t$ are publicly computable from param, $\mathsf{nsig}_{M,\mathsf{NE},\mathsf{NR}} = (\mathbf{z}, \mathbf{y}_1)$ and $\mathsf{pk}_{\mathsf{NE}} = \mathbf{B}_{\mathsf{NE}}$. The witness $\mathbf{x} = [\mathbf{x}_0 || \mathbf{x}_1]^t$ is known only to the nominee NE which is stored in its current internal state $\mathsf{state}_{\mathsf{NE}}$. It runs $\mathsf{NS.Verify}(\mathsf{param}, \mathsf{state}_{\mathsf{NE}}, \mathsf{pk}_{\mathsf{NE}}, \mathsf{PK}_{\mathsf{NR}}, M, \mathsf{nsig}_{M,\mathsf{NE},\mathsf{NR}})$. If the output is valid then it returns $(\mu = 1, \Pi_{\mathsf{confORdisav}})$ to the verifier VR. Otherwise, it sends $(\mu = 0, \Pi_{\mathsf{confORdisav}})$ to the verifier VR.

(ii) $\mathsf{TMverifier}(\mathsf{param}, \mathsf{pk}_{\mathsf{NE}}, \mathsf{PK}_{\mathsf{NR}}, M, \mathsf{nsig}_{M,\mathsf{NE},\mathsf{NR}}, \mu, \Pi_{\mathsf{confORdisav}}) \rightarrow \beta$. On receiving a pair $(\mu, \Pi_{\mathsf{confORdisav}})$ from the nominee NE, the verifier VR checks the bit μ.

- If $\mu = 1$, then it verifies the following.
  (a) $\mathsf{VRF}_\gamma = 1$ for all γ where $\mathsf{VRF}_\gamma \leftarrow \mathsf{ZKAoK.Verification}(\mathbf{P}, \mathbf{b}, \mathsf{RSP}_\gamma, \mathsf{Ch}[\gamma], \mathsf{COM}_\gamma)$. Here $\mathbf{P}$, $\mathbf{b}$ are computed by the verifier using param, $\mathsf{nsig}_{M,\mathsf{NE},\mathsf{NR}} = (\mathbf{z}, \mathbf{y}_1)$ and $\mathsf{PK}_{\mathsf{NE}} = \mathbf{B}_{\mathsf{NE}}$. Note that the witness $\mathbf{x}$ is known only to the nominee NE.
  (b) $\mathbf{A}_{\mathsf{NR}} \cdot \mathbf{z} = \mathbf{y}_1$ by extracting $\mathbf{z}$, $\mathbf{y}_1$ from nsig and using $\mathsf{PK}_{\mathsf{NR}} = \mathbf{A}_{\mathsf{NR}}$.
  If the verification succeeds, it outputs $\beta = 1$ indicating that the verifier VR agrees with the confirmation proof $\Pi_{\mathsf{confORdisav}}$ and convinces in zero knowledge that the nominator is not a cheater. Otherwise it disagrees with the confirmation proof by returning $\beta = 0$. This means that the verifier VR is not satisfied with the confirmation proof $\Pi_{\mathsf{confORdisav}}$.
- If the bit $\mu = 0$ then the verifier VR verifies whether any of the above mentioned conditions (a), (b) are violated, thereby agrees with the disavowal proof $\Pi_{\mathsf{confORdisav}}$ and convinces in zero knowledge that the nominator NR is a cheater. Otherwise it disagrees with the disavowal proof and returns $\beta = 0$ indicating that the verifier VR is not convinced with the proof.

Correctness:

- $((n, q, m, \sigma, \beta, H, H_1) = \mathsf{param}) \leftarrow \mathsf{NS.Setup}(\lambda)$,
- $(\mathsf{PK}_{\mathsf{NR}} = \mathbf{A}_{\mathsf{NR}}, \mathsf{SK}_{\mathsf{NR}} = \mathsf{T}_{\mathbf{A}_{\mathsf{NR}}}) \leftarrow \mathsf{NS.KeygenNR}(\mathcal{Y}, \mathsf{NR})$,
- $(\mathsf{pk}_{\mathsf{NE}} = \mathbf{B}_{\mathsf{NR}}, \mathsf{sk}_{\mathsf{NE}} = \mathsf{T}_{\mathbf{B}_{\mathsf{NE}}}) \leftarrow \mathsf{NS.KeygenNE}(\mathcal{Y}, \mathsf{NE})$,

- $(\mathsf{Sig}_{M,\mathsf{NE},\mathsf{NR}} = (\Pi, \mathbf{y}_1)) \leftarrow \mathsf{NS.SignNE}(\mathsf{param}, \mathsf{sk}_{\mathsf{NE}}, \mathsf{pk}_{\mathsf{NE}}, \mathsf{PK}_{\mathsf{NR}}, M)$ where $\mathbf{y}_1 = \mathbf{B}_{\mathsf{NE}}^t \cdot (r_1 \mathbf{y}) + \mathbf{v} \bmod q$, $r_1 \in [-\beta, \beta]$, $\mathbf{v} \in \mathbb{Z}_q^m$ is a short vector satisfying $\mathbf{B}_{\mathsf{NE}} \cdot \mathbf{v} = \mathbf{y} \bmod q$ with $||\mathbf{v}|| \leq \sigma\sqrt{m}$,
- $(\mathsf{nsig}_{M,\mathsf{NE},\mathsf{NR}} = \mathbf{z}) \leftarrow \mathsf{NS.SignNR}(\mathsf{param}, \mathsf{SK}_{\mathsf{NR}}, \mathsf{PK}_{\mathsf{NR}}, \mathsf{pk}_{\mathsf{NE}}, M, \mathsf{Sig}_{M,\mathsf{NE},\mathsf{NR}})$ where $\mathbf{z}$ satisfies the equation $\mathbf{A}_{\mathsf{NR}} \cdot \mathbf{z} = \mathbf{y}_1 \bmod q$,
- $(\mu, \Pi_{\mathsf{confORdisav}}) \leftarrow \mathsf{TMnominee}(\mathsf{param}, \mathsf{state}_{\mathsf{NE}}, \mathsf{pk}_{\mathsf{NE}}, \mathsf{PK}_{\mathsf{NR}}, M, \mathsf{nsig}_{M,\mathsf{NE},\mathsf{NR}})$ where $\mu \in \{0, 1\}$ and $\Pi_{\mathsf{confORdisav}} \leftarrow \mathsf{NS.ConfOrDisav.TMnominee}$ is a zero knowledge proof for the relation $\mathcal{R} = \{(\mathbf{P}, \mathbf{b}) \in \mathbb{Z}_q^{D \times L} \times \mathbb{Z}_q^D, \mathbf{x} \in \mathsf{VALID} : \mathbf{P}\mathbf{x} = \mathbf{b} \bmod q\}$.

If the nominee NE, the nominator NR and the verifier VR are honest then we have the following.

(i) $\mathsf{NS.Verify}(\mathsf{param}, \mathsf{state}_{\mathsf{NE}}, \mathsf{pk}_{\mathsf{NE}}, \mathsf{PK}_{\mathsf{NR}}, M, \mathsf{nsig}_{M,\mathsf{NE},\mathsf{NR}}) \rightarrow \mathsf{valid}$ as $\mathbf{A}_{\mathsf{NR}} \cdot \mathbf{z} = \mathbf{y}_1 \bmod q$.

(ii) $\mathsf{NS.ConfOrDisav.TMverifier}(\mathsf{param}, \mathsf{pk}_{\mathsf{NE}}, \mathsf{PK}_{\mathsf{NR}}, M, \mathsf{nsig}_{M,\mathsf{NE},\mathsf{NR}}, \mu, \Pi_{\mathsf{confORdisav}}) \rightarrow (\beta = 1)$

4 Security

Threat Model. Security attributes of a nominative signature can be broadly classified into four categories –
(*Unforgeability against malicious nominee*) The nominee NE alone cannot produce a valid nominative signature where the nominee NE and the message M both are chosen by the nominator NR.
(*Unforgeability against malicious nominator*) The nominator NR alone cannot produce a valid nominative signature and cannot convince a verifier about the validity or invalidity of a nominative signature.
(*Security against invisibility*) Only the nominee NE can verify the nominative signature nsig.
(*Security against repudiation*) If the nominative signature nsig is valid then the nominee NE cannot mislead a verifier VR and cannot prove the invalidity of nsig to the verifier VR and vice versa.

4.1 Oracles for Adversaries

An adversary $\mathcal{A}$ invokes the following oracles accessible in the attack games and interacts with a stateful interface I who runs NS.Setup to generate param and maintains seven private lists: LcreateNR, LcreateNE, LcorruptNR, LcorruptNE, LsignNR, LsignNE, LconfORdisav.

- CreateNR Query: When $\mathcal{A}$ invokes this oracle on a nominator u, the interface I returns PK_u to $\mathcal{A}$ by running $\mathsf{NS.KeygenNR}(\mathsf{param}, u) \rightarrow (\mathsf{PK}_u, \mathsf{SK}_u)$. The interface I stores $(\mathsf{PK}_u, \mathsf{SK}_u)$ in the list LcreateNR.
- CreateNE Query: In response to this query for a nominee v from $\mathcal{A}$, the interface I runs $\mathsf{NS.KeygenNE}(\mathsf{param}, v) \rightarrow (\mathsf{pk}_v, \mathsf{sk}_v)$ and passes pk_v to $\mathcal{A}$. The interface stores the pair $(\mathsf{pk}_v, \mathsf{sk}_v)$ in the list LcreateNE.

- CorruptNR Query: On receiving this query on a nominator u from $\mathcal{A}$, the interface I checks whether $(\mathsf{PK}_u, \mathsf{SK}_u) \in \mathsf{LcreateNR}$. If not, it returns $\perp$. Otherwise, I sends SK_u to $\mathcal{A}$ and stores PK_u in the list $\mathsf{LcorruptNR}$.
- CorruptNE Query: In response to this query on a nominee v from $\mathcal{A}$, the interface I checks whether $(\mathsf{pk}_v, \mathsf{sk}_v) \in \mathsf{LcreateNE}$. If not, it returns $\perp$. Otherwise, I returns $(\mathsf{sk}_v, \mathsf{state}_v)$ to $\mathcal{A}$ and stores pk_v in the list $\mathsf{LcorruptNE}$. Here state_v is the current internal secret state of the nominee v which is initially empty.
- SignNE Query: On querying this oracle on a tuple (v, u, M) by $\mathcal{A}$ where v is a nominee, u is a nominator and M is a message, the interface I checks whether $(\mathsf{pk}_v, \mathsf{sk}_v) \in \mathsf{LcreateNE}$ and $(\mathsf{PK}_u, \mathsf{SK}_u) \in \mathsf{LcreateNR}$. If not, I returns $\perp$. Otherwise, I outputs the signature $\mathsf{Sig}_{M,v,u} \leftarrow \mathsf{NS.SignNE}(\mathsf{param}, \mathsf{sk}_v, \mathsf{pk}_v, \mathsf{PK}_u, M)$ of the nominee v on M and stores $(\mathsf{Sig}_{M,v,u}, \mathsf{state}_v)$ in the list $\mathsf{LsignNE}$ where state_v is the current internal secret state of the nominee v.
- SignNR Query: In response to this query on $\mathsf{Sig}_{M,v,u}$ from $\mathcal{A}$, the interface I verifies whether $(\mathsf{Sig}_{M,v,u}, \mathsf{state}_v) \in \mathsf{LSignNE}$. If so, the interface I returns the nominative signature $\mathsf{nsig}_{M,v,u} \leftarrow \mathsf{NS.SignNR}(\mathsf{param}, \mathsf{SK}_u, \mathsf{PK}_u, \mathsf{pk}_v, M, \mathsf{Sig}_{M,v,u})$ to $\mathcal{A}$ and stores $(\mathsf{Sig}_{M,v,u}, \mathsf{nsig}_{M,v,u})$ in the list $\mathsf{LsignNR}$. Otherwise, I returns $\perp$.
- ConfOrDisav Query: The interface I responses on receiving this query on $\mathsf{nsig}_{M,v,u}$ from $\mathcal{A}$ by checking if $(\mathsf{Sig}_{M,v,u}, \mathsf{nsig}_{M,v,u}) \in \mathsf{LsignNR}$. If not, I aborts. Otherwise, I extracts state_v from $(\mathsf{Sig}_{M,v,u}, \mathsf{state}_v) \in \mathsf{LSignNE}$ and returns $(\mu, \Pi_{\mathsf{confORdisav}}) \leftarrow \mathsf{NS.ConfOrDisav.TMnominee}(\mathsf{param}, \mathsf{state}_v, \mathsf{pk}_v, \mathsf{PK}_u, M, \mathsf{nsig}_{M,v,u})$ to $\mathcal{A}$. The interface I stores $(\mathsf{nsig}_{M,v,u}, \mu, \Pi_{\mathsf{confORdisav}})$ in the list $\mathsf{LconfORdisav}$.

4.2 Security Model for Unforgeability Against Malicious Nominee

This is a security game $\mathsf{Exp}^{\mathsf{unforg}}_{\mathcal{F}}$ explained in Fig. 2 played between a forger $\mathcal{F}$ and a simulator $\mathcal{S}$.

Definition 6 (Unforgeability against malicious nominee)**.** *We say that a nominative signature is secure under unforgeability against malicious nominee if*

$$\mathsf{Adv}^{\mathsf{unforg}}_{\mathcal{F}}(\lambda) = \mathsf{Prob}[\mathsf{Exp}^{\mathsf{unforg}}_{\mathcal{F}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

for every PPT adversary $\mathcal{F}$ in the experiment $\mathsf{Exp}^{\mathsf{unforg}}_{\mathcal{F}}(\lambda)$ defined in Fig. 2 where $\mathsf{negl}(\lambda)$ is a negligible function in λ i.e., $\mathsf{negl}(\lambda) = \lambda^{-\omega(1)}$.

4.3 Security Model Under Unforgeability Against Malicious Nominator

Let $\mathcal{F}$ be a forger and $\mathcal{S}$ be a simulator. This security is modeled by the game $\mathsf{Exp}^{\mathsf{unforgNR}}_{\mathcal{F}}(\lambda)$ between $\mathcal{F}$ and $\mathcal{S}$ as provided in Fig. 3.

Definition 7 (Unforgeability against malicious nominator)**.** *We say that a nominative signature is secure against malicious nominator if*

$$\mathsf{Adv}^{\mathsf{unforgNR}}_{\mathcal{F}}(\lambda) = \mathsf{Prob}[\mathsf{Exp}^{\mathsf{unforgNR}}_{\mathcal{F}}(\lambda) = 1] \leq \mathsf{negl}(\lambda)$$

for every PPT adversary $\mathcal{F}$ in the experiment $\mathsf{Exp}^{\mathsf{unforgNR}}_{\mathcal{F}}(\lambda)$ defined in Fig. 3 and $\mathsf{negl}(\lambda)$ is a negligible function of λ.

5 Security Model Against Invisibility

Let $\mathcal{D}$ be a distinguisher and $\mathcal{C}$ be the challenger. The invisibility game $\mathsf{Exp}^{\mathsf{invis}}_{\mathcal{D}}(\lambda, b)$ is described in Fig. 4.

Definition 8 (security against invisibility)**.** *A nominative signature scheme is secure under invisibility*

$$\mathsf{Adv}^{\mathsf{invis}}_{\mathcal{D}}(\lambda) = |\mathsf{Prob}[\mathsf{Exp}^{\mathsf{invis}}_{\mathcal{D}}(\lambda, 0)] - \mathsf{Prob}[\mathsf{Exp}^{\mathsf{invis}}_{\mathcal{D}}(\lambda, 1)]| \leq \mathsf{negl}(\lambda)$$

---

1. The simulator $\mathcal{S}$ generates system parameters $\mathsf{param} \leftarrow \mathsf{NS.Setup}(\lambda)$ and sends it to the forger $\mathcal{F}$.
2. The forger $\mathcal{F}$ makes polynomially many, say α, queries to $\mathcal{S}$ for each of the oracles CreateNR, CreateNE, CorruptNR, CorruptNE, SignNR, SignNE, ConfOrDisav, thereby has the knowledge of VIEW where

$$\mathsf{VIEW} = \begin{cases} \mathsf{param},\ \{\mathsf{PK}_u \mid (\mathsf{PK}_u, \mathsf{SK}_u) \in \mathsf{LcreateNR}\},\ \{\mathsf{pk}_v \mid (\mathsf{pk}_v, \mathsf{sk}_v) \in \mathsf{LcreateNE}\}, \\ \{\mathsf{SK}_u \mid \mathsf{PK}_u \in \mathsf{LcorruptNR}\},\ \{\mathsf{sk}_v \mid \mathsf{pk}_v \in \mathsf{LcorruptNE}\}, \\ \mathsf{LsignNE} = \{(\mathsf{Sig}_{M,v,u}, \mathsf{state}_v) \mid \mathsf{Sig}_{M,v,u} \leftarrow \mathsf{NS.SignNE}(\mathsf{param}, \mathsf{sk}_v, \mathsf{pk}_v, \mathsf{PK}_u, M)\}, \\ \mathsf{LsignNR} = \Big\{(\mathsf{Sig}_{M,v,u}, \mathsf{nsig}_{M,v,u}) \mid (\mathsf{Sig}_{M,v,u}, \mathsf{state}_v) \in \mathsf{LsignNE} \text{ and} \\ \qquad \mathsf{nsig}_{M,v,u} \leftarrow \mathsf{NS.SignNR}(\mathsf{param}, \mathsf{SK}_u, \mathsf{PK}_u, \mathsf{pk}_v, M, \mathsf{Sig}_{M,v,u})\Big\}, \\ \mathsf{LconfORdisav} = \Big\{(\mathsf{nsig}_{M,v,u}, \mu, \Pi_{\mathsf{confORdisav}}) \mid (\mathsf{Sig}_{M,v,u}, \mathsf{nsig}_{M,v,u}) \in \mathsf{LsignNR} \\ \qquad \text{and } (\mu, \Pi_{\mathsf{confORdisav}}) \leftarrow \mathsf{NS.ConfOrDisav.TMnominee}(\mathsf{param}, \\ \qquad\qquad \mathsf{state}_v, \mathsf{pk}_v, \mathsf{PK}_u, M, \mathsf{nsig}_{M,v,u})\Big\} \\ \text{where each of } |\mathsf{LcreateNR}|,\ |\mathsf{LcreateNE}|,\ |\mathsf{LcorruptNR}|,\ |\mathsf{LcorruptNE}|, \\ |\mathsf{LsignNE}|,\ |\mathsf{LsignNR}|,\ |\mathsf{LconfORdisav}| \leq \alpha. \end{cases}$$

3. Finally, $\mathcal{F}$ outputs a forgery $(M^*, \mathsf{nsig}^*_{M^*,\mathsf{NE},\mathsf{NR}})$ on a corrupted nominee NE and an uncorrupted nominator NR such that $(\mathsf{PK}_{\mathsf{NR}}, \mathsf{SK}_{\mathsf{NR}}) \in \mathsf{LcreateNR}$ and $\mathsf{pk}_{\mathsf{NE}} \in \mathsf{LcorruptNE}$.
4. The simulator $\mathcal{S}$ returns 1 if the following conditions hold:
   (a) $\mathsf{NS.Verify}(\mathsf{param}, \mathsf{state}_{\mathsf{NE}}, \mathsf{pk}_{\mathsf{NE}}, \mathsf{PK}_{\mathsf{NR}}, M^*, \mathsf{nsig}^*_{M^*,\mathsf{NE},\mathsf{NR}}) \rightarrow \mathsf{valid}$ i.e., $\mathsf{nsig}^*_{M^*,\mathsf{NE},\mathsf{NR}}$ is a valid signature,
   (b) $\mathsf{PK}_{\mathsf{NR}} \notin \mathsf{LcorruptNR}$ i.e., nominator NR is not corrupted,
   (c) $(\mathsf{Sig}'_{M^*,\mathsf{NE},\mathsf{NR}}, \mathsf{nsig}^*_{M^*,\mathsf{NE},\mathsf{NR}}) \notin \mathsf{LsignNR}$ where $(\mathsf{Sig}'_{M^*,\mathsf{NE},\mathsf{NR}}, \mathsf{state}_{\mathsf{NE}}) \in \mathsf{LsignNE}$ i.e., $\mathsf{nsig}'_{M^*,\mathsf{NE},\mathsf{NR}} \neq \mathsf{nsig}^*_{M^*,\mathsf{NE},\mathsf{NR}}$ for the pair $(\mathsf{Sig}'_{M^*,\mathsf{NE},\mathsf{NR}}, \mathsf{nsig}'_{M^*,\mathsf{NE},\mathsf{NR}}) \in \mathsf{LsignNR}$, and SignNR query is made only ones on $\mathsf{Sig}'_{M^*,\mathsf{NE},\mathsf{NR}}$.
   (d) $(\mathsf{nsig}^*_{M^*,\mathsf{NE},\mathsf{NR}}, \mu, \Pi_{\mathsf{confORdisav}}) \notin \mathsf{LconfORdisav}$ i.e., $\mathsf{nsig}^*_{M^*,\mathsf{NE},\mathsf{NR}}$ has not been queried to the protocol NS.ConfOrDisav for the conformation or disavowal proof of the validity of the nominative signature $\mathsf{nsig}^*_{M^*,\mathsf{NE},\mathsf{NR}}$.
   Otherwise, $\mathcal{S}$ returns 0.
5. The forger $\mathcal{F}$ wins the game if $\mathcal{S}$ returns 1.

Fig. 2. Small Security game $\mathsf{Exp}^{\mathsf{unforg}}_{\mathcal{F}}(\lambda)$ under unforgeability against malicious nominee.

for every PPT adversary in the experiment $\mathsf{Exp}^{\mathsf{invis}}_{\mathcal{D}}(\lambda, b)$ *defined in Fig. 4 where* $b \in \{0,1\}$ *and* $\mathsf{negl}(\lambda)$ *is a negligible function in* λ.

Remark 2. In the above security game (Fig. 2) if SignNR query is made more than ones on $\mathsf{Sig}'_{M^*,\mathsf{NE},\mathsf{NR}}$ then the adversary can compute a nominator's signature as follows:

Suppose an adversary queried SignNR on $\mathsf{Sig}'_{M^*,\mathsf{NE},\mathsf{NR}}$ two or more times then the adversary has $\mathbf{A}_{\mathsf{NR}} \cdot \mathbf{z}_1 = \mathbf{y}_1 \bmod q$ and $\mathbf{A}_{\mathsf{NR}} \cdot \mathbf{z}_2 = \mathbf{y}_1 \bmod q$. That gives to the adversary $\mathbf{A}_{\mathsf{NR}} \cdot (\mathbf{z}_1 + \mathbf{z}_2)/2 = \mathbf{y}_1 \bmod q$. As q is a prime, 2 is invertible in $\mathbb{Z}_q$. Thus the adversary has another signature $\mathsf{nsig}_{M^*,\mathsf{NE},\mathsf{NR}} = (\mathbf{z}_1 + \mathbf{z}_2)/2$.

---

1. The simulator $\mathcal{S}$ generates system parameters $\mathsf{param} \leftarrow \mathsf{NS.Setup}(\lambda)$ and sends it to the forger $\mathcal{F}$.
2. The forger $\mathcal{F}$ makes polynomially many, say α, queries to $\mathcal{S}$ for each of the oracles CreateNR, CreateNE, CorruptNR, CorruptNE, SignNR, SignNE, ConfOrDisav and has the same VIEW as given in Figure 2.
3. Finally, the forger $\mathcal{F}$ outputs a forgery $(M^*, \mathsf{Sig}^*_{M^*,\mathsf{NE},\mathsf{NR}}, \mathsf{nsig}^*_{M^*,\mathsf{NE},\mathsf{NR}})$ on a corrupted nominator NR and an uncorrupted nominee NE such that $\mathsf{PK}_{\mathsf{NR}} \in \mathsf{LcorruptNR}$, $(\mathsf{pk}_{\mathsf{NE}}, \mathsf{sk}_{\mathsf{NE}}) \in \mathsf{LcreateNE}$.
4. The simulator $\mathcal{S}$ returns 1 if the following holds:
   (a) $\mathsf{NS.Verify}(\mathsf{param}, \mathsf{state}_{\mathsf{NE}}, \mathsf{pk}_{\mathsf{NE}}, \mathsf{PK}_{\mathsf{NR}}, M^*, \mathsf{nsig}^*_{M^*,\mathsf{NE},\mathsf{NR}}) \rightarrow \mathsf{valid}$.
   (b) $\mathsf{pk}_{\mathsf{NE}} \notin \mathsf{LcorruptNE}$.
   (c) $\mathsf{Sig}^*_{M^*,\mathsf{NE},\mathsf{NR}} \notin \mathsf{LsignNE}$ and $(M^*, \mathsf{NE}, \mathsf{NR})$ query is made only ones to the SignNE oracle.
   (d) $\mathsf{VRF}_\gamma = 1$ for all $\gamma = 1, 2, \ldots, s$ by computing $\mathsf{VRF}_\gamma \leftarrow \mathsf{ZKAoK.Verification}(\mathbf{P}, \mathbf{b}, \mathsf{RSP}_\gamma, \mathsf{Ch}[\gamma], \mathsf{COM}_\gamma)$ where RSP_γ, $\mathsf{Ch}[\gamma]$, COM_γ are as defined in step (vi) of the algorithm NS.SignNE.
   (e) $(\mathsf{nsig}^*_{M^*,\mathsf{NE},\mathsf{NR}}, \mu, \Pi^*_{M^*,\mathsf{NE},\mathsf{NR}}) \notin \mathsf{LcreateORdisav}$.
   Otherwise, $\mathcal{S}$ returns 0.
5. The forger $\mathcal{F}$ wins the game if $\mathcal{S}$ returns 1.

Fig. 3. Security game $\mathsf{Exp}^{\mathsf{unforgNR}}_{\mathcal{F}}(\lambda)$ under security against malicious nominator

5.1 Security Model for Non-repudiation

Let $\mathcal{A}$ be a cheating nominee and $\mathcal{C}$ be the challenger. Its security game $\mathsf{Exp}^{\mathsf{rep}}_{\mathcal{A}}(\lambda)$ is explained in Fig. 5.

Definition 9 (Non-repudiation)**.** *A nominative signature scheme is secure against non-repudiation if*

$$\mathsf{Adv}^{\mathsf{rep}}_{\mathcal{A}}(\lambda) = |\mathsf{prob}[\mathsf{Exp}^{\mathsf{rep}}_{\mathcal{A}}(\lambda) = 1]| \leq \mathsf{negl}(\lambda)$$

for every PPT adversary in the experiment $\mathsf{Exp}^{\mathsf{rep}}_{\mathcal{A}}(\lambda)$ *defined in Fig. 5 and* $\mathsf{negl}(\lambda)$ *is a negligible function of* λ.

1. The challenger $\mathcal{C}$ generates system parameters $\mathsf{param} \leftarrow \mathsf{NS.Setup}(\lambda)$ and sends it to the distinguisher $\mathcal{D}$.
2. Next the distinguisher $\mathcal{D}$ makes polynomially many, say α queries to $\mathcal{S}$ for each of the oracles CreateNR, CreateNE, CorruptNR, CorruptNE, SignNR, SignNE, ConfOrDisav to get the knowledge of VIEW where VIEW is same as in Figure 2.
3. At any point of the game, $\mathcal{D}$ submits a tuple $(M^*, \mathsf{NE}, \mathsf{NR})$ where M^* is a message to be signed with NE as the nominee and NR as the nominator such that $\mathsf{pk_{NE}}$, $\mathsf{PK_{NR}} \in \mathsf{VIEW}$ but $\mathsf{sk_{NE}} \notin \mathsf{VIEW}$ i.e., the nominee NE is not corrupted.
4. The challenger chooses a random bit $b \in \{0, 1\}$. If $b = 1$, the challenger $\mathcal{C}$ generates $\mathsf{Sig}_{M^*,\mathsf{NE},\mathsf{NR}} \leftarrow \mathsf{NS.SignNE}(\mathsf{param}, \mathsf{sk_{NE}}, \mathsf{pk_{NE}}, \mathsf{PK_{NR}}, M^*)$, $\mathsf{nsig}_{M^*,\mathsf{NE},\mathsf{NR}} \leftarrow \mathsf{NS.SignNR}(\mathsf{param}, \mathsf{SK_{NR}}, \mathsf{PK_{NR}}, \mathsf{pk_{NE}}, M^*, \mathsf{Sig}_{M^*,\mathsf{NE},\mathsf{NR}})$ and sets $\mathsf{K}_b = \mathsf{nsig}_{M^*,\mathsf{NE},\mathsf{NR}}$. Else, K_b is generated uniformly.
5. The distinguisher $\mathcal{D}$ observes K_b, outputs a guess b' and wins the game if
   (i) $b' = b$
   (ii) $\mathcal{D}$ does not corrupt $\mathsf{sk_{NE}}$ i.e., $\mathsf{pk_{NE}} \notin \mathsf{LcorruptNE}$

Fig. 4. Security game $\mathsf{Exp}^{\mathsf{invis}}_{\mathcal{D}}(\lambda, b)$ against invisibility

1. The challenger $\mathcal{C}$ generates $\mathsf{param} \leftarrow \mathsf{NS.Setup}(\lambda)$ and sends it to the adversary $\mathcal{A}$.
2. The adversary $\mathcal{A}$ may make polynomially many, say α, queries to oracles CreateNR, CreateNE, CorruptNR, CorruptNE, SignNR, SignNE, ConfOrDisav. The adversary $\mathcal{A}$ has the same VIEW as in Figure 2.
3. The adversary $\mathcal{A}$ prepares a tuple $(M^*, \mathsf{nsig}_{M^*,\mathsf{NE},\mathsf{NR}}, \mu)$ where NE is any nominee with $\mathsf{pk_{NE}} \in \mathsf{LcorruptNE}$, NR is a nominator such that $(\mathsf{PK_{NR}}, \mathsf{SK_{NR}}) \in \mathsf{LcreateNR}$, $\mathsf{nsig}_{M^*,\mathsf{NE},\mathsf{NR}}$ is a signature on M^* and μ is a bit. If $\mathsf{NS.Verify}(\mathsf{param}, \mathsf{state}_{M^*,\mathsf{NE},\mathsf{NR}}, \mathsf{pk_{NE}}, \mathsf{PK_{NR}}, M^*, \mathsf{nsig}_{M^*,\mathsf{NE},\mathsf{NR}}) \rightarrow \mathsf{valid}$ then $\mu = 1$. Else $\mu = 0$. Note that $\mathsf{pk_{NE}} \in \mathsf{LcorruptNE}$ means $\mathcal{A}$ has the knowledge of $(\mathsf{sk_{NE}}, \mathsf{state}_{M,\mathsf{NE},\mathsf{NR}})$.
4. To mislead, the adversary $\mathcal{A}$ runs the disavowal proof $\Pi_{\mathsf{confORdisav}}$ if $\mu = 1$. Otherwise, $\mathcal{A}$ computes the confirmation proof $\Pi_{\mathsf{confORdisav}}$. challenger $\mathcal{C}$ runs $\mathsf{NS.ConfOrDisav.TMverifier}(\mathsf{param}, \mathsf{pk_{NE}}, \mathsf{PK_{NR}}, M^*, \mathsf{nsig}_{M^*,\mathsf{NE},\mathsf{NR}}, \mu, \Pi_{\mathsf{confORdisav}}) \rightarrow \beta$ and returns β.
   The adversary $\mathcal{A}$ wins the game if $\beta = 1$.

Fig. 5. Security game $\mathsf{Exp}^{\mathsf{rep}}_{\mathcal{A}}(\lambda)$ under non-repudiation

Theorem 2. *Assuming the hardness of* SIS *search problem, the construction of our nominative signature scheme* NS = {Setup, KeygenNR, KeygenNE, SignNE, SignNR, Verify, ConfOrDisav = (TMnominee, TMverifier)} *described in Sect. 3 is secure under the unforgeability against malicious nominee as per the Definition 6 for the security game given in Fig. 2.*

Theorem 3. *Assuming the hardness of* SIS *search problem, the construction of our nominative signature scheme* NS = {Setup, KeygenNR, KeygenNE, SignNE,

SignNR, Verify, ConfOrDisav = (TMnominee, TMverifier)} *described in Sect. 3 is secure in the random oracle model under the unforgeability against malicious nominator as per the Definition 7 for the security game given in Fig. 3.*

Theorem 4. *Assuming the hardness of* decisional SIS *and* LWE*, the construction of our nominative signature scheme* NS = {Setup, KeygenNR, KeygenNE, SignNE, SignNR, Verify, ConfOrDisav = (TMnominee, TMverifier)} *described in Sect. 3 is secure under invisibility as per the Definition 8 for the security game given in Fig. 4.*

Theorem 5. *Our nominative signature scheme is secure against repudiation by nominee if no PPT cheating nominee has a non negligible advantage in the security game given in Fig. 5.*

Proof. By the soundness property of a proof system, the verifier will accept a language $\mathbf{x} \notin$ VALID with probability atmost $\epsilon \in [0, 1/2)$ while for any language $\mathbf{x} \in$ VALID, the verifier will reject with probability $\epsilon \in [0, 1/2)$.

Proofs of all the above Theorems 2, 3 and 4 will be given in the full version of the paper.

References

1. Ajtai, M.: Generating hard instances of lattice problems. In: Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing, pp. 99–108. ACM (1996)
2. Alwen, J., Peikert, C.: Generating shorter bases for hard random lattices. Theory Comput. Syst. **48**(3), 535–553 (2011)
3. Fiat, A., Shamir, A.: How to prove yourself: practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO 1986. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (1987). https://doi.org/10.1007/3-540-47721-7_12
4. Huang, Q., Liu, D.Y., Wong, D.S.: An efficient one-move nominative signature scheme. Int. J. Appl. Cryptogr. **1**(2), 133–143 (2008)
5. Huang, Z., Wang, Y.: Convertible nominative signatures. In: Wang, H., Pieprzyk, J., Varadharajan, V. (eds.) ACISP 2004. LNCS, vol. 3108, pp. 348–357. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-27800-9_30
6. Kim, S.J., Park, S.J., Won, D.H.: Nominative signatures. In: ICEIC: International Conference on Electronics, Informations and Communications, pp. 68–71 (1995)
7. Libert, B., Ling, S., Mouhartem, F., Nguyen, K., Wang, H.: Signature schemes with efficient protocols and dynamic group signatures from lattice assumptions. In: Cheon, J.H., Takagi, T. (eds.) ASIACRYPT 2016. LNCS, vol. 10032, pp. 373–403. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-53890-6_13
8. Ling, S., Nguyen, K., Stehlé, D., Wang, H.: Improved zero-knowledge proofs of knowledge for the ISIS problem, and applications. In: Kurosawa, K., Hanaoka, G. (eds.) PKC 2013. LNCS, vol. 7778, pp. 107–124. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-36362-7_8
9. Liu, D.Y.W., Chang, S., Wong, D.S., Mu, Y.: Nominative signature from ring signature. In: Miyaji, A., Kikuchi, H., Rannenberg, K. (eds.) IWSEC 2007. LNCS, vol. 4752, pp. 396–411. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-75651-4_27

10. Liu, D.Y.W., et al.: Formal definition and construction of nominative signature. In: Qing, S., Imai, H., Wang, G. (eds.) ICICS 2007. LNCS, vol. 4861, pp. 57–68. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-77048-0_5
11. Lyubashevsky, V.: Lattice signatures without trapdoors. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 738–755. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_43
12. Micciancio, D., Peikert, C.: Trapdoors for lattices: simpler, tighter, faster, smaller. In: Pointcheval, D., Johansson, T. (eds.) EUROCRYPT 2012. LNCS, vol. 7237, pp. 700–718. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-29011-4_41
13. Regev, O.: On lattices, learning with errors, random linear codes, and cryptography. J. ACM (JACM) **56**(6), 34 (2009)
14. Schuldt, J.C.N., Hanaoka, G.: Non-transferable user certification secure against authority information leaks and impersonation attacks. In: Lopez, J., Tsudik, G. (eds.) ACNS 2011. LNCS, vol. 6715, pp. 413–430. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-21554-4_24
15. Susilo, W., Mu, Y.: On the security of nominative signatures. In: Boyd, C., González Nieto, J.M. (eds.) ACISP 2005. LNCS, vol. 3574, pp. 329–335. Springer, Heidelberg (2005). https://doi.org/10.1007/11506157_28