

Quantum-Resistant Lattice-Based Proxy Signature

Jiaoyan Luo, Liming Zuo * and Hao Liu

School of Science, East China Jiaotong University, Nanchang 330013, China; jiaoyanluo@163.com (J.L.)

* Correspondence: limingzuo@126.com

Abstract: With the advancement of quantum computing, the utilization of quantum algorithms such as Shor's algorithm enables the efficient resolution of problems that are intractable in classical computing paradigms, posing a significant threat to traditional signature schemes. Lattice-based cryptography is considered one of the most promising post-quantum cryptographic algorithms due to its computational advantages and potential resistance to quantum attacks. Proxy signature is an authorization mechanism that allows the original signer to delegate the signing power to a proxy. The security of existing proxy signature schemes is mostly based on classical hard problems, which cannot guarantee security under quantum attacks. Therefore, this paper combines lattice-based cryptography with proxy signatures to propose a new lattice-based proxy signature scheme (NLBPS). NLBPS constructs signatures using lattice-based trapdoor sampling algorithms and preimage sampling algorithms. Comparative analysis shows that the proposed scheme has relatively smaller key and signature sizes compared to some existing lattice-based proxy signature schemes, and it also offers a certain improvement in computational efficiency.

Keywords: quantum resistance; lattice; proxy signature; small integer solution problem

1. Introduction

The concept of proxy signatures was introduced by Mambo, Usuda and Okamoto [1] in 1996. In this signature scheme, the original signer is allowed to delegate the signing authority to the proxy signer. By running an interactive proxy delegation protocol, the proxy signer can obtain a proxy signing key to sign messages on behalf of the original signer. In 2003, Bodyreva et al. [2] formally defined the security model for proxy signature schemes, and proposed a provably secure strong proxy signature scheme. To optimize the key management mechanism in public key cryptosystems, Shamir et al. [3] pioneeringly introduced identity-based signature schemes, where a user's public key can be directly derived from their identity information. Subsequently, Xu et al. [4] combined identity information with proxy signature technology to design an identity-based proxy signature scheme; however, the security model of this scheme only defined unforgeability and did not consider scenarios involving adaptive chosen messages and adaptive chosen identity attacks. In 2024, Bannore et al. [5] developed a role-based proxy signature scheme on elliptic curve groups, combining role authorization mechanisms with proxy signature technology, which is suitable for the field of e-government.

With the rapid development of quantum computing technology, malicious attackers can efficiently solve classically difficult problems in polynomial time using Shor's quantum algorithm [6], which poses a serious security threat to traditional schemes. Among

Academic Editor: Sergei Odintsov

Received: 12 January 2025

Revised: 31 January 2025

Accepted: 3 February 2025

Published: 9 February 2025

Citation: Luo, J.; Zuo, L.; Liu, H. Quantum-Resistant Lattice-Based Proxy Signature. *Symmetry* **2025**, *17*, 261. <https://doi.org/10.3390/sym17020261>

Copyright: © 2025 by the authors. Submitted for possible open access publication under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

the many quantum-resistant cryptographic schemes, lattice-based cryptography has garnered widespread attention due to its strong security proof in terms of its worst-case hardness [7] and efficient implementation. In 2008, Gentry et al. [8] designed a preimage sampling function using the Gaussian sampling algorithm and constructed a provably secure lattice signature scheme based on this technology, known as the GPV scheme. In 2010, Agrawal et al. [9] proposed a lattice basis delegation in a fixed dimension and presented two hierarchical identity-based encryption schemes. These schemes achieved fixed-dimensional delegation authorization, but the computational complexity of the public key generation process was relatively high. In 2013, Kim et al. [10] combined preimage sampling and fixed dimensional basis delegation technology to propose an identity-based proxy signature, and the scheme is proxy-protected in an adaptive security model, but only considered unforgeability in the security analysis of the scheme. In 2019, Wu et al. [11] proposed a proxy signature scheme on the NTRU lattice, which shortened the signature and key length. In 2021, Luo et al. [12] gave a formal concept of attribute-based proxy re-signature and discussed the potential applications of the scheme, whose public key and signature are relatively long. In 2022, Wang et al. [13] proposed a lattice proxy signature scheme suitable for cloud storage scenarios, but the complexity of the proxy authorization and proxy signing processes was relatively high. In 2023, Yu et al. [14] applied the lattice proxy signature scheme to the network coding environment, reducing the computational complexity and improving computational efficiency by reducing the dimension of the proxy key.

In this paper, we propose a new lattice-based proxy signature scheme called NLBPS. NLBPS uses a trapdoor sampling algorithm [15] to generate the public and private key matrices for the original signer and the proxy signer. It utilizes the preimage sampling function for proxy authorization and signature integrity. NLBPS is based on the hard assumption of the small integer solution problem and is provably secure in the random oracle model. Compared to some existing lattice-based proxy signature schemes, NLBPS achieves fixed-dimensional lattice-based proxy signatures with fewer original algorithms, and the required signature and public key sizes are shorter.

2. Preliminaries

2.1. Lattice and Lattice Problems

Definition 1. Let $B = [b_1, b_2, \dots, b_m] \in \mathbb{R}^n$ consist of m linearly independent vectors, and the lattice Λ is defined as follows:

$$\Lambda = \Lambda(B) = \{Bc : c \in \mathbb{Z}^m\} = \{\sum_{i=1}^m c_i b_i, c_i \in \mathbb{Z}\}.$$

B is the basis of the lattice Λ . n is the dimension of the lattice, m is the dimension of the vector and the rank of the lattice. If $m = n$, the lattice Λ is classified as a full-rank lattice. Unless stated otherwise, all lattices discussed in this paper are assumed to be full-rank lattices.

Typically, there are two kinds of special full-rank integer lattices established on $\mathbb{Z}_q$. For a matrix $B \in \mathbb{Z}_q^{n \times m}$, the orthogonal lattice and the q -ary integer lattice are defined as follows:

$$\begin{aligned}\Lambda_q^\perp(B) &= \{e \in \mathbb{Z}_q^m : Be = 0 \bmod q\} \\ \Lambda_q^u(B) &= \{e \in \mathbb{Z}_q^m : u = Be \bmod q, u \in \mathbb{Z}_q^n\}\end{aligned}$$

Definition 2. Let B is a basis of $\Lambda(B)$. The dual lattice of $\Lambda(B)$, denoted Λ^* , is defined to be $\Lambda^* = \Lambda^*(B^*) = \{y \in \text{span}(B) | \forall x \in \Lambda(B), \langle x, y \rangle \in \mathbb{Z}\}$, while the dual basis $B^* = (B^{-1})^T$ is a basis of Λ^* .

Definition 3. Given a basis $B \in \mathbb{Z}^{m \times n}$ of a lattice Λ , a rational $r > 0$, output a set $S = \{a_1, a_2, \dots, a_n\}$ of n linearly independent lattice vectors where $\|S\| \leq r\lambda_n$. Let this problem be denoted as $SIVP_{n,m,r}$.

Definition 4. Given $n, m, q \in \mathbb{N}$, a real $\beta > 0$ and a random matrix $B \in \mathbb{Z}_q^{n \times m}$, find a non-zero integer vector $e \in \mathbb{Z}^m$ such that $Be = 0 \bmod q$ and $\|e\| \leq \beta$. Let this problem be denoted as $SIS_{n,m,q,\beta}$.

Lemma 1 ([8]). For any $\beta = \text{poly}(n)$ and any prime $q \geq \beta\omega(\sqrt{n \log n})$, given $n, m \in \mathbb{N}^+$, the average-case problem $SIS_{n,m,q,\beta}$ is as hard as approximating the $SIVP_{n,m,r}$ problem in the worst case with certain $r = \beta \cdot \tilde{O}(\sqrt{n})$ factors.

Definition 5 ([8]). For any real $s > 0$, define the Gaussian function on $\mathbb{R}^n$ with vector $c \in \mathbb{R}^n$ as the center and $s > 0$ as the parameter as

$$\forall x \in \mathbb{R}^n, \rho_{s,c}(x) = \exp(-\pi\|x - c\|^2/s^2).$$

For any real $s > 0$ and vector $c \in \mathbb{R}^n$, define the discrete Gaussian distribution over Λ centered at vector c with parameter s as

$$\forall x \in \Lambda, D_{\Lambda,s,c}(x) = \frac{\rho_{s,c}(x)}{\rho_{s,c}(\Lambda)}.$$

When the subscripts s and c are taken to 1 and 0, respectively, they can be omitted.

Definition 6 ([8]). For any full-rank n -dimensional lattice Λ , real $\varepsilon > 0$ and $s > 0$, the smoothing parameter $\eta_\varepsilon(\Lambda)$ is the smallest real s such that $\rho_{1/s}(\Lambda^* \setminus \{0\}) \leq \varepsilon$.

Lemma 2 ([8]). For any full-rank n -dimensional lattice Λ , Gaussian center $c \in \mathbb{R}^n$, real $\varepsilon > 0$, $s \geq 2\eta_\varepsilon(\Lambda)$, and for every $x \in \Lambda$, we have $D_{\Lambda,s,c}(x) \leq \frac{1+\varepsilon}{1-\varepsilon} \cdot 2^{-n}$. In particular, for $\varepsilon < \frac{1}{3}$, the min-entropy of $D_{\Lambda,s,c}(x)$ is at least $n - 1$.

2.2. Lattice Trapdoor Algorithms

Lemma 3 ([15]). For any prime $q = \text{poly}(n)$ and $m = O(n \log q)$, there is a probabilistic polynomial-time algorithm $\text{TrapGen}(1^n, 1^m, q)$ that outputs a matrix $B \in \mathbb{Z}_q^{n \times m}$ and a full-rank set $S \in \Lambda_q^\perp(B)$, where the distribution of B is statistically close to uniform over $\mathbb{Z}_q^{n \times m}$ and the length $\|S\| \leq L = m^{2.5}$.

In particular, the set S can be converted efficiently to a basis $T \in \mathbb{Z}_q^{m \times m}$ of $\Lambda_q^\perp(B)$ such that $\|\tilde{T}\| \leq \|\tilde{S}\| \leq L$, $\tilde{S}$ and $\tilde{T}$ denote the Gram–Schmidt orthogonalization of S and T .

Lemma 4 ([8]). For any prime $q = \text{poly}(n)$, $m \geq 5n \log q$ and a matrix $B \in \mathbb{Z}_q^{n \times m}$, let $T \in \mathbb{Z}_q^{m \times m}$ be a basis for a full-rank n -dimensional lattice $\Lambda_q^\perp(B)$. Then, for a parameter $s > \|\tilde{T}\| \cdot \omega(\sqrt{\log m})$ and a center $c \in \mathbb{R}^m$:

- (1) There is a probabilistic polynomial-time algorithm $\text{SampleD}(T, s, c)$ that outputs a sample from a distribution that is statistically close to $D_{\Lambda_q^\perp, s, c}$.
- (2) There is a probabilistic polynomial-time algorithm $\text{Sample}(B, T, s, u)$ that, on input of a random vector $u \in \mathbb{Z}_q^n$, a matrix $B \in \mathbb{Z}_q^{n \times m}$ and a basis $T \in \mathbb{Z}_q^{m \times m}$, outputs $e \in \Lambda_q^\perp(B)$ sampled from a distribution that is statistically close to $D_{\Lambda_q^\perp, s, c}$.

Lemma 5 ([8]). There is a collection of collision-resistant $\text{Sample}(B, T, s, u)$ if the average-case problem $SIS_{n,m,q,\beta}$ is hard.

Lemma 6 ([8]). For any matrix $B \in \mathbb{Z}_q^{n \times m}$, an efficiently computable function $h_B(x) = Bx \bmod q$ with domain $D_n = \{x \in \mathbb{Z}^m \mid \|x\| \leq s\sqrt{m}\}$, and range $R_n = \mathbb{Z}_q^n$, there is a probabilistic

polynomial-time algorithm $\text{SampleDom}(1^m)$ that outputs an x sampled from some distribution over D_n for which the distribution of $h_B(x)$ is uniform over R_n .

3. Lattice-Based Proxy Signature and System Model

3.1. Formal Definition

In a lattice-based proxy signature scheme, there are four different entities: a trusted key generation center (KGC), an original signer O with ID_O , a proxy signer P with ID_P and a verifier. The scheme consists of the following seven algorithms:

(1) *Setup*: KGC inputs the system security parameter n , outputs the system master key pair (A, T) and the system public parameters $params$, keeps the system secret master key T , and publishes $params$ to all system users.

(2) *KeyExtract*: Given a user's identity ID , KGC generates the key pair (A_{ID}, T_{ID}) for a user's identity ID , and returns (A_{ID}, T_{ID}) to the corresponding user through a security channel.

(3) *Delegate*: The original signer O generates the warrant w , where w describes the delegation relationship, scope and valid period of delegation, and the identities of O and P . The original signer O inputs the warrant w , their key pair, the delegation secret and identity information, and outputs the delegation $W_{O \rightarrow P}$.

(4) *DVerify*: For the proxy signer P , they verify the legality of the delegation information $W_{O \rightarrow P}$. If it is legal, the output is 1 and the delegation is accepted. Otherwise, the output is 0 and the delegation is rejected.

(5) *PKGen*: The proxy signer P generates the delegation secret whether the delegation is accepted. He inputs his key pair, his delegation secret and $W_{O \rightarrow P}$, and outputs the delegation secret key pair (A_{OP}, T_{OP}) .

(6) *PSign*: The proxy signer P inputs the message μ , the delegation $W_{O \rightarrow P}$, the delegation secret key pair (A_{OP}, T_{OP}) and his delegation secret, and outputs the proxy signature e_{OP} of the message μ .

(7) *PVerify*: For arbitrary users, they verify the delegated signature e_{OP} of the message μ with regard to the delegation $W_{O \rightarrow P}$. If it is legal, the output is 1 and the proxy signature is accepted. Otherwise, the output is 0 and the proxy signature is rejected.

3.2. System Model

There are three types of adversary models in the lattice-based proxy signature scheme. Let O be the original signer and P be the proxy signer, as shown in Figure 1.

(1) Type I: The adversary Attack_1 knows the public keys of O and P , and their identity information.

(2) Type II: The adversary Attack_2 knows the key pair of O , the public key of P , and their identity information.

(3) Type III: The adversary Attack_3 knows the key pair of P , the public key of O , and their identity information.

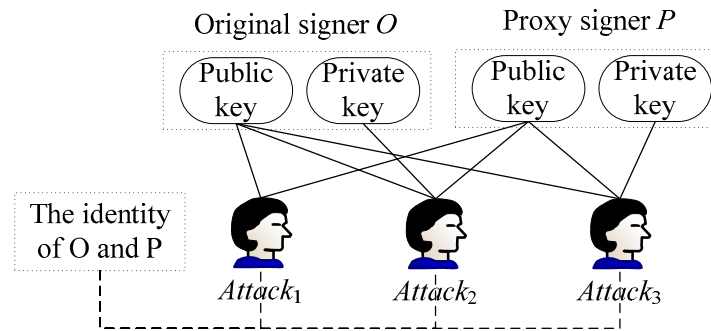


Figure 1. Capabilities of three types of adversaries.

It is clear that, for the above three types of adversary models mentioned, if the lattice-based proxy signature scheme is secure against type II (or type III) adversaries, it will also be secure against type I adversaries. Consequently, the security evaluation of the lattice-based proxy signature scheme presented in this paper primarily focuses on whether the signature scheme demonstrates unforgeability under the type II and type III adversary models. The game interaction process between the challenger and the type II and type III adversaries is shown in Figures 2 and 3.

Game 1: Game 1, that is interactive between the adversary $Attack_2$ and the challenger C , is shown in Figure 2.

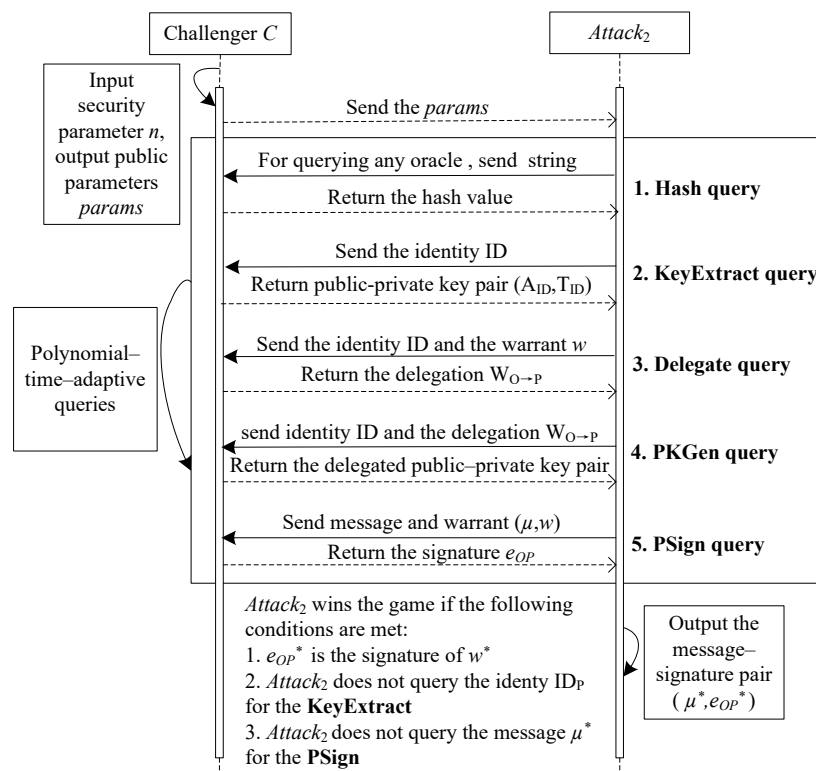


Figure 2. The game between $Attack_2$ and C .

Game 2: Game 2, that is interactive between the adversary $Attack_3$ and the challenger C , is shown in Figure 3.

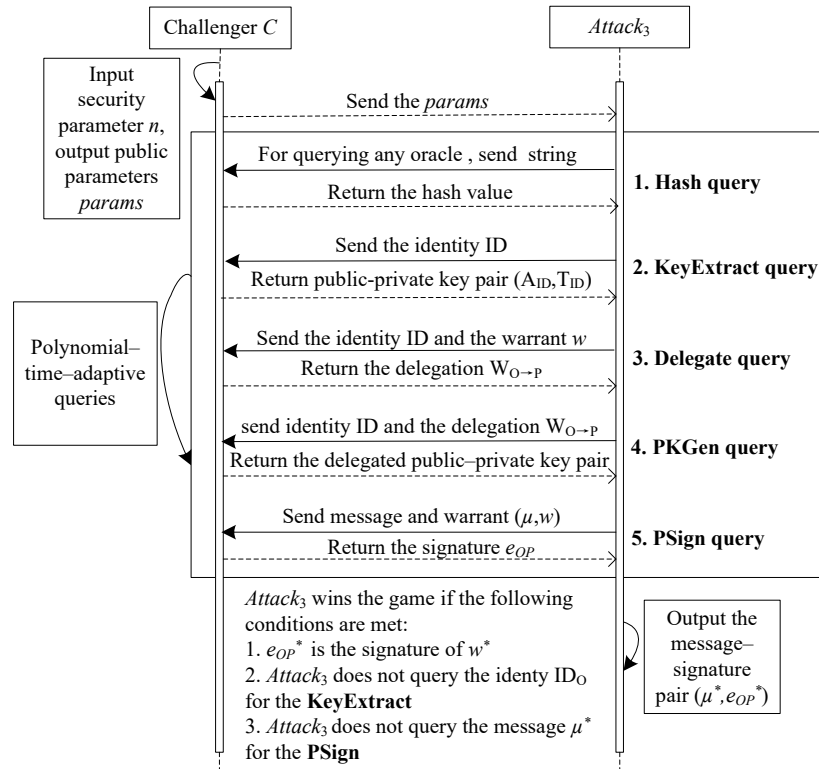


Figure 3. The game between $Attack_3$ and C .

4. Our Scheme

Let O be the original signer and P be the proxy signer. Now, we propose a new lattice-based proxy signature scheme (NLBPS) as follows:

- **Setup:** On inputting the system security parameter n , set $q = \text{poly}(n)$. Set $m = O(n \log q)$ is the dimension of the lattices and $s = \tilde{L}\omega(\sqrt{n \log m})$ is the Gaussian parameter. KGC works as follows:
 1. Choose two secure cryptographic hash functions: $H_1: Z_q^n \times Z_q^n \times \{0,1\}^* \rightarrow Z_q^n$ and $H_2: Z_q^n \times Z_q^n \times \{0,1\}^* \rightarrow Z_q^n$.
 2. Invoke $\text{TrapGen}(1^n, 1^m, q)$ to generate a random matrix $A \in Z_q^{n \times m}$ and a short basis $T \in Z_q^{m \times m}$ for $\Lambda_q^\perp(A)$ such that $\|\tilde{T}\| \leq \tilde{L}$.
 3. Output the master key pair (A, T) and the system public parameters $params = \langle n, m, q, \tilde{L}, s, A, H_1, H_2 \rangle$.
- **KeyExtract:** Given a user's identity ID , KGC invokes $\text{TrapGen}(1^n, 1^m, q)$ to generate the key pair (A_{ID}, T_{ID}) for a user's identity ID , where $A_{ID} \in Z_q^{n \times m}$ is the public key and $T_{ID} \in Z_q^{m \times m}$ is the private key. The original signer O and the proxy signer P both obtain the public-private key pair through this procedure. O and P hold (A_O, T_O) and (A_P, T_P) , respectively.
- **Delegate:** The original signer O generates the warrant $w = \{0,1\}^*$, where w describes the delegation relationship, scope and valid period of delegation, and the identities of O and P . Then, the original signer O works as follows:
 1. Choose $k_O \in Z_q^n$ at random as the authorization token and publish it.
 2. Evaluate $d_O \leftarrow \text{Sampre}(A_O, T_O, s, k_O)$ to obtain $d_O \in Z_q^m$ as his delegation secret.
 3. Compute $h_O = H_1(ID_O \parallel ID_P \parallel w) \in Z_q^n$ and $u_O = A_O d_O + h_O \in Z_q^n$.
 4. Set $e_O \leftarrow \text{Sampre}(A_O, T_O, s, u_O)$. Note that $A_O e_O = u_O \text{ mod } q$ in Z_q^n .

Finally, the original signer O outputs the delegation $W_{O \rightarrow P} = (w, e_O, k_O)$.

- **DVerify**: For the proxy signer P , they verify the legality of the delegation $W_{O \rightarrow P} = (w, e_o, k_o)$ as follows:
 1. Compute $h_o = H_1(ID_o \parallel ID_p \parallel w) \in Z_q^n$.
 2. If $A_o e_o = k_o + h_o$ and $\|e_o\| \leq s\sqrt{m}$, output 1 and accept the delegation $W_{O \rightarrow P}$. Otherwise, output 0 and reject it.
 - **PKGen**: The proxy signer P inputs their key pair (A_o, T_o) , their delegation secret k_o and $W_{O \rightarrow P} = (w, e_o, k_o)$; if the delegation $W_{O \rightarrow P} = (w, e_o, k_o)$ is accepted, they follow the next steps:
 1. Choose $k_p \in Z_q^n$ at random and compute $u_p = k_p + A_o e_o \in Z_q^n$.
 2. Evaluate $d_p \leftarrow \text{Sample}(A_p, T_p, s, u_p)$ to obtain $d_p \in Z_q^m$ as their delegation secret.
- Finally, the proxy signer P outputs the delegated private key $T_{OP} = (T_p, d_p)$ and the delegated public key $A_{OP} = (A_p, A_o, k_p, k_o)$. For arbitrary users, they can determine the legitimacy of the delegation through the delegated public key A_{OP} .
- **PSign**: The proxy signer P inputs the message μ , the delegation $W_{O \rightarrow P}$ and the delegated public–private key pair (A_{OP}, T_{OP}) , and taking the following steps:
 1. Compute $u_{OP} = H_2(k_o \parallel k_p \parallel \mu) + A_p d_p \in Z_q^n$.
 2. Evaluate $e_{OP} \leftarrow \text{Sample}(A_p, T_p, s, u_{OP})$ to obtain the delegated signature of $e_{OP} \in Z_q^m$ of the message μ .
 - **PVerify**: For arbitrary users, they verify the delegated signature $e_{OP} \in Z_q^m$ of the message μ as follows:
 1. Compute $h_o = H_1(ID_o \parallel ID_p \parallel w)$.
 2. If $A_p e_{OP} = H_2(k_o \parallel k_p \parallel \mu) + k_o + k_p + h_o$ and $\|e_{OP}\| \leq s\sqrt{m}$, accept the signature and output 1. Otherwise, output 0 and reject the signature.

5. Scheme Analysis

5.1. The Correctness of NLBPS

1. For the **DVerify** algorithm, the correctness of the verification equation is elaborated as follows:

$$A_o e_o = u_o = A_o d_o + h_o = k_o + h_o.$$

2. For the **PVerify** algorithm, the correctness of the verification equation is elaborated as follows:

$$\begin{aligned} A_p e_{OP} &= u_{OP} \\ &= H_2(k_o \parallel k_p \parallel \mu) + u_p \\ &= H_2(k_o \parallel k_p \parallel \mu) + k_p + A_o e_o \\ &= H_2(k_o \parallel k_p \parallel \mu) + k_p + u_o \\ &= H_2(k_o \parallel k_p \parallel \mu) + k_p + A_o d_o + h_o \\ &= H_2(k_o \parallel k_p \parallel \mu) + k_p + k_o + h_o \end{aligned}$$

5.2. The Security Analysis of NLBPS

Our scheme security consists of three types of adversaries; if NLBPS has existential unforgeability against type II and type III adversaries, it will also have existential unforgeability against type I adversaries. Therefore, we only take two types of adversaries into consideration.

Theorem 1. *If the $SIS_{n,m,q,2\sqrt{m}}$ problem is hard, the proposed lattice-based proxy signature scheme NLBPS is existential unforgeable against an adaptive chosen message and identity attacks in the random oracle model.*

Proof of Theorem 1. We prove this theorem by contradiction. Assuming that an adversary breaks the existential unforgeability of the signature scheme with a non-negligible probability, then we can construct a challenger to solve the problem $SIS_{n,m,q,2\sqrt{m}}$ by running the adversary as a subroutine. $\square$

Corollary 1. Let Q_{H_i} be the maximum number of queries that $Attack_2$ makes to the random oracle H_i at a single query time for $i = 1, 2$. Define Q_k , Q_p , Q_{pk} and Q_s as the maximum number of **KeyExtract** queries, **Delegate** queries, **PKGen** queries, and **PSign** queries, respectively. t_k , t_p , t_{pk} and t_s correspond to the time for each **KeyExtract** query, **Delegate** query, **PKGen** query, and **PSign** query, respectively. If there is a type II adversary $Attack_2$ that can break the existential unforgeability of NLBPS in the random oracle model with a non-negligible probability ε within a time bound t , then there exists a polynomial-time challenger C that solves the problem $SIS_{n,m,q,2\sqrt{m}}$ with a non-negligible probability:

$$\varepsilon' \geq \frac{\varepsilon}{Q_{H_2}} \left(1 - \frac{1}{2^m}\right) \left(1 - \frac{1}{Q_k}\right) \left(1 - \frac{1}{Q_{pk}Q_{H_2}}\right)^{Q_s}.$$

This solution is achievable within a time constraint of

$$t' < t + (Q_k t_k + Q_p t_p + Q_{pk} t_{pk} + Q_s t_s + Q_{H_1} t_{H_1} + Q_{H_2} t_{H_2})$$

Proof of Corollary 1. The type II adversary $Attack_2$ knows the key pair of O , the public key of P , and the identity information of both agents. First of all, $Attack_2$ announces to C the identity ID_P^* and the message μ^* that will be challenged.

The challenger C receives an instance of the problem $SIS_{n,m,q,2\sqrt{m}}$. Given $q \in N^+$ and a uniformly random matrix $A_p^* \in Z_q^{n \times m}$, C tries to find small non-zero vector $e \in Z^m$ such that $A_p^* e = 0 \bmod q$ and $\|e\| \leq 2s\sqrt{m}$.

Setup: On inputting the system security parameter n , set $q = \text{poly}(n)$. Set $m = O(n \log q)$ as the dimension of the lattices and $s = \tilde{L}\omega(\sqrt{n \log m})$ as the Gaussian parameter. C invokes $\text{TrapGen}(1^n, 1^m, q)$ to generate a random matrix $A \in Z_q^{n \times m}$ and a basis $T \in Z_q^{m \times m}$ for $\Lambda_q^\perp(A)$ such that $\|T\| \leq \tilde{L}$. C maintains five lists— L_1 , L_2 , LD , LP , LS —that are initially empty. C sets the master key pair (A, T) and system public parameters $\text{params} = (n, m, q, \tilde{L}, s, A, H_1, H_2)$, and sends params to $Attack_2$.

Random oracle hash queries: We assume that the $Attack_2$'s queries are distinct; otherwise, the simulator will consistently produce the same output for identical inputs without increasing the query counter. Then, $Attack_2$ performs the following queries:

1. **H_1 query:** C maintains a list L_1 of tuples (ID_O, ID_P, w, h_1) . For the query to L_1 , if (ID_O, ID_P, w) is in the list L_1 , then C returns h_1 to $Attack_2$. Otherwise, C chooses $h_1 \in Z_q^n$ at random, adds it to list L_1 and then returns it to $Attack_2$.
2. **H_2 query:** C maintains a list of tuples $(k_1, h_1, \mu, A_p, d_2, k_2, h_2, e_{OP})$ which is called the L_2 list. For the query on (k_1, h_1, μ, A_p) , C searches $(k_1, h_1, \mu, A_p, d_2, k_2, h_2, e_{OP})$ in the list L_2 . If it exists, C returns (k_2, h_2, d_2, e_{OP}) to $Attack_2$. Otherwise,
 - (a) If $ID_P = ID_P^*$, C invokes $d_2 \leftarrow \text{SampleDom}(1^m)$ and $e_{OP} \leftarrow \text{SampleDom}(1^m)$, then computes $h_2 = A_p e_{OP} - A_p d_2$ and $k_2 = A_p d_2 - k_1 - h_1$.
 - (b) If $ID_P \neq ID_P^*$, C chooses $k_2 \in Z_q^n$ at random, computes $u_p = k_2 + A_o e_1 \in Z_q^n$ and runs $\text{Sample}(A_p, T_p, s, u_p)$ to generate $d_2 \in Z_q^m$. Then, C chooses $h_2 \in Z_q^n$ at random, computes $u_{OP} = h_2 + A_p d_2 \in Z_q^n$ and runs $\text{Sample}(A_p, T_p, s, u_{OP})$ to generate $e_{OP} \in Z_q^m$.

Finally, C returns (k_2, h_2, d_2, e_{OP}) to $Attack_2$ and adds $(k_1, h_1, \mu, A_p, d_2, k_2, h_2, e_{OP})$ to list L_2 .

KeyExtract query: For the query on the identity ID , if $ID = ID_P^*$, C returns $\perp$ to $Attack_2$, then declares failure and halts. This event is recorded as $Event_1$. Otherwise, if

$ID \neq ID_P^*$, C invokes $TrapGen(1^n, 1^m, q)$ to return the corresponding key pair (A_{ID}, T_{ID}) to $Attack_2$.

Delegate query: C maintains a list LD of tuples $(ID_O, ID_P, w, e_1, k_1, h_1)$. $Attack_2$ submits (ID_O, ID_P, w) to C for querying the delegation. C firstly calls H_1 query to obtain (ID_O, ID_P, w, h_1) . Then, C looks into list LD ; if it can find the corresponding $(ID_O, ID_P, w, e_1, k_1, h_1)$ in LD , C returns (w, e_1, k_1, h_1) to $Attack_2$. Otherwise, C chooses $k_1 \in \mathbb{Z}_q^n$ at random, and runs algorithm $Sampre(A_O, T_O, s, k_1)$ to generate $d_1 \in \mathbb{Z}_q^m$. Then, C returns (w, e_1, k_1, h_1) to $Attack_2$ and adds $(ID_O, ID_P, w, e_1, k_1, h_1)$ to list LD .

PKGen query: C maintains a list LP of tuples $(ID_O, ID_P, e_1, k_2, T_P^*, d_2)$. For the query on (ID_O, ID_P, e_1) , C searches $(ID_O, ID_P, e_1, k_2, T_P^*, d_2)$ in the list LP . If it exists, C returns $T_{OP} = (T_P^*, d_2)$ to $Attack_2$. Otherwise, C calls **Delegate** query to obtain (w, e_1, k_1, h_1) and calls H_2 query to obtain (k_2, h_2, d_2, e_{OP}) .

1. If $ID_P = ID_P^*$, let $T_P^* = T_O$; the delegated private key and the corresponding public key are $T_{OP} = (T_P^*, d_2)$ and $A_{OP} = (A_P, A_O, k_1, k_2)$, respectively.
2. If $ID_P \neq ID_P^*$, let $T_P^* = T_P$; the delegated private key and the corresponding public key are $T_{OP} = (T_P^*, d_2)$ and $A_{OP} = (A_P, A_O, k_1, k_2)$, respectively.

Finally, C publishes the delegated public key $A_{OP} = (A_P, A_O, k_1, k_2)$, and returns $T_{OP} = (T_P^*, d_2)$ to $Attack_2$. Then, C adds $(ID_O, ID_P, e_1, k_2, T_P^*, d_2)$ to list LP .

PSign query: C maintains a list LS of tuples $(ID_O, ID_P, \mu, w, e_{OP})$. $Attack_2$ submits (ID_O, ID_P, μ, w) to C for querying the signature.

1. If $ID_P = ID_P^*$ and $\mu = \mu^*$, C returns $\perp$ to $Attack_2$, then declares failure and halts. This event is recorded as $Event_2$.
2. If $ID_P = ID_P^*$ and $\mu \neq \mu^*$, C searches $(ID_O, ID_P, \mu, w, e_{OP})$ in the list LS . If it exists, C returns (μ, e_{OP}) to $Attack_2$. Otherwise, C calls **Delegate** query to obtain (w, e_1, k_1, h_1) , calls **PKGen** query to obtain (k_2, T_P^*, d_2) and calls H_2 query to obtain (k_2, h_2, d_2, e_{OP}) . Then, C returns (μ, e_{OP}) to $Attack_2$ and adds $(ID_O, ID_P, e_1, T_P^*, d_2)$ to list LS .
3. If $ID_P \neq ID_P^*$, C searches $(ID_O, ID_P, \mu, w, e_{OP})$ in the list LS . If it exists, C returns (μ, e_{OP}) to $Attack_2$. Otherwise, C calls **Delegate** query to obtain (w, e_1, k_1, h_1) , calls **PKGen** query to obtain (k_2, T_P^*, d_2) and calls H_2 query to obtain (k_2, h_2, d_2, e_{OP}) . Then, C returns (μ, e_{OP}) to $Attack_2$ and adds $(ID_O, ID_P, e_1, T_P^*, d_2)$ to list LS .

If $Event_1$ and $Event_2$ do not occur, then C 's simulation is perfect. The adversary $Attack_2$ forges a valid signature pair (μ^*, e_{OP}^*) of the tuple (ID_O, ID_P, μ^*, w) with a non-negligible probability ϵ . This forgery satisfies the condition $\|e_{OP}^*\| \leq s\sqrt{m}$, and the equation $A_P e_{OP}^* = H_2(k_1 \parallel k_2 \parallel \mu^*) + k_1 + k_2 + \hat{h}_1$. Thus, the following equation holds:

$$\begin{aligned} A_P e_{OP}^* &= H_2(k_O \parallel k_P \parallel \mu^*) + k_1 + k_2 + h_1 \\ &= A_P e_{OP} - A_P d_2 + k_1 + k_2 + h_1 \\ &= A_P e_{OP} - A_P d_2 + k_1 + (A_P d_2 - k_1 - h_1) + h_1 \\ &= A_P e_{OP} \end{aligned}$$

Therefore, we obtain that $A_P e_{OP} - A_P e_{OP}^* = A_P (e_{OP} - e_{OP}^*) = 0 \bmod q$. According to Lemma 2, the minimum entropy of the preimage of the *Sampre* algorithm is at least $n - 1$. Consequently, the probability that the equation $e_{OP} = e_{OP}^*$ is negligible. Furthermore, given that $\|e_{OP}\| \leq s\sqrt{m}$ and $\|e_{OP}^*\| \leq s\sqrt{m}$, it follows that $\|e_{OP} - e_{OP}^*\| \leq \|e_{OP}\| + \|e_{OP}^*\| \leq 2s\sqrt{m}$. Let $A_P^* = A_P$, then the challenger C can output $e_{OP} - e_{OP}^*$ as a solution to the instance of the problem $SIS_{n,m,q,2\sqrt{m}}$.

If $Event_1$ and $Event_2$ do not occur, **Game 1** proceeds normally. The probability of **Game 1** running without these events is as follows:

$$\Pr(\neg \text{Event}_1 \wedge \neg \text{Event}_2) \geq \left(1 - \frac{1}{Q_k}\right) \left(1 - \frac{1}{Q_{pk}Q_{H_2}}\right)^{Q_s}.$$

In the event that the adversary Attack_2 forges a valid signature without performing adaptive queries, the probability of such an occurrence is $\frac{1}{2^m}$. Consequently, if Attack_2 forges the signature with a non-negligible probability ε in time t through adaptive queries, then there exists a challenger C who, within a polynomial time $t' < t + (Q_k t_k + Q_p t_p + Q_{pk} t_{pk} + Q_s t_s + Q_{H_1} t_{H_1} + Q_{H_2} t_{H_2})$, can solve the problem $\text{SIS}_{n,m,q,2\sqrt{m}}$ with a non-negligible probability ε' given by

$$\varepsilon' \geq \frac{\varepsilon}{Q_{H_2}} \left(1 - \frac{1}{2^m}\right) \left(1 - \frac{1}{Q_k}\right) \left(1 - \frac{1}{Q_{pk}Q_{H_2}}\right)^{Q_s}.$$

Corollary 2. Let Q_{H_i} be the maximum number of queries that Attack_3 makes to the random oracle H_i at a single query time for $i = 1, 2$. Define Q_k , Q_p , Q_{pk} and Q_s as the maximum number of **KeyExtract** queries, **Delegate** queries, **PKGen** queries, and **PSign** queries, respectively. t_k , t_p , t_{pk} and t_s correspond to the time for each **KeyExtract** query, **Delegate** query, **PKGen** query, and **PSign** query, respectively. If there is a type III adversary Attack_3 that can break the existential unforgeability of NLBPS in the random oracle model with a non-negligible probability ε within a time bound t , then there exists a polynomial-time challenger C that solves the problem $\text{SIS}_{n,m,q,2\sqrt{m}}$ with a non-negligible probability

$$\varepsilon' \geq \frac{\varepsilon}{Q_{H_2}} \left(1 - \frac{1}{2^m}\right) \left(1 - \frac{1}{Q_k}\right) \left(1 - \frac{1}{Q_p Q_{H_1}}\right)^{Q_s + Q_{H_2}}.$$

This solution is achievable within a time constraint of

$$t' < t + (Q_k t_k + Q_p t_p + Q_s t_s + Q_{H_1} t_{H_1} + Q_{H_2} t_{H_2}). \quad \square$$

Proof of Corollary 2. The type III adversary Attack_3 knows the key pair of O , the public key of P , and the identity information of both agents. First of all, Attack_3 announces to C the identity ID_O^* and the message μ^* that will be challenged.

The challenger C receives an instance of the problem $\text{SIS}_{n,m,q,2\sqrt{m}}$. Given $q \in \mathbb{N}^+$ and a random matrix $A_p^* \in \mathbb{Z}_q^{n \times m}$, C tries to find a small non-zero vector $e \in \mathbb{Z}^m$ such that $A_p^* e = 0 \bmod q$ and $\|e\| \leq 2s\sqrt{m}$.

Setup: On inputting the system security parameter n , set $q = \text{poly}(n)$. Set $m = O(n \log q)$ as the dimension of the lattices and $s = \tilde{L}\omega(\sqrt{n \log m})$ as the Gaussian parameter. C invokes $\text{TrapGen}(1^n, 1^m, q)$ to generate a random matrix $A \in \mathbb{Z}_q^{n \times m}$ and a basis $T \in \mathbb{Z}_q^{m \times m}$ for $\Lambda_q^\perp(A)$ such that $\|T\| \leq \tilde{L}$. C maintains five lists— L_1 , L_2 , LD , LP , LS —that are initially empty. C sets the master key pair (A, T) and system public parameters $\text{params} = \langle n, m, q, \tilde{L}, s, A, H_1, H_2 \rangle$ and sends params to Attack_3 .

Random oracle hash queries: We assume that the Attack_3 's queries are distinct; otherwise, the simulator will consistently produce the same output for identical inputs without increasing the query counter. Then, Attack_2 performs the following queries:

1. **H_1 query:** C maintains a list L_1 of tuples $(ID_O, ID_P, w, A_O, k_1, e_1, h_1)$. For the query to L_1 , if $(ID_O, ID_P, w, A_O, k_1, e_1)$ is in the list L_1 , then C returns h_1 to Attack_3 . Otherwise,
 - (a) If $ID_O = ID_O^*$, C computes $h_1 = A_O e_1 - k_1 \in \mathbb{Z}_q^n$.
 - (b) If $ID_O \neq ID_O^*$, C randomly chooses $h_1 \in \mathbb{Z}_q^n$.

Finally, C returns h_1 to Attack_3 and adds $(ID_O, ID_P, w, A_O, k_1, e_1, h_1)$ to list L_1 .

2. **H_2 query:** C maintains a list of tuples (k_1, k_2, μ, h_2) which is called the L_2 list. For the query on (k_1, k_2, μ) , C searches (k_1, k_2, μ, h_2) in the list L_2 . If it exists, C

returns h_2 to $Attack_3$. Otherwise, C chooses $h_2 \in Z_q^n$ at random, returns h_2 to $Attack_3$ and adds (k_1, k_2, μ, h_2) to list L_2 .

KeyExtract query: For the query on the identity ID , if $ID = ID_0^*$, C returns $\perp$ to $Attack_3$, then declares failure and halts. This event is recorded as $Event_1$. Otherwise, if $ID \neq ID_0^*$, C invokes $TrapGen(1^n, 1^m, q)$ to return the corresponding key pair (A_{ID}, T_{ID}) to $Attack_3$.

Delegate query: C maintains a list LD of tuples $(ID_0, ID_P, w, e_1, k_1)$. $Attack_3$ submits (ID_0, ID_P, w) to C for querying the delegation. Then, C looks into list LD ; if it can find the corresponding $(ID_0, ID_P, w, e_1, k_1)$ in LD , C returns (w, e_1, k_1) to $Attack_3$. Otherwise,

1. If $ID_0 = ID_0^*$, C invokes $e_1 \leftarrow SampreDom(1^m)$ and chooses $k_1 \in Z_q^n$ at random. Then, C calls H_1 query to obtain $h_1 \in Z_q^n$.
2. If $ID_0 \neq ID_0^*$, C randomly chooses $k_1 \in Z_q^n$, and runs algorithm $Sampre(A_0, T_0, s, k_1)$ to generate $d_1 \in Z_q^m$ as the delegated secret. Then, C calls H_1 query to obtain $h_1 \in Z_q^n$, computes $u_0 = A_0 d_0 + h_1 \in Z_q^n$ and invokes $e_1 \leftarrow Sampre(A_0, T_0, s, u_0)$.

Finally, C returns (w, d_1, e_1, k_1) to $Attack_3$ and adds $(ID_0, ID_P, w, e_1, k_1)$ to list LD .

PKGen query: C maintains a list LP of tuples $(ID_0, ID_P, e_1, T_P, d_2)$. For the query on (ID_0, ID_P, e_1) , C searches $(ID_0, ID_P, e_1, T_P, d_2)$ in the list LP . If it exists, C returns $T_{OP}^* = (T_P, d_2)$ to $Attack_3$. Otherwise, C randomly chooses a matrix $k_2 \in Z_q^n$, computes $u_2 = k_2 + A_0 e_1 \in Z_q^n$ and invokes $d_2 \leftarrow Sampre(A_P, T_P, s, u_2)$. Finally, C publishes the delegated public key $A_{OP}^* = (A_P, A_0, k_1, k_2)$ and returns $T_{OP}^* = (T_P, d_2)$ to $Attack_3$. Then, C adds $(ID_0, ID_P, e_1, k_2, T_P^*, d_2)$ to list LP .

PSign query: C maintains a list LS of tuples $(ID_0, ID_P, \mu, w, e_{OP})$. $Attack_3$ submits (ID_0, ID_P, μ, w) to C for querying the signature.

1. If $ID_0 = ID_0^*$ and $\mu = \mu^*$, C returns $\perp$ to $Attack_3$, then declares failure and halts. This event is recorded as $Event_2$.
2. If $ID_0 = ID_0^*$ and $\mu \neq \mu^*$, C searches $(ID_0, ID_P, \mu, w, e_{OP})$ in the list LS . If it exists, C returns (μ, e_{OP}) to $Attack_3$. Otherwise, C calls **Delegate** query to obtain (w, e_1, k_1) , calls **PKGen** query to obtain (e_1, T_P, d_2) and calls H_2 query to obtain (k_1, k_2, μ, h_2) . Then, C computes $u_{OP} = h_2 + A_P d_2 \in Z_q^n$ and invokes $e_{OP} \leftarrow Sampre(A_P, T_P, s, u_{OP})$. Finally, C returns (μ, e_{OP}) to $Attack_3$ and adds $(ID_0, ID_P, \mu, w, e_{OP})$ to list LS .
3. If $ID_0 \neq ID_0^*$, C searches $(ID_0, ID_P, \mu, w, e_{OP})$ in the list LS . If it exists, C returns (μ, e_{OP}) to $Attack_3$. Otherwise, C calls **Delegate** query to obtain (w, e_1, k_1) , calls **PKGen** query to obtain (e_1, T_P, d_2) , and calls H_2 query to obtain (k_1, k_2, μ, h_2) . Then, C computes $u_{OP} = h_2 + A_P d_2 \in Z_q^n$ and invokes $e_{OP} \leftarrow Sampre(A_P, T_P, s, u_{OP})$. Then, C returns (μ, e_{OP}) to $Attack_3$ and adds $(ID_0, ID_P, \mu, w, e_{OP})$ to list LS .

If $Event_1$ and $Event_2$ do not occur, then C 's simulation is perfect. The adversary $Attack_3$ forges a valid signature pair (μ^*, e_{OP}^*) of the tuple (ID_0, ID_P, μ^*, w) with a non-negligible probability ε . This forgery satisfies the condition $\|e_{OP}^*\| \leq s\sqrt{m}$, and the equation $A_P e_{OP}^* = H_2(k_0 \parallel k_P \parallel \mu^*) + k_1 + k_2 + h_1$. Thus, the following equation holds:

$$\begin{aligned}
 A_P e_{OP}^* &= H_2(k_0 \parallel k_P \parallel \mu^*) + k_1 + k_2 + h_1 \\
 &= H_2(k_0 \parallel k_P \parallel \mu^*) + k_1 + k_2 + (A_0 e_1 - k_1) \\
 &= H_2(k_0 \parallel k_P \parallel \mu^*) + k_2 + A_0 e_1 \\
 &= H_2(k_0 \parallel k_P \parallel \mu^*) + A_0 d_2 \\
 &= A_P e_{OP}
 \end{aligned}$$

Therefore, we obtain $A_P e_{OP} - A_P e_{OP}^* = A_P (e_{OP} - e_{OP}^*) = 0 \bmod q$. According to Lemma 2, the minimum entropy of the preimage of the *Sampre* algorithm is at least $n - 1$. The probability that $e_{OP} = e_{OP}^*$ is negligible. Furthermore, given that $\|e_{OP}\| \leq s\sqrt{m}$ and $\|e_{OP}^*\| \leq s\sqrt{m}$, it follows that $\|e_{OP} - e_{OP}^*\| \leq \|e_{OP}\| + \|e_{OP}^*\| \leq 2s\sqrt{m}$. Let $A_P^* = A_P$; then, the challenger C can output $e_{OP} - e_{OP}^*$ as a solution to the instance of the problem $SIS_{n,m,q,2\sqrt{m}}$.

If $Event_1$ and $Event_2$ do not occur, **Game 2** proceeds normally. The probability of **Game 2** running without these events is as follows:

$$Pr(\neg Event_1 \wedge \neg Event_2) \geq \left(1 - \frac{1}{Q_k}\right) \left(1 - \frac{1}{Q_p Q_{H_1}}\right)^{Q_s + Q_{H_2}}.$$

In the event that the adversary $Attack_3$ forges a valid signature without performing adaptive queries, the probability of such an occurrence is $\frac{1}{2^m}$. Consequently, if $Attack_3$ forges the signature with a non-negligible probability ε in time t through adaptive queries, then there exists a challenger C who, within a polynomial time $t' < t + (Q_k t_k + Q_p t_p + Q_s t_s + Q_{H_1} t_{H_1} + Q_{H_2} t_{H_2})$, can solve the problem $SIS_{n,m,q,2\sqrt{m}}$ with a non-negligible probability ε' given by

$$\varepsilon' \geq \frac{\varepsilon}{Q_{H_2}} \left(1 - \frac{1}{2^m}\right) \left(1 - \frac{1}{Q_k}\right) \left(1 - \frac{1}{Q_p Q_{H_1}}\right)^{Q_s + Q_{H_2}}. \quad \square$$

Furthermore, Boneh et al. [16] introduced the concept of a history-free reduction, and prove that such reductions imply security in the quantum model. They showed that the GPV scheme has a history-free reduction, thereby establishing its post-quantum security. Specifically, the GPV scheme is secure in the quantum random oracle model (QROM), where a quantum adversary can issue a query to the random oracle that represents a superposition of an exponentially large number of states. Our proposed NLBPS is a variant of the GPV scheme, built upon the same foundational principles. This ensures that NLBPS maintains its security in the presence of quantum attacks. The detailed proof process is not reiterated here.

5.3. The Performance Comparison of NLBPS

In this section, our scheme is compared with other proxy signature schemes in terms of computation costs and storage size. Specifically, the scheme presented in [10] is a lattice-based proxy signature scheme, the scheme in [13] describes a lattice-based proxy signature scheme in cloud storage, and the scheme in [17] constructs an attribute-based proxy signing scheme. All three schemes involve the delegation of signing authority to a proxy. The comparison results are summarized in Table 1 and define some notations as follows: T_{Ha} denotes the execution time of a generic hash function operation, while T_{Sam} denotes the execution time of *Sampre* algorithm operation. Additionally, V_T , V_S , V_B , and V_O represent the *TrapGen* algorithm, the *Sampre* algorithm, the basis delegation without dimension increase algorithm [9], and other algorithms, respectively.

Table 1. Performance comparison among Refs. [10,13,17] and ours.

Scheme	[10]	[13]	[17]	Our Scheme
Signing cost	$2T_{Ha} + 2T_{Sam}$	$T_{Ha} + T_{Sam}$	$T_{Ha} + T_{Sam}$	$T_{Ha} + T_{Sam}$
Verification cost	$2T_{Ha}$	$(n + k + 2)T_{Ha}$	$3T_{Ha}$	$2T_{Ha}$
The size of signature	$2m \log q$	$(2m + n) \log q$	$2m \log q$	$m \log q$
The size of delegated key	$2m^2 \log q$	$2m^2 \log q$	$2m^2 \log q$	$m^2 \log q$
The number of algorithms	$V_T + V_S + V_B$	$V_T + V_S + 2V_O$	$V_T + V_S + V_B$	$V_T + V_S$

The comparison of computation costs and storage size are listed in Table 1. The computation costs include signing cost and verification cost. Our scheme offers computational improvements over the methods presented in [10]. Specifically, our proxy signing process requires less time by omitting one execution of the hash function and the *Sampre* algorithm, thus saving $T_{Ha} + T_{Sam}$. The signature verification also saves time by eliminating one hash function execution, saving T_{Ha} . The storage size includes the size of signature and the delegated secret key. Compared with the schemes in [10], our scheme has the shortest storage size. The size of signature and the delegated secret key of our scheme are $m \log q$ and $m^2 \log q$, respectively. Furthermore, our scheme requires fewer algorithms, including only the *TrapGen* and *Sampre* algorithms, which makes it more suitable for resource-limited applications such as the Internet of Things (IOT).

6. Conclusions

In this paper, we present a new lattice-based proxy signature scheme (NLBPS) that combines trapdoor sampling algorithms with preimage sampling algorithms. Utilizing a trapdoor sampling algorithm [15], we generate public–private key pairs for both the original signer and the proxy signer. Subsequently, proxy authorization and signature messages are carried out through the application of preimage sampling algorithms. Based on the hardness of the small integer solution (SIS) problem, we present the security analysis of NLBPS in the random oracle model. The performance comparison shows that our scheme has some advantages in computation costs and storage size. In the future, this section would be followed by a detailed discussion of the security proof and a comparative analysis with existing schemes, including a description of the computational complexity and potential areas for future research. The preimage sampling algorithms employed in the proxy signature process depend on the trapdoor basis to obtain the preimage, which consequently incurs high computational costs. In the future, we will consider using proxy signature without trapdoor as a basis for implementing a lattice-based proxy signature scheme.

Author Contributions: Conceptualization, L.Z. and J.L.; data curation, J.L.; formal analysis, H.L.; investigation, J.L.; methodology, L.Z. and J.L.; project administration, L.Z. and J.L.; software, J.L. and H.L.; supervision, L.Z.; validation, J.L. and H.L.; visualization, J.L. and H.L.; writing—original draft, J.L.; writing—review and editing, J.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: No new data were created.

Acknowledgments: The authors of this paper would like to express their sincere gratitude to the editors and reviewers for their hard work.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Mambo, M.; Usuda, K.; Okamoto, E. Proxy Signatures for Delegating Signing Operation. In Proceedings of the Proceedings of the 3rd ACM Conference on Computer and Communications Security, New Delhi, India, 14–15 March 1996; Association for Computing Machinery: New York, NY, USA, 1996; pp. 48–57.
2. Boldyreva, A.; Palacio, A.; Warinschi, B. Secure Proxy Signature Schemes for Delegation of Signing Rights. *J. Cryptol.* **2012**, *25*, 57–115.
3. Shamir, A. Identity-Based Cryptosystems and Signature Schemes. In *Proceedings of the Advances in Cryptology: Proceedings of CRYPTO 84 4*; Springer: Berlin/Heidelberg, Germany, 1985; pp. 47–53.

4. Chen, G. *Parallel and Distributed Processing and Applications-ISPA 2005 Workshops: ISPA 2005 International Workshops, AEPP, ASTD, BIOS, GCIC, IADS, MASN, SGCA, and WISA, Nanjing, China, 2–5 November 2005, Proceedings*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2005; Volume 3759.
5. Bannore, A.; Patil, R.Y.; Patil, Y.H.; Deshpande, H. Proxy Signature-Based Role Delegation Scheme: Formal Analysis and Simulation. *Int. J. Inf. Technol.* **2024**, *16*, 4027–4038.
6. Shor, P.W. Polynomial-Time Algorithms for Prime Factorization and Discrete Logarithms on a Quantum Computer. *SIAM Rev.* **1999**, *41*, 303–332.
7. Bernstein, D.J.; Lange, T. Post-Quantum Cryptography. *Nature* **2017**, *549*, 188–194.
8. Gentry, C.; Peikert, C.; Vaikuntanathan, V. Trapdoors for Hard Lattices and New Cryptographic Constructions. In Proceedings of the Fortieth Annual ACM Symposium on Theory of Computing, Victoria, BC, Canada, 17–20 May 2008; ACM: New York, NY, USA, 2008; pp. 197–206.
9. Agrawal, S.; Boneh, D.; Boyen, X. Lattice Basis Delegation in Fixed Dimension and Shorter-Ciphertext Hierarchical IBE. In *Advances in Cryptology—CRYPTO 2010*; Rabin, T., Ed.; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2010; Volume 6223, pp. 98–115, ISBN 978-3-642-14622-0.
10. Kim, K.S.; Hong, D.; Jeong, I.R. Identity-Based Proxy Signature from Lattices. *J. Commun. Netw.* **2013**, *15*, 1–7, doi:10.1109/JCN.2013.000003.
11. Wu, F.; Yao, W.; Zhang, X.; Wang, W.; Zheng, Z. Identity-based Proxy Signature over NTRU Lattice. *Int. J. Commun. Syst.* **2019**, *32*, e3867, doi:10.1002/dac.3867.
12. Luo, F.; Al-Kuwari, S.; Susilo, W.; Duong, D.H. Attribute-Based Proxy Re-Signature from Standard Lattices and Its Applications. *Comput. Stand. Interfaces* **2021**, *75*, 103499, doi:10.1016/j.csi.2020.103499.
13. Wang, Q.; Cheng, C.; Xu, R.; Ding, J.; Liu, Z. Analysis and Enhancement of a Lattice-Based Data Outsourcing Scheme With Public Integrity Verification. *IEEE Trans. Serv. Comput.* **2022**, *15*, 2226–2231, doi:10.1109/TSC.2020.3041324.
14. Yu, H.; Wang, N. Certificateless Network Coding Proxy Signatures from Lattice. *Front. Comput. Sci.* **2023**, *17*, 175810, doi:10.1007/s11704-022-2128-z.
15. Alwen, J.; Peikert, C. Generating Shorter Bases for Hard Random Lattices. *Theory Comput. Syst.* **2011**, *48*, 535–553, doi:10.1007/s00224-010-9278-3.
16. Boneh, D.; Dagdelen, Ö.; Fischlin, M.; Lehmann, A.; Schaffner, C.; Zhandry, M. Random Oracles in a Quantum World. In Proceedings of the Advances in Cryptology—ASIACRYPT 2011: 17th International Conference on the Theory and Application of Cryptology and Information Security, Seoul, South Korea, 4–8 December 2011; Proceedings 17; Springer: Berlin/Heidelberg, Germany, 2011; pp. 41–69.
17. Wei, L.; Li, D.; Liu, Z. Provable Secure Attribute-Based Proxy Signature Over Lattice Small Integer Solution Problem in Random Oracle Model. *Electronics* **2023**, *12*, 1619, doi:10.3390/electronics12071619.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.