LaSDVS : A Post-Quantum Secure Compact Strong-Designated Verifier Signature

Shanu Poddar[1], Sweta Mishra[1], Tapaswini Mohanty[2], Vikas Srivastava[3*], Sugata Gangopadhyay[2]

[1]Department of Computer Science and Engineering, Shiv Nadar Institute of Eminence, NCR, 201314, Delhi, India.
[2]Department of Computer Science and Engineering, Indian Institute of Technology Roorkee, Roorkee, 247667, Uttarakhand, India.
[3*]Department of Mathematics, Indian Institute of Technology Madras, Chennai, 600036, Tamil Nadu, India.

*Corresponding author(s). E-mail(s): vikas.math123@gmail.com;
Contributing authors: shanupoddar2112@gmail.com;
sweta.mishra@snu.edu.in; mtapaswini37@gmail.com;
sugata.gangopadhyay@cs.iitr.ac.in ;

Abstract

Digital signatures are fundamental cryptographic primitives that ensure the authenticity and integrity of digital communication. However, in scenarios involving sensitive interactions—such as e-voting or e-cash—there is a growing need for more controlled signing mechanisms. *Strong-Designated Verifier Signature* (SDVS) offers such control by allowing the signer to specify and restrict the verifier of a signature. The existing state-of-the-art SDVS are mostly based on number-theoretic hardness assumptions. Thus, they are not secure against quantum attacks. Moreover, Post-Quantum Cryptography (PQC)-based SDVS are inefficient and have large key and signature sizes. In this work, we address these challenges and propose an efficient post-quantum SDVS (namely, LaSDVS) based on ideal lattices under the hardness assumptions of the Ring-SIS and Ring-LWE problems. LaSDVS achieves advanced security properties including strong unforgeability under chosen-message attacks, non-transferability, non-delegatability, and signer anonymity. By employing the algebraic structure of rings and the *gadget trapdoor* mechanism of Micciancio et al., we design LaSDVS to minimize computational overhead and significantly reduce key and signature sizes. Notably, our scheme achieves a compact signature size of $\mathcal{O}(n \log q)$,

compared to $\mathcal{O}(n^2)$ size, where n is the security parameter, in the existing state-of-the-art PQC designs. To the best of our knowledge, LaSDVS offers the *smallest private key and signature size* among the existing PQC-based SDVS schemes.

Keywords: Lattice-Based Cryptography, Post-Quantum Cryptography, Strong Designated Verifier Signature, Ring-SIS, Ring-LWE

1 Introduction

Digital Signatures [14] are a crucial cryptographic primitive that provides security properties such as *authentication*, *integrity*, and *non-repudiation* [7]. In particular, it ensures that a message originates from a particular sender, has not been tampered with in between, and prevents any denial caused by signers in digital communication. A digital signature is publicly verifiable using the signer's public key and cannot prevent dishonest verifiers from transferring the validation of sensitive signed messages. However, certain real-life applications exist where the sender of a digital message wants to ensure that only the designated receivers can verify and be convinced whether the signature is valid. As a solution to this problem, the notion of *Designated Verifier Signature* (DVS) was introduced by Jakobsson et al. [6] in Eurocrypt'96. In a DVS scheme, only the designated verifiers can verify the signature and obtain the conviction of correctness of the proof. In this scheme, the signer and the designated verifier both have the equivalent signing privileges, i.e., the designated verifier also has the right to produce a valid signature over the same message. Any third party except the signer and the designated verifier can not distinguish whether the signature was generated by the actual signer or the designated verifier. This property is named as *Non-Transferability* (*NT*). *Strong Designated Verifier Signature* (SDVS), which is a stronger notion of the DVS scheme (also proposed by Jakobsson et al [6] and further explained by Vergnaud et al. [8]) provides some extra useful privacy properties. Specifically, in SDVS, the private key of the designated verifier is also involved while verifying the signature. This means that nobody else (including a third party) can check the validity of the signature, even if they have the public key. SDVS further provides *Non-Delegatability* (*ND*), which is a desired property in cases where the responsibility of the signer becomes very important. For example, an e-voting protocol where it should not be possible to delegate the signing rights to others. Thus, SDVS can be useful in many applications, e.g., e-voting, digital subscription system, online contract agreements, product licensing, etc.

Several SDVS exist in the literature, but the existing state-of-the-art SDVS protocols are based on the number-theoretic assumptions [8]. Quantum algorithms like Shor's algorithm [18] may be used for the cryptanalysis of the existing SDVS designs. To withstand the quantum threats, there is an urgent need to design a quantum secure SDVS protocol. A new direction of research called *Post-Quantum Cryptography* (PQC) has been announced by the *NIST* in 2016 [13]. The goal of PQC is to design and analyze protocols that can securely withstand quantum attacks. PQC can be divided into several categories, such as Lattice-based [5], Multivariate-based

[3], Hash-based [19], Code-based [16], and Isogeny-based [12]. Among these, Lattice-based Cryptography is one of the most important research directions in PQC. In this setting, the security of the proposed cryptographic protocols relies on the worst-case hardness of lattice-based problems [17].

Related Works. A lot of fundamental work has been done in the domain lattice-based cryptography over the last two decades. One of the most groundbreaking works was GPV [4] - the first lattice-based provably secure signature scheme. In [4], the authors introduced a new notion of trapdoor function called *Pre-image Samplable Function* (PSF). Based on the idea of this trapdoor, several lattice-based cryptographic protocols have been defined. The first lattice-based DVS scheme was proposed by Wang et al. [20], which employed PSF and utilized Bonsai Trees for the basis delegation process. Although the scheme was proven secure in the random oracle model, it suffers from large key and signature sizes. In 2017, Noh et al. [15] proposed an SDVS scheme that was proven secure in the standard model. They used the technique of Learning with Errors (LWE)-based public key cryptosystem, and lattice-based chameleon hash function. The SDVS design in [15] also suffered from complex computations and large key and signature sizes. In 2019, Cai et al. [2] proposed a comparatively efficient SDVS scheme based on the hardness of the Ring-SIS problem. They used the idea of rejection sampling to generate a signature of efficient size. *ND* was introduced for SDVS by Lipma et al. [10]. As discussed above, *ND* is a crucial property in real-life applications of SDVS. Earlier constructions of SDVS with *ND* security property were proposed only in the classical setting. In 2024, the first lattice-based SDVS with *ND* property was proposed by Zhang et al. [21], which provides provable security based on the hardness of SIS and LWE problems. A summary of the related works is provided in Table 1.

Our Contribution: The major contribution of this work is summarized below.

1. We propose an efficient post-quantum SDVS, namely LaSDVS based on ideal lattice under the hardness assumptions of Ring-SIS and Ring-LWE
2. LaSDVS provides advanced security properties such as strong unforgeability under the chosen-message attack, non-transferability, non-delegatability, and privacy of signer's identity.
3. We design LaSDVS using the algebraic structures of the ring. Thus, we are able to minimize the computational costs, parameter sizes, and signature and keys overhead by a significant margin. We exploited the idea of the trapdoor, called the Gadget trapdoor, defined by Micciancio et al. in [11]. We emulated this trapdoor definition in the ideal lattice computations that gave us more efficiency than the standard lattices based SDVS designs.
4. LaSDVS is very efficient and compact when compared to the existing state-of-the-art SDVS. In particular, LaSDVS provides the *smallest private key size* among the existing PQC-based SDVS design.
5. Our scheme achieves a *compact signature size* of $\mathcal{O}(n \log q)$, compared to the $\mathcal{O}(n^2)$ signature sizes in standard constructions. This corresponds to a reduction

Table 1: Tecniques, advantages, and limitations of existing lattice-based SDVS schemes

Scheme	Cryptographic Techniques	Advantages	Limitations
Wang et al. [20]	• Pre-image Sampling Functions • Bonsai Trees	• First lattice-based SDVS scheme •Based on the hardness of LWE and SIS problem • Provides *EU* and *NT* security	• Does not provide *PSI* and *ND* security • Complex computations • Large key sizes
Noh et al. [15]	• Used LWE-based PKC • Chameleon Hash function	• First lattice-based SDVS scheme in standard model • Provides *SU*, *NT*, and *PSI* security • Based on the hardness of SIS and LWE problems	• Does not provide *ND* security • Complex computations • Very large key sizes
Cai et al. [2]	• Hard problems in ideal lattice • Filtering techniques	• Not using PSF and Bonsai Trees •Resisting side Channel attacks • Provides *EU, NT*, and *PSI* security • Based on the hardness of $\mathcal{R}-$SIS problem	• Does not offer *ND* security • Large key sizes, but small signature size.
Zhang et al. [21]	• Rejection sampling • Pre-image Samplable Functions	• First lattice-based SDVS with *ND* security •Provable secure based on SIS and LWE problem •Provides *EU, NT*, and *PSI* security	• Large key and signature sizes • Impractical for real-world applications

EU: Existential Unforgeability, *SU*: Strong Unforgeability, *NT*: Non-Transferability, *PSI*: Privacy for Signer's Identity, *ND*: Non-Delegatability. *SIS*: Short Integer Solution,*LWE*: Learning With Errors, *PKC*: Public Key Cryptosystems.

by a factor of $n/\log q$. In fact, the signature size in LaSDVS is *smallest* among all PQC-based SDVS designs.

2 Preliminaries

2.1 Notations

The notations, which have been used in this paper, are described in Table 2.

2.2 Background of Lattice-Based Cryptography

In this section, we discuss the concepts of lattices, ideal lattices, and the important results in the ideal lattices.

Table 2: Notations for the symbols used in the paper

$\mathcal{R}$	Ring of polynomials of degree $n-1$ $\mathbb{Z}[x]/ < x^n + 1 >$ with integer coefficients
$\mathcal{R}_q$	Ring of polynomials of degree $n-1$ $\mathbb{Z}_q[x]/ < x^n + 1 >$ with coefficients in $\mathbb{Z}_q$
$\xleftarrow{\$}$	Sampling uniformly random elements
$\mathbf{a}$	Bold small-case letter denotes a vector
$\mathbf{R}$	Bold capital-case letter denotes a matrix
t	Normal small-case letter denotes an element of the ring
$(\text{mod } q)$	Elements from the set $(-(q-1)/2, \ldots, 0, \ldots, (q-1)/2]$
PPT	Probabilistic Polynomial Time
$\|\|$	Concatenation
$\|\| \cdot \|\|$	Euclidean norm
$\mathbf{a} \cdot \mathbf{b}$	Inner product of two vectors in the ring
$\mathbf{a}t$	Scalar product between a vector of the ring and a ring element
$\emptyset$	Empty set
$\mathbf{a}\|\mathbf{b}$	Concatenation of elements of $\mathbf{a}$ followed by the elements of $\mathbf{b}$

Definition 1 (Lattice [9]) Let $\mathbf{v_1}, \ldots, \mathbf{v_n} \in \mathbb{R}^m$ be a set of linearly independent vectors. The lattice Λ generated by $\mathbf{v_1}, \ldots, \mathbf{v_n}$ is the set of linear combinations of $\mathbf{v_1}, \ldots, \mathbf{v_n}$ with coefficients in integers $\mathbb{Z}$ i.e;

$$\Lambda = \{a_1\mathbf{v_1} + \ldots + a_n\mathbf{v_n} : a_1, a_2, \ldots, a_n \in \mathbb{Z}\}.$$

Three types of integer lattices are mainly considered in the literature. For a given integer modulus q, a matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, and $\mathbf{u} \in \mathbb{Z}_q^n$, define:

$$\Lambda_q(\mathbf{A}^T) = \{\mathbf{x} \in \mathbb{Z}^m : \exists\, \mathbf{s} \in \mathbb{Z}_q^n \text{ s.t. } \mathbf{A}^T \cdot \mathbf{s} = \mathbf{x} \pmod q\}$$

$$\Lambda_q^{\perp}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} \equiv \mathbf{0} \pmod q\}$$

$$\Lambda_q^{\mathbf{u}}(\mathbf{A}) = \{\mathbf{x} \in \mathbb{Z}^m : \mathbf{A} \cdot \mathbf{x} = \mathbf{u} \pmod q\}$$

We now provide the definition of the discrete Gaussian distribution over a lattice, which plays a central role in the design and analysis of lattice-based cryptographic schemes.

Definition 2 (Discrete Gaussian [1]) Let $\Lambda \subset \mathbb{Z}^m, c \in \mathbb{R}^m, \sigma \in \mathbb{R}^+$. Define:

$$\rho_{\sigma,c}(x) = \exp(-\pi \frac{||x-c||^2}{\sigma^2}) \text{ and } \rho_{\sigma,c}(\Lambda) = \sum_{x \in \Lambda} \rho_{\sigma,c}(x).$$

The discrete gaussian distribution over Λ with center c and parameter σ is defined as

$$\forall x \in \Lambda,\ \mathcal{D}_{\Lambda,\sigma,c}(x) = \frac{\rho_{\sigma,c}(x)}{\rho_{\sigma,c}(\Lambda)}.$$

We now define the notion of ideal lattices, which are structured lattices arising from polynomial rings and are fundamental to the efficiency and algebraic properties of lattice-based cryptographic constructions.

Definition 3 (Ideal Lattice [9]) Consider the ring $\mathcal{R} = \mathbb{Z}[x]/(x^n + 1)$ and $\mathcal{R}_q = \mathbb{Z}_q[x]/(x^n + 1)$ for a variable x. Corresponding to the integer lattices defined above, we define three types of lattices over the ring $\mathcal{R}$. For a integral modulus q, $\mathbf{a} \in \mathcal{R}_q^k$, and $u \in \mathcal{R}_q$, define:

$$\Lambda_q(\mathbf{a}) = \{\mathbf{x} \in \mathcal{R}^k : \exists s \in \mathcal{R}_q \text{ s.t. } \mathbf{a} \cdot s = \mathbf{x} \pmod q\}$$

$$\Lambda_q^{\perp}(\mathbf{a}^T) = \{\mathbf{x} \in \mathcal{R}^k : \mathbf{a}^T \cdot \mathbf{x} = 0 \pmod q\}$$

$$\Lambda_q^{u}(\mathbf{a}^T) = \{\mathbf{x} \in \mathcal{R}^k : \mathbf{a}^T \cdot \mathbf{x} = u \pmod q\}$$

Definition 4 (Short Integer Solution (SIS) Problem [17]) Given parameters n, m, and q; where n is the security parameter, $m = \mathcal{O}(n \log q)$ and $q = poly(n)$ is a prime modulus value. For a uniform random matrix $\mathbf{A} \in \mathbb{Z}_q^{n \times m}$, find a non-zero vector $\mathbf{e} \in \Lambda_q^{\perp}(\mathbf{A}) \subset \mathbb{Z}^m$ with "small norm" s.t. $\mathbf{Ae} = 0 \pmod q$

Definition 5 (Decisional Learning With Errors (LWE) Problem [17]) Given parameters n, m, and q: where n is the security parameter, $m = \mathcal{O}(n \log q)$ and $q = poly(n)$ is a prime modulus value. For a given pair $(\mathbf{A}, \mathbf{b} = \mathbf{A}^T\mathbf{s} + \mathbf{e}) \in \mathbb{Z}_q^{n \times m} \times \mathbb{Z}^m$ s.t. $\mathbf{A} \xleftarrow{\$} \mathbb{Z}^{n \times m}$, $\mathbf{s} \xleftarrow{\$} \mathbb{Z}^n$ and $\mathbf{e} \in \mathbb{Z}^m$ is coming from a gaussian distribution defined over $\mathbb{Z}^m$; distinguish this pair with a uniformly random pair $(\mathbf{A}, \mathbf{b}) \in \mathbb{Z}^{n \times m} \times \mathbb{Z}^m$.

Definition 6 (Ring-SIS Problem [1]) Given a ring $\mathcal{R}_q = \mathbb{Z}_q[x]/ < x^n + 1 >$ and uniformly random elements $a_1, \ldots, a_l \in \mathcal{R}_q$, where n is the security parameter, $q = poly(n)$ a prime modulus, and $l = \mathcal{O}(\log q)$. Find l "short elements" $\mathbf{e} = \{e_1, \ldots, e_l\} \in \Lambda_q^{\perp}(\mathbf{a^T}) \subset \mathcal{R}^l$ s.t. $a_1e_1 + \ldots + a_le_l = 0 \pmod q$.

Definition 7 (Decisional Ring-LWE Problem[1]) Given $\mathbf{a} = (a_1, \ldots, a_l)^T \in \mathcal{R}_q^l$, a vector of l uniformly random polynomials, and $\mathbf{b} = \mathbf{a}s + \mathbf{e} \pmod q$, where $s \xleftarrow{\$} \mathcal{R}_q$ and $\mathbf{e} \xleftarrow{\$} \mathcal{R}_q^l$ from a gaussian distribution defined over $\mathcal{R}_q$; distinguish $(\mathbf{a}, \mathbf{b} = \mathbf{a}s + \mathbf{e})$ from $(\mathbf{a}, \mathbf{b})$ drawn uniformly at random from $\mathcal{R}_q^l \times \mathcal{R}_q^l$.

We now introduce the concept of trapdoors for ideal lattices, which enable efficient preimage sampling and form the foundation of many lattice-based cryptographic constructions, including identification and signature schemes.

Definition 8 (g-Trapdoor for Ideal Lattice[1]) Let $\mathbf{a} \in \mathcal{R}_q^{l+k}$ and $\mathbf{g} = (1, 2, \ldots, 2^{k-1}) \in \mathcal{R}_q^k$. A $\mathbf{g}-$trapdoor for $\mathbf{a}$ is a collection of linearly independent vectors of ring elements $\mathbf{R} = (\mathbf{r_1}, \ldots, \mathbf{r_k}) \in \mathcal{R}_q^{l \times k}$ such that $\mathbf{a}^T \begin{bmatrix} \mathbf{R} \\ \mathbf{I}_k \end{bmatrix} = h\mathbf{g}^T$ for some non-zero ring element $h \in \mathcal{R}_q$. The value h is known as the tag of the trapdoor. The effectiveness of the trapdoor is evaluated based on its largest singular value, $s_1(\mathbf{R})$, which is determined as the maximum singular value of the matrix representation of $\mathbf{R}$ in $\mathbb{Z}_q^{ln \times ln}$.

Lemma 1 ($(\mathbf{a}, \mathbf{R}) \leftarrow \mathsf{ringGenTrap}^{D}(\mathbf{a}_0, h)$ [9]) Given as input a vector of ring elements $\mathbf{a}_0 = (a_1, \ldots, a_l)^T \in \mathcal{R}_q^l$, a non-zero invertible ring element $h \in \mathcal{R}_q$, a distribution $\chi^{l \times k}$ over $\mathcal{R}^{l \times k}$. (If no particular $\mathbf{a}_0, h$ are given as input, then the algorithm may choose them itself, e.g., picking $\mathbf{a}_0 \leftarrow \mathcal{R}_q^l$ uniformly, and setting $h = 1$.), the algorithm outputs a vector of ring elements $\mathbf{a} = (\mathbf{a}_0^T, \mathbf{a}_1^T = h\mathbf{g}^T - \mathbf{a}_0^T\mathbf{R})^T \in \mathcal{R}_q^{l+k}$, and a trapdoor $\mathbf{R} = (\mathbf{r}_1, \ldots, \mathbf{r}_k) \in \mathcal{R}^{l \times k}$ with tag $h \in \mathcal{R}_q$. Moreover, the distribution of $\mathbf{a}$ is close to uniform (either statistically or computationally) as long as the distribution of $(\mathbf{a}_0^T, -\mathbf{a}_0^T\mathbf{R})$ is.

Lemma 2 (Regularity Lemma [9]) Let $a_i \xleftarrow{\$} \mathcal{R}_q$ and $r_i \xleftarrow{\$} \chi$ for $i = 1, \ldots, l$, where χ is a distribution over $\mathcal{R}_q$. Then we obtain that the statistical distance between $\mathbf{b} = \sum_{i=1}^{l} a_i r_i$ and the uniform distribution over $\mathcal{R}_q$ is $2^{-\Omega(n)}$. We can further extend this lemma for a vector of elements from $\mathcal{R}_q$.

Lemma 3 ($\mathsf{ringInvert}^{\mathcal{O}}(\mathbf{R}, \mathbf{a}, \mathbf{b})$ [9]) Given as input an oracle $\mathcal{O}$ for inverting the function $\alpha_{\mathbf{g}}(s', \mathbf{e}') = \mathbf{g}s' + \mathbf{e}' \pmod q$ where $\mathbf{e}' \in \mathcal{R}^k$ is suitably small from a gaussian like distribution and $s' \in \mathcal{R}_q$, a vector of ring elements $\mathbf{a} \in \mathcal{R}_q^{l+k}$, a $\mathbf{g}$-trapdoor $\mathbf{R} \in R^{l \times k}$ for $\mathbf{a}$ with tag h, a vector $\mathbf{b} = \mathbf{a}s + \mathbf{e} \pmod q$ for any random $s \in \mathcal{R}_q$ and suitably small $\mathbf{e} \in \mathcal{R}^{l+k}$ coming from a narrow distribution over $\mathcal{R}_q^{l+k}$, the algorithm outputs s and $\mathbf{e}$.

Given a trapdoor $\mathbf{R}$ for $\mathbf{a} \in \mathcal{R}_q^{l+k}$ and $u = \mathbf{a}^{\mathbf{T}} \cdot \mathbf{x} \pmod q$, the sampling algorithm in the following lemma finds the solution $\mathbf{x}$ from desired distribution.

Lemma 4 ($\mathsf{ringSample}^{\mathcal{O}}(\mathbf{R}, \mathbf{a_0}, h, u, \sigma)$ [9]) Given as input in offline mode (i) an oracle $\mathcal{O}(v)$ for gaussian sampling over a desired coset $\Lambda_q^v(\mathbf{g}^T)$ with parameter σ, where $v \in \mathcal{R}_q$, (ii) a vector of ring elements $\mathbf{a_0} \in \mathcal{R}_q^l$, (iii) a trapdoor $\mathbf{R} \in \mathcal{R}^{l \times k}$, (iv) a gaussian parameter σ. In addition, given as input in online phase, (i) a non-zero tag $h \in \mathcal{R}_q$ defining $\mathbf{a} = (\mathbf{a_0}^T, h\mathbf{g}^T - \mathbf{a}_0^T\mathbf{R})^T \in \mathcal{R}_q^{l+k}$, and (ii) a syndrome $u \in \mathcal{R}_q$, the algorithm outputs a vector $\mathbf{x}$ drawn from a distribution statistically close to $\mathcal{D}_{\Lambda_q^v(\mathbf{a_0}^T), \sigma'}$ for some Gaussian parameter σ'.

2.3 Definition of Strong Designated Verifier Signature (SDVS)

A SDVS is a collection of the following algorithms. We follow the syntax and definition of SDVS from [21]. Refer to Figure 1 for an illustrative summary.

$\mathsf{Setup}(1^n)$*:* It is a probabilistic algorithm. On input a security parameter n, it outputs the public parameters pp.

$\mathsf{SigKeyGen}(pp)$*:* It is a probabilistic (or deterministic) algorithm. On input pp, it outputs the public key pk_S and private key sk_S for a signer S.

$\mathsf{VerKeyGen}(pp)$*:* It is a probabilistic (or deterministic) algorithm. On input pp, it outputs the public key pk_V and private key sk_V for a designated verifier V.

$\mathsf{Sign}(pp, sk_S, pk_S, pk_V, \mu)$: It is a probabilistic algorithm. On input pp, a private key sk_S, the public keys pk_S and pk_V of the signer S and a designated verifier V and a message $\mu \in \mathcal{M}$, it outputs a real designated verifier signature $\mathsf{sig} \in \mathcal{S}$.

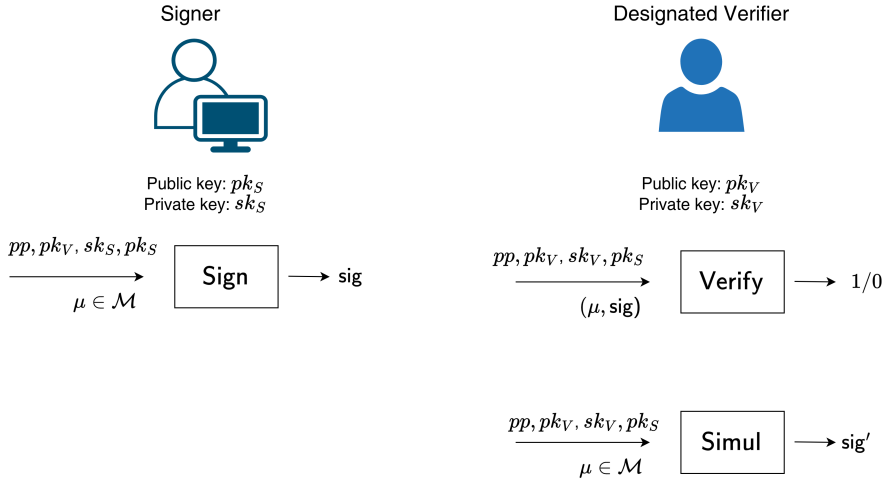

Fig. 1: High level overview of SDVS

Verify$(pp, sk_V, pk_S, pk_V, \text{sig}, \mu)$*:* It is a deterministic algorithm. On input pp, a private key sk_V, the public keys pk_S and pk_V of the signer S and a designated verifier V, a message $\mu \in \mathcal{M}$, and a signature $\text{sig} \in \mathcal{S}$, it outputs a boolean decision b: $b = 1$ denotes accepting or $b = 0$ denotes rejecting it.

Simul$(pp, sk_V, pk_S, pk_V, \mu)$*:* It is a probabilistic algorithm. On input pp, a private key sk_V, the public keys pk_S and pk_V of the signer S and a designated verifier V, and a message $\mu \in \mathcal{M}$, it outputs a simulated designated verifier signature $\text{sig}' \in \mathcal{S}$.

2.4 Security Model

An SDVS scheme must satisfy unforgeability over chosen message attack, non-transferability, privacy of the signer's identity, and non-delegatability. The security definitions are given below.

Strong Unforgeable Chosen Message Attack (*SU-CMA*) Security

For our SDVS scheme, we define strong unforgeability, which is a stronger notion than existential unforgeability. It states that any PPT adversary, without the private key of either the signer or the designated verifier, can not provide a valid designated verifier signature for a message that has even been queried in either of the signing or simulating queries. We define this unforgeable security, where the attacker can choose the message in an adaptive manner, by the following game between a challenger $\mathcal{C}$ and a PPT adversary $\mathcal{A}$:

Setup*:* $\mathcal{C}$ runs Setup(1^n) to get pp, SignKeyGen(pp), and VerKeyGen(pp) to get (pk_S, sk_S) and (pk_V, sk_V) for the signer S and the designated verifier V. $\mathcal{C}$ keeps (sk_S, sk_V) in secret and sends (pp, pk_S, pk_V) to $\mathcal{A}$.

Signing Queries: $\mathcal{A}$ chooses a message μ_i adaptively, $\mathcal{C}$ runs $\mathsf{Sign}(pp, sk_S, pk_S, pk_V, \mu_i)$ to obtain a signature sig_i and sends it to $\mathcal{A}$.

Simulating Queries: $\mathcal{A}$ chooses a message μ_i adaptively, $\mathcal{C}$ runs $\mathsf{Simul}(pp, sk_V, pk_S, pk_V, \mu_i)$ to obtain a signature sig'_i and sends it to $\mathcal{A}$. $\mathcal{A}$ is allowed to query the signing and the simulating oracle at most $q_s = poly(n)$ times.

Verification Queries: $\mathcal{A}$ chooses a message-signature pair (μ_i, sig_i) adaptively, $\mathcal{C}$ runs $\mathsf{Verify}(pp, sk_V, pk_S, pk_V, \mathsf{sig}_i, \mu_i)$ to obtain a decisional value 1 for valid and 0 otherwise, and sends it to $\mathcal{A}$. $\mathcal{A}$ is allowed to query the verification oracle at most $q_v = poly(n)$ times.

Output: $\mathcal{A}$ outputs a message-signature pair (μ^*, sig^*), and wins if the following conditions hold-

- $\mathsf{Verify}(pp, sk_V, pk_S, pk_V, \mathsf{sig}^*, \mu^*) = 1$.
- $(\mathsf{sig}^*, \mu^*) \neq (\mathsf{sig}_i, \mu_i)$, $\forall i \in \{1, 2, \ldots, q_s\}$.

The advantage of $\mathcal{A}$ in the above game, i.e., the probability of $\mathcal{A}$ wins, is denoted by $Adv^{SU-CMA}_{SDVS,\mathcal{A}}(n)$, and the probability is taken over the randomness of $\mathcal{A}$, the algorithms $\mathsf{Sign}, \mathsf{Simul}$, and Verify. We say that the SDVS scheme is unforgeable if $Adv^{SU-CMA}_{SDVS,\mathcal{A}}(n)$ is negligible in the security parameter n.

Non-Transferability (NT) Security

In the context of the SDVS scheme, non-transferability means that the designated verifier should not be able to transfer the conviction of the validity of a real designated verifier signature. This is accomplished by the $\mathsf{Sign}, \mathsf{Simul}$ algorithm, using which the designated verifier can generate signatures that are indistinguishable from those generated by the actual signer. The NT security of our SDVS is defined as follows:

Setup: The challenger $\mathcal{C}$ runs $\mathsf{Setup}(1^n)$ to get pp, then runs $\mathsf{SignKeyGen}(pp)$, and $\mathsf{VerKeyGen}(pp)$ to get (pk_S, sk_S) and (pk_V, sk_V) for the signer S and the designated verifier V. $\mathcal{C}$ sends (pk_S, pk_V) to $\mathcal{A}$.

Challenge: $\mathcal{A}$ chooses a message μ^* adaptively, $\mathcal{C}$ runs $\mathsf{Sign}(pp, sk_S, pk_S, pk_V, \mu^*)$ to obtain a signature sig^*_0, then runs $\mathsf{Simul}(pp, sk_V, pk_S, pk_V, \mu^*)$ to obtain a signature sig^*_1. $\mathcal{C}$ chooses a random bit $b \in \{0, 1\}$ and sends sig^*_b to $\mathcal{A}$.

Output: $\mathcal{A}$ outputs a bit b^*, and wins the challenge if $b^* = b$.

The advantage of $\mathcal{A}$ in the above game is defined by $Adv^{NT}_{SDVS,\mathcal{A}}(n) = |Pr[b^* = b] - 1/2|$, and the probability is taken over the randomness of $\mathcal{A}$, the algorithms Sign and Simul. We say that the SDVS scheme is non-transferable if $Adv^{NT}_{SDVS,\mathcal{A}}(n)$ is negligible in the security parameter n.

Privacy of Signer's Identity (PSI) Security

PSI security provides privacy to the identity of the signer. Given a designated verifier signature and two valid signing public keys, any adversary/eavesdropper in between the signer and the designated verifier can not determine which private key, corresponding to the given public keys, has been used to create the signature with a non-negligible

probability. This feature is, precisely, made possible by involving the private key of the designated verifier in the verification process. The *PSI* security of the SDVS scheme is defined by the following game between a PPT adversary $\mathcal{A}$ and a challenger $\mathcal{C}$:

Setup*:* $\mathcal{C}$ runs $\mathsf{Setup}(1^n)$ to get pp, runs $\mathsf{SignKeyGen}(pp)$ twice to get (pk_{S_0}, sk_{S_0}), (pk_{S_1}, sk_{S_1}), and runs $\mathsf{VerKeyGen}(pp)$ to get (pk_V, sk_V) for the signers S_0, S_1 respectively and the designated verifier V. $\mathcal{C}$ keeps sk_V in secret and sends $(pp, pk_{S_0}, sk_{S_0}, pk_{S_1}, sk_{S_1}, pk_V)$ to $\mathcal{A}$.

Simulating Queries*:* $\mathcal{A}$ chooses a message μ_i adaptively and a bit $b \in \{0,1\}$, $\mathcal{C}$ runs $\mathsf{Simul}(pp, sk_V, pk_{S_b}, pk_V, \mu_i)$ to obtain a signature $\mathsf{sig}'_{i,b}$ and sends it to $\mathcal{A}$. $\mathcal{A}$ is allowed to query the simulating oracle at most $q_s = poly(n)$ times.

Verification Queries: $\mathcal{A}$ chooses a message-signature pair (μ_i, sig_i) and a bit $b \in \{0,1\}$ adaptively, $\mathcal{C}$ runs $\mathsf{Verify}(pp, sk_V, pk_{S_b}, pk_V, \mathsf{sig}_i, \mu_i)$ to obtain a decisional value 1 for valid and 0 otherwise, and sends it to $\mathcal{A}$. $\mathcal{A}$ is allowed to query the verification oracle at most $q_v = poly(n)$ times.

Challenge*:* $\mathcal{A}$ chooses a message μ^*, $\mathcal{C}$ runs $\mathsf{Sign}(pp, sk_{S_0}, pk_{S_0}, pk_V, \mu^*)$ to obtain a signature sig_0^*, then runs $\mathsf{Sign}(pp, sk_{S_1}, pk_{S_1}, pk_V, \mu^*)$ to obtain a signature sig_1^*, chooses a random bit $b \in \{0,1\}$, and sends sig_b^* to $\mathcal{A}$.

Output*:* $\mathcal{A}$ outputs a bit $b^* \in \{0,1\}$, and wins if the following conditions hold-

- $b^* = b$,.
- $(\mathsf{sig_b}^*, \mu^*)$ was not queried in the verification queries for $b^* \in \{0,1\}$.

The advantage of $\mathcal{A}$ is defined as $Adv^{PSI}_{SDVS,\mathcal{A}}(n) = |Pr[b^* = b] - 1/2$, and we say that the SDVS scheme holds *PSI* security if $Adv^{PSI}_{SDVS,\mathcal{A}}(n)$ is negligible in n.

Non-Delegatability (*ND*) Security

In SDVS scheme, *ND* security tells that without knowing the private key of either the signer or the designated verifier, no one can generates a valid designated verifier signature. In other words, a *ND* secure SDVS accomplish the proof of knowledge of either signer's private key or designated verifier's private key in a non-interactive manner. This security is defined in the form of a game between an extractor $\mathcal{E}$ and a black-box $\mathcal{A}$ that produces a valid signature:

Setup*:* The extractor $\mathcal{E}$ runs $\mathsf{Setup}(1^n)$ to get pp and sends it to $\mathcal{A}$. $\mathcal{A}$ then sends either pk_S or pk_V to $\mathcal{E}$.

– If $\mathcal{A}$ sends pk_S to $\mathcal{E}$, then the game proceeds as follows-

VerKeyGen*:* $\mathcal{E}$ runs $\mathsf{VerKeyGen}(pp)$ to obtain (pk_V, sk_V), keeps sk_V in secret and sends pk_V to $\mathcal{A}$.

Simulating Queries*:* $\mathcal{A}$ chooses a message μ_i adaptively , $\mathcal{E}$ runs $\mathsf{Simul}(pp, sk_V, pk_S, pk_V, \mu_i)$ to obtain a signature sig_i and sends it to $\mathcal{A}$. $\mathcal{A}$ is allowed to query the simulating oracle at most $q_s = poly(n)$ times.

Verification Queries: $\mathcal{A}$ chooses a message-signature pair (μ_i, sig_i) adaptively, $\mathcal{E}$ runs $\mathsf{Verify}(pp, sk_V, pk_S, pk_V, \mathsf{sig}_i, \mu_i)$ to obtain a decisional value, 1 for valid and 0 otherwise, and sends it to $\mathcal{A}$. $\mathcal{A}$ is allowed to query the verification oracle at most $q_v = poly(n)$ times.

Challenge: $\mathcal{E}$ chooses a message μ^* and $\mathcal{A}$ sends a corresponding designated verifier signature sig^* to $\mathcal{E}$.

Output: $\mathcal{E}$ outputs the private key sk_S of the signer.

– If $\mathcal{A}$ sends pk_V to $\mathcal{E}$, then the game proceeds as follows-

SigKeyGen: $\mathcal{E}$ runs $\mathsf{SigKeyGen}(pp)$ to obtain (pk_S, sk_S), keeps sk_S in secret and sends pk_S to $\mathcal{A}$.

Signing Queries: $\mathcal{A}$ chooses a message μ_i adaptively, $\mathcal{E}$ runs $\mathsf{Sign}(pp, sk_S, pk_S, pk_V, \mu_i)$ to obtain a signature sig_i and sends it to $\mathcal{A}$. $\mathcal{A}$ is allowed to query the signing oracle at most $q_s = poly(n)$ times.

Challenge: $\mathcal{E}$ chooses a message μ^* and $\mathcal{A}$ sends a corresponding designated verifier signature sig^* to $\mathcal{E}$.

Output: $\mathcal{E}$ outputs the private key sk_V of the designated verifier.

The advantage of $\mathcal{E}$ in the above game, in time t is denoted by $Adv^{ND}_{SDVS,\mathcal{E}}(n)$, and the probability is taken over the randomness of $\mathcal{A}$, the algorithms $\mathsf{Sign}, \mathsf{Simul}$, and Verify. We say that the SDVS scheme is non-delegatable against $\mathcal{A}$, if $Adv^{ND}_{SDVS,\mathcal{E}}(n) \geq poly(\epsilon')$ and $t < poly(t')$ where $\mathcal{A}$ can produce a designated verifier signature in time t' with a probability ϵ'.

3 Proposed Ideal Lattice-based SDVS Scheme

High Level Overview. In this section, we describe the proposed construction (called LaSDVS) in detail. We first give a high-level overview of the design. LaSDVS is a privacy-preserving digital signature protocol built upon the hardness of mathematical problems over *ideal lattices*. Unlike traditional signature schemes, LaSDVS ensures that only a designated verifier can check the validity of a signature, while also allowing the verifier to simulate signatures that are computationally indistinguishable from real ones. LaSDVS consists of six algorithms: (Setup $\mathsf{SigKeyGen}$, $\mathsf{VerKeyGen}$, Sign, Verify, Simul). In the Setup phase, public parameters are initialized based on underlying ring structures and hash function instantiation. The algorithms $\mathsf{SigKeyGen}$ and $\mathsf{VerKeyGen}$ generate key pairs for the signer and the verifier, respectively. The signing algorithm Sign enables the signer to produce a short signature for a message, while the Verify algorithm utilizes the trapdoor to validate the signature's correctness. The Simul algorithm allows the verifier to simulate signatures without interacting with the signer, thus providing non-transferability. We provide a detailed description of the algorithms below.

Setup: Let n be a security parameter, and let $q = poly(n)$ be the underlying ring modulus. Let k and κ denote the hash parameters such that $k = \lceil \log q \rceil$ and $2^\kappa \binom{k}{\kappa} \geq 2^{100}$. Let $d = q^{1/\gamma}$ with $\gamma > 1$, $\sigma = \mathcal{O}(\sqrt{\log n})$, and $\chi = \mathcal{D}_{\mathcal{R},\sigma}$. $\mathcal{R} = \mathbb{Z}[x]/ < x^n + 1 >$. $\mathcal{R}_q = \mathbb{Z}_q[x]/ < x^n + 1 >$. l is chosen suct that $l + k = \mathcal{O}(\log q)$. Let η be a constant positive real value. Then, the Setup phase is executed as follows.

- Sample $\mathbf{a} \xleftarrow{\$} \mathcal{R}_q^{l+k}$.

- Select a collision resistant hash function $\mathcal{H} : \{0,1\}^* \rightarrow \{c, c \in \mathcal{R}_q, 0 < ||c|| < \sqrt{\kappa}\}$.

The public parameter pp is set to be $\{n, q, k, \kappa, d, \sigma, \chi, \mathcal{R}, \mathcal{R}_q, l, \mathbf{a}, \mathcal{H}\}$

$(pk_S, sk_S) \leftarrow$ SignKeyGen(pp): On input the public parameters pp, the following steps are performed to generate the public key and secret key for the signer.

- Sample $\mathbf{s} \xleftarrow{\$} \{-d, \ldots, 0, \ldots, d\}^{l+k}$
- Define $t = \mathbf{a} \cdot \mathbf{s} \pmod{q} \in \mathcal{R}_q$
- Output $pk_S = t$ and $sk_S = \mathbf{s}$

$(pk_V, sk_V) \leftarrow$ VerKeyGen(pp): Similar to the previous algorithm, on input the public parameters pp, the following operations are executed to produce the public key and secret key for the designated verifier.

- Sample $\tilde{\mathbf{b}}_\mathbf{0}, \tilde{\mathbf{b}}_\mathbf{1} \xleftarrow{\$} \mathcal{R}_q^l$
- Execute ringGenTrap with inputs $\tilde{\mathbf{b}}_\mathbf{0}$ and $h_0 = 1$ to output $\mathbf{b_0} \in \mathcal{R}_q^{l+k}$ and a trapdoor $\mathbf{R}_{\mathbf{b_0}} \in \mathcal{R}_q^{l \times k}$.
- Execute ringGenTrap with inputs $\tilde{\mathbf{b}}_\mathbf{1}$ and $h_1 = 1$ to output $\mathbf{b_1} \in \mathcal{R}_q^{l+k}$ and a trapdoor $\mathbf{R}_{\mathbf{b_1}} \in \mathcal{R}_q^{l \times k}$.
- Output $pk_V = (\mathbf{b_0}, \mathbf{b_1})$ and $sk_V = (\mathbf{R}_{\mathbf{b_0}}, \mathbf{R}_{\mathbf{b_1}})$

sig $\leftarrow$ Sign$(pp, sk_S, pk_S, pk_V, \mu \in \mathcal{M})$: On input pp, sk_S, pk_S, and pk_V, the signer employs the algorithm Sign to generate a signature on the message μ as follows.

- Samples $s \xleftarrow{\$} \mathcal{R}_q$, $\mathbf{e}, \mathbf{y} \xleftarrow{\$} \mathcal{D}_{\mathcal{R}_q^{l+k}, \sigma}$
- Computes $\mathbf{c_0} = \mathbf{b_0} \cdot s + \mathbf{e} \in \mathcal{R}_q^{l+k}$
- Computes $c_1 = \mathcal{H}(\mathbf{a} \cdot \mathbf{y} + \mathbf{b_1} \cdot \mathbf{e} || t || s || \mu) \in \mathcal{R}_q$
- $\mathbf{z} = \mathbf{s} \cdot c_1 + \mathbf{y} \pmod{q} \in \mathcal{R}_q^{l+k}$
- Output sig $= (\mathbf{c_0}, c_1, \mathbf{z})$

$1/0 \leftarrow$ Verify$(pp, sk_V, pk_V, pk_S,$ sig$, \mu)$: On input pp, sk_V, pk_S and pk_V, the Verify outputs 1 if sig is a valid signature on the message μ; otherwise outputs 0.

- Parse sig
- Check $0 < ||\mathbf{z}|| < \eta\sigma\sqrt{l+k}$
- Execute ringInvert with input $\mathbf{R}_{\mathbf{b_0}}$, $\mathbf{c_0}, \mathbf{b_0}$ to output $s, \mathbf{e}$.
- Check whether $c_1 = \mathcal{H}(\mathbf{a} \cdot \mathbf{z} - \mathbf{t}c_1 + \mathbf{b_1} \cdot \mathbf{e} || \mathbf{t} || s || \mu)$
- Output 1 if all the above checks are satisfied, otherwise outputs 0.

sig' $\leftarrow$ Simul$(pp, sk_V, pk_V, pk_S, \mu)$: On input pp, sk_V, pk_S and pk_V, a designated verifier utilizes Simul to output a simulated signature sig' on the message μ

- Sample $s', u' \xleftarrow{\$} \mathcal{R}_q, \mathbf{z}' \xleftarrow{\$} \mathcal{D}_{\mathcal{R}_q^{l+k}, \sigma}$
- Let $c_1' = \mathcal{H}(u' || t || s' || \mu) \in \mathcal{R}_q$
- Run RingSample$(\mathbf{b_1}, h_1, \mathbf{R}_{\mathbf{b_1}}, u' - \mathbf{a} \cdot \mathbf{z} + \mathbf{t} \cdot c_1', \sigma)$ to get a short $\mathbf{e}' \in \mathcal{D}_{\mathcal{R}_q^{l+k}, \sigma}$.
- Define $\mathbf{c_0'} = \mathbf{b_0} \cdot s' + \mathbf{e}' \bmod q$.
- Output sig$' = (\mathbf{c_0'}, c_1', \mathbf{z}')$ using rejection sampling technique.

Correctness

In the verification phase, the verifier checks the validity of the given message-signature pair $(\mu,$ sig$)$ by taking (pp, sk_V, pk_V, pk_S) as input. The correctness analysis is given below. If the signature is coming from the actual signer i.e., sig $= (\mathbf{c_0}, c_1, \mathbf{z})$, then the

verifier first checks if $0 < ||\mathbf{z}|| < \eta\sigma\sqrt{l+k}$. Then using the subroutine $\mathsf{ringInvert}$, it obtains $s, \mathbf{e} \leftarrow \mathsf{ringInvert}(\mathbf{R_{b_0}}, \mathbf{b_0}, \mathbf{c_0})$. Finally, the verifier computes

$$\begin{aligned} c_1^* &= \mathcal{H}(\mathbf{a}\cdot\mathbf{z} - \mathbf{t}c_1 + \mathbf{b_1}\cdot\mathbf{e}||\mathbf{t}||s||\mu) \\ &= \mathcal{H}(\mathbf{a}\cdot(\mathbf{s}c_1 + \mathbf{y}) - \mathbf{a}\cdot\mathbf{s}c_1 + \mathbf{b_1}\cdot\mathbf{e}||\mathbf{t}||s||\mu) \\ &= \mathcal{H}(\mathbf{a}\cdot\mathbf{s}c_1 + \mathbf{a}\cdot\mathbf{y} - \mathbf{a}\cdot\mathbf{s}c_1 + \mathbf{b_1}\cdot\mathbf{e}||\mathbf{t}||s||\mu) \\ &= \mathcal{H}(\mathbf{a}\cdot\mathbf{y} + \mathbf{b_1}\cdot\mathbf{e}||\mathbf{t}||s||\mu) \\ &= c_1 \end{aligned}$$

If the signature is generated from the Simul algorithm by the designated verifier i.e., $\mathsf{sig}' = (\mathbf{c_0'}, c_1', \mathbf{z}')$ then the verifier first checks whether $0 < ||\mathbf{z}'|| < \eta\sigma\sqrt{l+k}$ and then computes $s', \mathbf{e}' \leftarrow \mathsf{ringInvert}(\mathbf{R_{b_0}}, \mathbf{b_0}, \mathbf{c_0'})$. Now, from Lemma 4, we get that $\mathbf{b_1}\cdot\mathbf{e}' = u' - \mathbf{a}\cdot\mathbf{z} + \mathbf{t}c_1' \bmod q$. In the end, the verifier computes,

$$\begin{aligned} \tilde{c'}_1 &= \mathcal{H}(\mathbf{a}\cdot\mathbf{z}' - \mathbf{t}c_1' + \mathbf{b_1}\cdot\mathbf{e}'||\mathbf{t}||s'||\mu) \\ &= \mathcal{H}(\mathbf{a}\cdot\mathbf{z}' - \mathbf{t}c_1' + u' - \mathbf{a}\cdot\mathbf{z}' + \mathbf{t}c_1'||\mathbf{t}||s'||\mu) \\ &= \mathcal{H}(u'||\mathbf{t}||s'||\mu) \\ &= c_1' \end{aligned}$$

4 Efficiency Analysis

In this section, we present a comprehensive comparative analysis of LaSDVS with the existing state-of-the-art PQC-based SDVS. We first present the communication and storage overhead of LaSDVS .

- The private key sk_S of the signer is of bit size $(l+k)\log(2d+1) = \mathcal{O}(\log q)$.
- The private key sk_V of the designated verifier is of bit size $2(l \times k)(n\log q) = 2n(\log q)^2 = \mathcal{O}(n\log q)$.
- The signature $\mathsf{sig} = (\mathbf{c_0}, \mathbf{c_1}, \mathbf{z})$ is of bit size $(l+k)n\log q + n\log q + (l+k)n\log q = 2(l+k)n\log q + n\log q = \mathcal{O}(n\log q)$.

We now compare LaSDVS with the existing lattice-based SDVS schemes proposed in [20, 15, 2, 21]. The comparative analysis considers sizes of the secret key of the signer and verifier, signature sizes. In addition, we also provided an assessment of security properties such as *SU-CMA, NT, PSI*, and *ND*. The results of the comparison is provided in Table 3. As we can see from the results mentioned in Table 3, LaSDVS outperforms existing schemes in the communication and storage overhead, while attaining all the desired security properties. Notably, LaSDVS achieves *SU-CMA, NT, PSI* and *ND* security properties, similar to [21], but with significantly lower keys and signature sizes. Specifically, LaSDVS attains a key and signature size of $\mathcal{O}(\log q)$ and $\mathcal{O}(n\log q)$, respectively, in contrast to earlier works whose sizes are polynomial in the security parameter n. This efficiency is primarily due to two design choices; (i) the use of a gadget-based trapdoor [11] in the ideal lattice setting, and (ii) the avoidance of the traditional GPV trapdoor [4] used in standard lattice constructions.

Wang et al. [20] incorporated the Bonsai Tree technique to support trapdoor delegation and achieve *NT* and *PSI*, but their construction remains inefficient due to the use of GPV trapdoors. It results in large $\mathcal{O}(n^2)$-sized keys and signatures. In Noh et al. [15], although the authors utilized a different trapdoor mechanism (from [11]) along with a lattice-based chameleon hash function, the use of high-dimensional matrices resulted in a signature and key size of $\mathcal{O}(n^2)$. Cai et al. [2] proposed an optimization using a filtering technique to reduce the signature size, but the improvement was marginal, and the scheme still results in a quadratic signature size. Similarly, Zhang et al. [21] introduced the first lattice-based SDVS supporting *ND*, but their use of GPV trapdoors in standard lattices again incurs $\mathcal{O}(n^2)$ complexity in all parameters. Thus, LaSDVS is the most efficient SDVS till date in the PQC. LaSDVS provides the smallest private key size and signature size among the existing PQC-based SDVS design. LaSDVS is compact and well suited for resource constrained devices.

Table 3: Comparison of lattice-based SDVS schemes

Schemes	$\lvert\mathbf{sk_S}\rvert$	$\lvert\mathbf{sk_V}\rvert$	$\lvert\mathsf{sig}\rvert$	*CMA*	*NT*	*PSI*	*ND*	**Model**
Wang et al. [20]	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	EU	Yes	Yes	No	ROM
Noh et al. [15]	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	EU	Yes	Yes	No	SM
Cai et al. [2]	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	EU	Yes	Yes	No	ROM
Zhang et al. [21]	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	$\mathcal{O}(n^2)$	SU	Yes	Yes	Yes	ROM
LaSDVS	$\mathcal{O}(\log q)$	$\mathcal{O}(n \log q)$	$\mathcal{O}(n \log q)$	SU	Yes	Yes	Yes	ROM

5 Security Analysis of LaSDVS

Theorem 1 LaSDVS *is strongly unforgeable under the adaptive chosen-message attack (SU-CMA).*

Proof We show that LaSDVS is *SU-CMA* secure. Refer to Section 2.4 for the detailed definition of *SU-CMA* game. We assume that there exists a PPT adversary $\mathcal{A}$ who makes queries to the signer and the random oracle $\mathcal{H}$ and then performs an adaptive chosen-message attack on our proposed SDVS scheme. The adversary, at last, is able to output a forged designated verifiable signature sig^* on a message μ^* with a non-negligible probability ϵ, then we show that there exists a simulator $\mathcal{B}$ that can exploit $\mathcal{A}'s$ success probability to solve an instance of the ring-SIS problem $(\mathbf{a}|\mathbf{b}) \cdot \mathbf{v} = 0 \pmod q$. We define a hybrid SDVS, to provide a valid simulation of the random oracle to the adversary $\mathcal{A}$ in the proof, in which no private keys are used and the output of which can not be distinguished from a real SDVS. In both, signing and simulating algorithms, a hybrid SDVS is generated as follows

1. Sample $s \xleftarrow{\$} \mathcal{R}_q$ and $\mathbf{e} \xleftarrow{\$} \mathcal{D}_{\mathcal{R}_q^{l+k},\sigma}$.
2. Define $\mathbf{c_0} = \mathbf{b}s + \mathbf{e} \pmod q$.
3. Sample $c_1 \xleftarrow{\$} \mathcal{R}_q$ and $\mathbf{z} \xleftarrow{\$} \mathcal{D}_{\mathcal{R}_q^{l+k},\eta\sigma}$.
4. Program the oracle $\mathcal{H}(\mathbf{a} \cdot \mathbf{z} - tc_1 + \mathbf{b_1} \cdot \mathbf{e}||t||s||\mu) = c_1$.

5. Output $\mathsf{sig} = (\mathbf{c_0}, c_1, \mathbf{z})$.

Throughout the proof, let $\mathcal{D}_{\mathcal{H}} = \{c_1 \in \mathcal{R}_q : 0 < ||c_1|| < \sqrt{\kappa}\}$ denotes the range of the random oracle $\mathcal{H}$ and w be bound on the number of times the oracle is called or programmed during $\mathcal{A}'s$ attack i.e. the random oracle query can be made by the adversary $\mathcal{A}$ directly or the random oracle can be programmed by the signing and simulating algorithms when $\mathcal{A}$ asks for a signature on some adaptively chosen messages. The interaction between $\mathcal{A}$ and $\mathcal{B}$ are as follows:

- Setup: Given $\mathbf{a}, \mathbf{b} \in \mathcal{R}_q^{l+k}$, $\mathcal{B}$ computes the following steps
  1. Sample $\mathbf{s} \xleftarrow{\$} \{-d, \ldots, 0, \ldots, d\}^{l+k}$ and define $\mathbf{t} = \mathbf{a} \cdot \mathbf{s} \pmod q$.
  2. Execute ringGenTrap with inputs $\tilde{\mathbf{b_0}}$ and $h_0 = 1$ to output $\mathbf{b_0} \in \mathcal{R}_q^{l+k}$ with a trapdoor $\mathbf{R_{b_0}} \in \mathcal{R}_q^{l \times k}$.
  3. Sample random coins ϕ for $\mathcal{A}$ and Φ for $\mathcal{B}$.
  4. Sample uniformly random $r_1, r_2, \ldots, r_w \xleftarrow{\$} \mathcal{D}_{\mathcal{H}}$, which will correspond to the responses of the random oracle.
  5. Define $(pk_S, sk_S) = (t, \mathbf{s})$ and $(pk_V, sk_V) = (\mathbf{b_0}, \mathbf{b_1} = \mathbf{b}, \mathbf{R_{b_0}}, \emptyset)$.
  6. Keep $(\mathbf{s}, \mathbf{R_{b_0}}, r_1, r_2, \ldots, r_w, \Phi)$ in secret, and send the public parameters $pp = (\mathbf{a}, \mathbf{b_0}, \mathbf{b_1}, t, \phi)$ to $\mathcal{A}$.
- Hash Queries: Given $(\mathbf{a} \cdot \mathbf{y} + \mathbf{b_1} \cdot \mathbf{e}||t||s||\mu)$, the simulator $\mathcal{B}$ executes the following steps-
  1. Check whether a corresponding hash output c is stored in the hash list $l_{\mathcal{H}}$. If yes, return it directly.
  2. Else, sample a random $r_i \xleftarrow{\$} \{r_1, r_2, \ldots, r_w\}$ that has not been used yet. Store $(\mathbf{a} \cdot \mathbf{y} + \mathbf{b_1} \cdot \mathbf{e}||t||s||\mu||c = r_i)$ in $l_{\mathcal{H}}$ and return it to $\mathcal{A}$.
- Signing Queries: Given a message μ, $\mathcal{B}$ executes the following steps
  1. Run the signing algorithm in Hybrid SDVS using the random coins Φ to produce a signature $\mathsf{sig} = (\mathbf{c_0}, c_1, \mathbf{z})$.
  2. Store the hash inputs $(\mathbf{a} \cdot \mathbf{z} - tc_1 + \mathbf{b_1} \cdot \mathbf{e}||t||s||\mu)$ and output c_1 into $l_{\mathcal{H}}$.
  3. Return sig to $\mathcal{B}$.
- Simulating Queries: Given a message μ, $\mathcal{B}$ does as in the Signing Queries and returns a signature $\mathsf{sig} = (\mathbf{c_0}, c_1, \mathbf{z})$ to $\mathcal{B}$.
- Verification Queries: Given a signature $\mathsf{sig} = (\mathbf{c_0}, c_1, \mathbf{z})$ and a message μ, $\mathcal{B}$ executes the following steps-
  1. Check whether $0 < ||\mathbf{z}|| < \eta\sigma\sqrt{l+k}$ where $1 < \eta < 2$ is a constant.
  2. Run ringInvert with the inputs $\mathbf{c_0}, \mathbf{b_0}$, and $\mathbf{R_{b_0}}$ to output $s, \mathbf{e}$.
  3. Check whether $(\mathbf{a} \cdot \mathbf{z} - tc_1 + \mathbf{b_1} \cdot \mathbf{e}||t||s||\mu||c_1)$ is stored in the hash list $l_{\mathcal{H}}$.
  4. Output 1 if all the above are satisfied, otherwise 0.
- Output: With a probability ϵ, $\mathcal{A}$ outputs a message-signature pair $(\mu^*, \mathsf{sig}^* = (\mathbf{c_0}^*, c_1^*, \mathbf{z}^*))$ which satisfies $0 < ||\mathbf{z}^*|| < \eta\sigma\sqrt{l+k}$ and $c_1^* = \mathcal{H}(\mathbf{a} \cdot \mathbf{z}^* - tc_1^* + \mathbf{b_1} \cdot \mathbf{e}^*||t||s^*||\mu^*)$ where $\mathbf{e}^* \in \mathcal{R}_q^{l+k}$ and $s^* \in \mathcal{R}_q$ had been used in $\mathbf{b_0}^*$.

With a probability $1 - 1/|\mathcal{D}_{\mathcal{H}}|$, c_1^* must be one of the $r_i \in \{r_1, r_2, \ldots, r_w\}$. The success probability of $\mathcal{A}$ in the forgery with the condition that c_1^* must be one of the $r_i's$, is at least $\epsilon - 1/|\mathcal{D}_{\mathcal{H}}|$. If $c_1^* = r_i \in \{r_1, r_2, \ldots, r_w\}$, then there will be two cases:

Case I If $c_1^* = r_i$ was programmed during the Signing Queries or the Simulating Queries, then we analyse this as follows. Assume that the simulator $\mathcal{B}$ programmed the random oracle $c_1^* = \mathcal{H}(\mathbf{a} \cdot \mathbf{z} - tc_1^* + \mathbf{b_1} \cdot \mathbf{e}||t||s||\mu)$ when signing a message μ. Since the adversary outputs a valid forgery $(\mathsf{sig}^* = (\mathbf{c_0}^*, c_1^*, \mathbf{z}^*))$ for μ^*, we have $\mathcal{H}(\mathbf{a} \cdot \mathbf{z} - tc_1^* + \mathbf{b_1} \cdot \mathbf{e}||t||s||\mu) =$

$\mathcal{H}(\mathbf{a}\cdot\mathbf{z}^* - tc_1^* + \mathbf{b_1}\cdot\mathbf{e}^*||t||s^*||\mu^*)$. Thus according to the collision-resistance property of $\mathcal{H}$, $\mathcal{B}$ computes $\mathbf{e}, \mathbf{e}^*$ from the values $\mathbf{c_0}, \mathbf{c_0}^*$ using the subroutine ringInvert respectively.So, $\mathcal{B}$ computes

$$\mathbf{a}\cdot\mathbf{z} - tc_1^* + \mathbf{b_1}\cdot\mathbf{e} = \mathbf{a}\cdot\mathbf{z} - tc_1^* + \mathbf{b}\cdot\mathbf{e} = \mathbf{a}\cdot\mathbf{z}^* - tc_1^* + \mathbf{b_1}\cdot\mathbf{e}^*$$
$$\implies \mathbf{a}(\mathbf{z} - \mathbf{z}^* + \mathbf{b}(\mathbf{e} - \mathbf{e}^*)) = \mathbf{0} \pmod q$$
$$\implies (\mathbf{a}|\mathbf{b})\begin{bmatrix}\mathbf{z}-\mathbf{z}^*\\ \mathbf{e}-\mathbf{e}^*\end{bmatrix} = \mathbf{0} \pmod q$$

Let $\mathbf{v} = \begin{bmatrix}\mathbf{z}-\mathbf{z}^*\\ \mathbf{e}-\mathbf{e}^*\end{bmatrix}$, it can be easily verified that $\mathbf{v} \neq \mathbf{0}$, because if it is 0 then the forged signature sig^* is exactly same as the old signature sig. Also, since $0 < ||\mathbf{z}^*||, ||\mathbf{z}|| < \eta\sigma\sqrt{l+k}$ and $0 < ||\mathbf{e}^*||, ||\mathbf{e}|| < \sigma\sqrt{l+k}$, we have that $0 < ||\mathbf{v}|| < 2\sigma\sqrt{(\eta^2+1)l+k} = \mathcal{O}(n)$.

Case II If $c_1^* = r_i$ was a response to the Hash Queries, then we analyse this as follows. The simulator $\mathcal{B}$ records the forged signature $\mathsf{sig}^* = (\mathbf{c_0}^*, c_1^*, \mathbf{z}^*)$ given by the adversary $\mathcal{A}$ on the message μ^* and generates new random values $r_i', r_{i+1}', \ldots, r_w' \xleftarrow{\$} \mathcal{D}_{\mathcal{H}}$. The simulator $\mathcal{B}$ then runs again with inputs $(\mathbf{a}, \mathbf{b_0}, \mathbf{b_1}, \mathbf{R_{b_0}}, t, r_1, \ldots, r_{i-1}, r_i', r_{i+1}', \ldots, r_w', \phi, \Phi)$. By the General Forking Lemma, $\mathcal{B}$ observes the probability that $r_i' \neq r_i$ and $\mathcal{A}$ uses the random oracle response in the forgery is at least ϵ^* and hence with the probability ϵ^*, $\mathcal{A}$ outputs a new signature $\mathsf{sig} = (\mathbf{b_0}, r_i', \mathbf{z})$ of the message μ^* and $(\mathbf{a}\cdot\mathbf{z}^* - tc_1^* + \mathbf{b_1}\cdot\mathbf{e}^*||t||s^*) = (\mathbf{a}\cdot\mathbf{z} - tc_1 + \mathbf{b_1}\cdot\mathbf{e}||t||s)$ where $c_1^* = r_i, c_1 = r_i'$ and $s = s^*$. $\mathcal{B}$ also computes $\mathbf{e}$ and $\mathbf{e}^*$ from the values $\mathbf{c_0}$ and $\mathbf{c_0^*}$. Thus by putting the values $t = \mathbf{a}\cdot\mathbf{s} \pmod q$, we have

$$(\mathbf{a}\cdot\mathbf{z} - tc_1 + \mathbf{b_1}\cdot\mathbf{e}) = (\mathbf{a}\cdot\mathbf{z} - tc_1 + \mathbf{b}\cdot\mathbf{e}) = (\mathbf{a}\cdot\mathbf{z}^* - tc_1^* + \mathbf{b}\cdot\mathbf{e}^*)$$
$$\implies \mathbf{a}\cdot(\mathbf{z} - \mathbf{z}^* + \mathbf{s}c_1^* - \mathbf{s}c_1) + \mathbf{b}\cdot(\mathbf{e} - \mathbf{e}^*) = \mathbf{0} \pmod q$$
$$\implies (\mathbf{a}|\mathbf{b})\begin{bmatrix}\mathbf{z}-\mathbf{z}^*+\mathbf{s}(c_1^*-c_1)\\ \mathbf{e}-\mathbf{e}^*\end{bmatrix} = \mathbf{0} \pmod q$$

Let $\mathbf{v} = \begin{bmatrix}\mathbf{z}-\mathbf{z}^*+\mathbf{s}(c_1^*-c_1)\\ \mathbf{e}-\mathbf{e}^*\end{bmatrix}$, since $||\mathbf{e}^*||, ||\mathbf{e}|| \leq \sigma\sqrt{l+k}$; $||\mathbf{z}^*||, ||\mathbf{z}|| \leq \eta\sigma\sqrt{l+k}$; $||\mathbf{s}c_1^*||, ||\mathbf{s}c_1|| \leq d\kappa\sqrt{l+k}$, we have that $||\mathbf{v}|| \leq 2\sqrt{l+k}\sqrt{(\eta\sigma + d\kappa)^2 + \sigma^2} = \mathcal{O}(n)$.

□

Theorem 2 LaSDVS *is Non-Transferable*

Proof According to the defined security model for non-transferability of the SDVS scheme, a game between the PPT adversary $\mathcal{A}$ and the simulator $\mathcal{B}$ is defined as follows

- Setup: $\mathcal{B}$ computes the following steps.
  1. Run Setup(1^n) to output the public parameter pp.
  2. Run SignKeyGen(pp) to output $(pk_S, sk_S) = (t, \mathbf{s})$ for the signer.
  3. Run VerKeyGen(pp) to output $(pk_V, sk_V) = (\mathbf{b_0}, \mathbf{b_1}, \mathbf{R_{b_0}}, \mathbf{R_{b_1}})$.

4. Send (pk_S, pk_V) to $\mathcal{A}$.

- Challenge: For a given message μ from the adversary $\mathcal{A}$, $\mathcal{B}$ computes the following steps.
  1. Run $\mathsf{Sign}(pp, sk_S, pk_S, pk_V, \mu)$ to output a real designated verifiable signature $\mathsf{sig}^{(0)} = (\mathbf{c_0}^{(0)}, c_1^{(0)}, \mathbf{z}^{(0)})$.
  2. Run $\mathsf{Simul}(pp, pk_S, sk_V, pk_V, \mu)$ to output a simulated designated verifiable signature $\mathsf{sig}^{(1)} = (\mathbf{c_0}^{(1)}, c_1^{(1)}, \mathbf{z}^{(1)})$.
  3. Sample a random bit $b \xleftarrow{\$} \{0,1\}$ and send the message-signature pair $(\mu, \mathsf{sig}^{(b)})$ to $\mathcal{A}$.
- Output: $\mathcal{A}$ outputs a bit $b^* \in \{0,1\}$.

In the above processes, by the property of hash function $\mathcal{H}$, we assure that $c_1^{(0)}$ and $c_1^{(1)}$ are random in $\mathcal{R}_q$. As the output of the real designated verifiable signature is done using the rejection sampling, the pairs $(c_1^{(0)},\ \mathbf{z^{(0)}} = \mathbf{s}{\cdot}c_1^{(0)} + \mathbf{y})$ and $(c_1^{(1)},\ \mathbf{z^{(1)}} \xleftarrow{\$} \mathcal{D}_{\mathcal{R}_\mathbf{q}^{\mathbf{l+k}},\sigma})$ are within the statistical distance of $2^{-\omega(log(l+k))}/M$. Furthermore, $\mathbf{c_0^{(0)}} = \mathbf{b_0}{\cdot}s^{(0)} + \mathbf{e}^{(0)}$ and $\mathbf{c_0^{(1)}} = \mathbf{b_0}{\cdot}s^{(1)} + \mathbf{e}^{(1)}$ where $s^{(0)}, s^{(1)} \xleftarrow{\$} \mathcal{R}_q$, $\mathbf{e}^{(0)} \xleftarrow{\$} \mathcal{D}_{\mathcal{R}_q^{l+k},\sigma}$, and $\mathbf{e}^{(1)} \in \mathcal{D}_{\mathcal{R}_q^{l+k},\sigma}$ is coming as an output from the RingSample algorithm, hence $\mathbf{c_0^{(0)}}$ and $\mathbf{c_0^{(1)}}$ are statistically indistinguishable. The real $(\mu, \mathsf{sig}^{(0)})$ and the simulated $(\mu, \mathsf{sig}^{(1)})$ designated verifiable signatures are statistically indistinguishable to $\mathcal{A}$ and hence $\mathcal{A}$ can not determine who is the actual producer of the pair $(\mu, \mathsf{sig}^{(b)})$ which implies that the advantage of the adversary $\mathcal{A}$ is negligible. □

Theorem 3 LaSDVS *satisfies the PSI security if the* Ring-LWE *assumption holds.*

Proof According to the defined security model for *PSI* of the SDVS scheme, a game between the PPT adversary $\mathcal{A}$ and the simulator $\mathcal{B}$ is defined as follows

- Setup: $\mathcal{B}$ computes the following steps-
  1. Run $\mathsf{Setup}(1^n)$ to output the public parameter pp.
  2. Run $\mathsf{SignKeyGen}(pp)$ twice to output $(pk_{S^{(0)}}, sk_{S^{(b)}}) = (t_0, \mathsf{s}_0)$ for the signer $\mathbf{S}_0$ and $(pk_{S^{(1)}}, sk_{S^{(1)}}) = (t_1, \mathsf{s}_1)$ for the signer $\mathbf{S}_1$
  3. Run $\mathsf{VerKeyGen}(pp)$ to output $(pk_V, sk_V) = (\mathbf{b_0}, \mathbf{b_1}, \mathbf{R_{b_0}}, \mathbf{R_{b_1}})$.
  4. Send $(pp, pk_{S^{(0)}}, sk_{S^{(0)}}, pk_{S^{(1)}}, sk_{S^{(1)}}, pk_V)$ to $\mathcal{A}$.
- Simulating Queries: For a message μ and a bit $b \in \{0,1\}$ given by the adversary $\mathcal{A}$, $\mathcal{B}$ runs $\mathsf{Simul}(pp, sk_V, pk_V, pk_{S^{(b)}}, \mu)$ and returns a simulated designated verifiable signature $\mathsf{sig}^{(b)} = (\mathbf{c_0}^{(b)}, c_1^{(b)}, \mathbf{z}^{(b)})$ to $\mathcal{A}$.
- Verification Queries: Given a message-signature pair $(\mu, \mathsf{sig}^{(b)})$ and a bit $b \in \{0,1\}$ from the adversary, $\mathcal{B}$ runs the $\mathsf{Verify}(pp, sk_V, pk_V, pk_{S^{(b)}, \mathsf{sig}^{(b)}}, \mu)$ and returns a decisional value 1 for valid and 0 otherwise to $\mathcal{A}$.
- Challenge: For a message μ, $\mathcal{B}$ sends the challenge as follows-
  1. Run $\mathsf{Sign}(pp, pk_{S^{(0)}}, sk_{S^{(0)}}, pk_V, \mu)$ to output $\mathsf{sig}^{(0)} = (\mathbf{c_0}^{(0)}, c_1^{(0)}, \mathbf{z}^{(0)})$.
  2. Run $\mathsf{Sign}(pp, pk_{S^{(1)}}, sk_{S^{(1)}}, pk_V, \mu)$ to output $\mathsf{sig}^{(1)} = (\mathbf{c_0}^{(1)}, c_1^{(1)}, \mathbf{z}^{(1)})$.
  3. Sample a random bit $b \xleftarrow{\$} \{0,1\}$ and send the pair $(\mu, \mathsf{sig}^{(b)})$ to $\mathcal{A}$.
- Output: $\mathcal{A}$ outputs a bit $b^* \in \{0,1\}$.

Similar to the arguments in the proof of non-transferability, by the property of hash function $\mathcal{H}$, we assure that $c_1^{(0)}$ and $c_1^{(1)}$ are random in $\mathcal{R}_q$. As the output of the real designated verifiable signature is done using the rejection sampling, the pairs $(c_1^{(0)}, \mathbf{z^{(0)}} = \mathbf{s^{(0)}} \cdot c_1^{(0)} + \mathbf{y}^{(0)} \pmod q)$ and $(c_1^{(1)}, \mathbf{z^{(1)}} = \mathbf{s^{(1)}} \cdot c_1^{(1)} + \mathbf{y}^{(1)}) \pmod q)$ are within the same distribution. Furthermore, $\mathbf{c_0^{(0)}} = \mathbf{b_0^{(0)}} \cdot s^{(0)} + \mathbf{e}^{(0)} \pmod q$ and $\mathbf{c_0^{(1)}} = \mathbf{b_0^{(1)}} \cdot s^{(1)} + \mathbf{e}^{(1)} \pmod q$ are the LWE instances where $s^{(0)}, s^{(1)} \xleftarrow{\$} \mathcal{R}_q$, $\mathbf{e^{(0)}}, \mathbf{e^{(1)}} \xleftarrow{\$} \mathcal{D}_{\mathcal{R}_q^{l+k}, \sigma}$, hence $\mathbf{c_0^{(0)}}$ and $\mathbf{c_0^{(1)}}$ are statistically close to uniform vectors in $\mathcal{R}_q^{l+k}$. Now, given a real message-signature pair $(\mu, \mathsf{sig}^{(\mathsf{b})})$, $\mathcal{A}$ computes the following steps

1. Parse the signature $(\mu, \mathsf{sig}^{(\mathsf{b})}) = (\mu, \mathbf{c_0}^{(b)}, c_1^{(b)}, \mathbf{z}^{(b)})$.
2. Check if $0 < ||\mathbf{z}^{(b)}|| < \eta\sigma\sqrt{l+k}$; where η is a constant.

According to the definition of the Ring-LWE problem, this is obvious that without the private key $sk_V = (\mathbf{R_{b_0}}, \mathbf{R_{b_1}})$ of a designated verifier, the quantities s and $\mathbf{e}$ can not be obtained and the verification can not be accomplished by the adversary $\mathcal{A}$. Hence the PPT adversary $\mathcal{A}$ can not verify the validity of a given message-signature pair $\mu, \mathsf{sig}^{(\mathsf{b})}$ and determine which one is the real signer and thus the advantage of $\mathcal{A}$ is negligible. □

Theorem 4 LaSDVS *satisfies Non-Delgatability.*

Proof As per the security definition defined above, let us consider that there exists a black box $\mathcal{A}$ producing a designated verifiable signature in time τ' with a non-negligible probability ϵ', a game between an extractor $\mathcal{E}$ and $\mathcal{A}$ is as follows:
-if $\mathcal{A}$ produces a public key $pk_S = t \in \mathcal{R}_q$ as the target:

- Setup: $\mathcal{E}$ and $\mathcal{A}$ perform the following steps:
  1. $\mathcal{E}$ runs $\mathsf{ringGenTrap}(\tilde{\mathbf{a_0}}, h_{\mathbf{a}} = 1)$ to output $\mathbf{a} \in \mathcal{R}_q^{l+k}$ and a trapdoor $\mathbf{R_a} \in \mathcal{R}_q^{l \times k}$, and sends $\mathbf{a}$ to $\mathcal{A}$.
  2. $\mathcal{A}$ produces $pk_S = t$ and sends it to $\mathcal{E}$.
  3. $\mathcal{E}$ runs $\mathsf{ringGenTrap}(\tilde{\mathbf{b_0}}, h_{\mathbf{b_0}} = 1)$ to output $\mathbf{b_0} \in \mathcal{R}_q^{l+k}$ and a trapdoor $\mathbf{R_{b_0}} \in \mathcal{R}_q^{l \times k}$.
  4. $\mathcal{E}$ runs $\mathsf{ringGenTrap}(\tilde{\mathbf{b_1}}, h_{\mathbf{b_1}} = 1)$ to output $\mathbf{b_1} \in \mathcal{R}_q^{l+k}$ and a trapdoor $\mathbf{R_{b_1}} \in \mathcal{R}_q^{l \times k}$.
  5. $\mathcal{E}$ samples random coins Φ, and uniformly random $r_1, \ldots, r_w \xleftarrow{\$} \mathcal{D}_{\mathcal{H}}$, which will correspond to the responses of the random oracle.
  6. $\mathcal{E}$ sends the $pk_V = (\mathbf{b_0}, \mathbf{b_1})$ to $\mathcal{A}$.
- Hash Queries: $\mathcal{E}$ performs this in the same way as the simulator $\mathcal{B}$ does in the *SU-CMA* proof.
- Simulating Queries: Given a message μ, $\mathcal{E}$ specifies the following steps:
  1. Sample $s, u \xleftarrow{\$} \mathcal{R}_q$, and $\mathbf{z} \xleftarrow{\$} \mathcal{D}_{\mathcal{R}_q^{l+k}, \eta\sigma}$.
  2. Program the random oracle $\mathcal{H}(u||t||s||\mu) = c_1$.
  3. Run $\mathsf{ringSample}(\mathbf{b_1}, h_{\mathbf{b_1}}, \mathbf{R_{b_1}}, u - \mathbf{a} \cdot \mathbf{z} + tc_1, \sigma)$ to get a short $\mathbf{e} \in \mathcal{D}_{\mathcal{R}_q^{l+k}, \sigma}$.
  4. Define $\mathbf{c_0} = \mathbf{b_0}s + \mathbf{e} \pmod q$.
  5. Output $\mathsf{sig} = (\mathbf{c_0}, c_1, \mathbf{z})$ using rejection sampling technique.
- Verification Queries: Similar to the Hash Queries, extractor $\mathcal{E}$ performs this in the same way as the simulator $\mathcal{B}$ does in the *SU-CMA* proof.
- Challenge: $\mathcal{E}$ selects μ^* and $\mathcal{A}$ sends the SDVS $\mathsf{sig}^* = (\mathbf{c_0^*}, c_1^*, \mathbf{z}^*)$ for μ^* to $\mathcal{E}$.

- Output: Given (μ^*, sig^*), $\mathcal{E}$ computes-
  1. Check the validity of sig^* on μ^*.
  2. Run $\mathsf{ringSample}(\mathbf{a}, \mathbf{R_a}, t, d)$ to get a short $\mathbf{s} \in \mathcal{D}_{\mathcal{R}_q^{l+k}, d}$.

Thus, the extractor $\mathcal{E}$ determines the private key $sk_S = \mathbf{s}$ of $\mathcal{A}$ which is the pre-image of the public key $pk_S = t$. The probability of success is the same as the probability of $\mathcal{A}$ producing a valid signature in the challenge phase.

-if $\mathcal{A}$ produces the public key $pk_V = (\mathbf{b_0}, \mathbf{b_1})$ as the target:

- Setup: $\mathcal{E}$ and $\mathcal{A}$ do the following steps:
  1. $\mathcal{E}$ samples $\mathbf{a} \xleftarrow{\$} \mathcal{R}_q^{l+k}$ and sends it to $\mathcal{A}$.
  2. $\mathcal{A}$ sends $pk_V = (\mathbf{b_0}, \mathbf{b_1})$ to $\mathcal{E}$.
  3. $\mathcal{E}$ samples a uniformly random $\mathbf{s} \xleftarrow{\$} \{-d, \ldots, 0, \ldots, d\}^{l+k}$.
  4. $\mathcal{E}$ samples random coin Φ, and uniformly random $\{r_1, \ldots, r_w\} \xleftarrow{\$} \mathcal{D}_{\mathcal{H}}$, that will correspond to the output of the random oracle.
  5. $\mathcal{E}$ defines $t = \mathbf{a} \cdot \mathbf{s} \pmod q$, and sends $pk_S = t$ to $\mathcal{A}$.
- Hash Queries: $\mathcal{E}$ performs this in the same way as the simulator $\mathcal{B}$ does in the *SU-CMA* proof.
- Signing Queries: Given a message μ, $\mathcal{E}$ executes the following steps-
  1. Run the signing algorithm in Hybrid SDVS, as defined in the proof of *SU-CMA*, using the random coins Φ to produce a signature $\mathsf{sig} = (\mathbf{c_0}, c_1, \mathbf{z})$.
  2. Store the hash inputs $(\mathbf{a} \cdot \mathbf{z} - tc_1 + \mathbf{b_1} \cdot \mathbf{e} || t || s || \mu)$ and output c_1 into $l_{\mathcal{H}}$.
  3. Return sig to $\mathcal{A}$.
- Challenge: $\mathcal{E}$ selects μ^*, and $\mathcal{A}$ sends the corresponding SDVS $\mathsf{sig}^* = (\mathbf{c_0^*}, c_1^*, \mathbf{z}^*)$ to $\mathcal{E}$.
- Output: The extractor $\mathcal{E}$ first checks the validity of $\mathsf{sig}^* = (\mathbf{c_0^*}, c_1^*, \mathbf{z}^*)$ on μ^* and adopts the same method as the simulator $\mathcal{B}$ does in the proof of *SU-CMA* by programming the random oracle $\mathcal{H}$ while signing a message. So, we have, due to the collision-resistant property of the hash function $\mathcal{H}$,

$$(\mathbf{a} \cdot \mathbf{z}^* - tc_1^* + \mathbf{b_1} \cdot \mathbf{e}^* || t || s^* || \mu^*) = (\mathbf{a} \cdot \mathbf{z} - tc_1 + \mathbf{b_1} \cdot \mathbf{e} || t || s || \mu)$$

by holding the conditions $c_1^* = c_1, \mathbf{z}^* = \mathbf{z}$, and $s^* = s$. The extractor $\mathcal{E}$, using the values $\mathbf{b_0}, s$ and $\mathbf{b_0^*}, s^*$ computes the values $\mathbf{e}$ and $\mathbf{e}^*$ respectively. Hence, we have

$$\mathbf{a} \cdot \mathbf{z} - tc_1 + \mathbf{b_1} \cdot \mathbf{e} = \mathbf{a} \cdot \mathbf{z}^* - tc_1^* + \mathbf{b_1} \cdot \mathbf{e}^*$$

$$\implies \mathbf{b_1} \cdot (\mathbf{e}^* - \mathbf{e}) = \mathbf{a} \cdot (\mathbf{z} - \mathbf{z}^*) + t(c_1^* - c_1) \pmod q$$

$$\implies \mathbf{b_1} \cdot (\mathbf{e}^* - \mathbf{e}) = 0 \pmod q$$

.

Now, let us assume that $\mathbf{v} = \mathbf{e}^* - \mathbf{e}$, then it can be easily checked that $\mathbf{v} \neq 0$, otherwise, if it is not so, then the forged signature sig^* is exactly as same as the programmed signature sig provided by the challenger during the signing queries. Therefore, the term $\mathbf{v} = (\mathbf{e}^* - \mathbf{e}) \in \mathcal{R}_q^{l+k}$ is a vector of polynomials of short norm. After replaying the above process at least k times, the extractor can get a matrix $\mathbf{V} \in \mathcal{R}_q^{(l+k) \times k}$, columns of which are of short norms.

Since, $\mathbf{b_1} = (\mathbf{b}_{1_0}^T, \mathbf{b}_{1_1}^T = h\mathbf{g}^T - \mathbf{b}_{1_0}^T \mathbf{R}_{\mathbf{b_1}})^T \in \mathcal{R}_q^{l+k}$, where $\mathbf{b}_{1_0}^T \in \mathcal{R}_q^l$ and $\mathbf{b}_{1_1}^T \in \mathcal{R}_q^k$. Therefore $\mathbf{b_1} \cdot \mathbf{V} = (\mathbf{b}_{1_0}^T, \mathbf{b}_{1_1}^T = h\mathbf{g}^T - \mathbf{b}_{1_0}^T \mathbf{R}_{\mathbf{b_1}})^T \cdot \mathbf{V} = 0 \pmod q$

Let $\mathbf{V} = \begin{bmatrix} \mathbf{V^{(0)}} \\ \mathbf{V^{(1)}} \end{bmatrix}$, where $\mathbf{V^{(0)}} \in \mathcal{R}_q^{l \times k}$ and $\mathbf{V^{(1)}} \in \mathcal{R}_q^{k \times k}$.

So, solving the equation

$$\mathbf{b_1} \cdot \mathbf{V} = (\mathbf{b}_{1_0}^T, \mathbf{b}_{1_1}^T = h\mathbf{g}^T - \mathbf{b}_{1_0}^T \mathbf{R}_{\mathbf{b_1}})^T \cdot \begin{bmatrix} \mathbf{V^{(0)}} \\ \mathbf{V^{(1)}} \end{bmatrix} = 0 \pmod{q}$$

$$\implies \mathbf{b}_{1_0}^T \cdot \mathbf{V^{(0)}} + (h\mathbf{g}^T - \mathbf{b}_{1_0}^T \mathbf{R}_{\mathbf{b_1}})^T \cdot \mathbf{V^{(1)}} = 0 \pmod{q}$$

the extractor $\mathcal{E}$ can retrieve the private key $\mathbf{R}_{\mathbf{b_1}}$ of the adversary $\mathcal{A}$.

□

6 Conclusion

In this work, we propose a post-quantum SDVS denoted as LaSDVS based on ideal lattice assumptions, namely Ring-SIS and Ring-LWE. LaSDVS provides strong security guarantees, including strong unforgeability under chosen-message attacks, non-transferability, non-delegatability, and signer anonymity. The signature and key sizes were minimized without compromising security. A signature size of $\mathcal{O}(n \log q)$ was achieved, which constituted a quadratic reduction compared to the conventional $\mathcal{O}(n^2)$ lattice-based SDVS schemes, resulting in a reduction by a factor of $n/\log q$. It is demonstrated that LaSDVS outperforms existing post-quantum SDVS designs in terms of efficiency and compactness.

References

[1] Pauline Bert, Pierre-Alain Fouque, Adeline Roux-Langlois, and Mohamed Sabt. Practical implementation of ring-sis/lwe based signature and ibe. In Tanja Lange and Rainer Steinwandt, editors, *Post-Quantum Cryptography*, pages 271–291, Cham, 2018. Springer International Publishing.

[2] Jie Cai, Han Jiang, Pingyuan Zhang, Zhihua Zheng, Guangshi Lyu, and Qiuliang Xu. An efficient strong designated verifier signature based on $\mathcal{R}-$ sis assumption. *IEEE Access*, 7:3938–3947, 2019.

[3] Jintai Ding, Jason E Gower, and Dieter S Schmidt. *Multivariate public key cryptosystems*, volume 25. Springer Science & Business Media, 2006.

[4] Craig Gentry, Chris Peikert, and Vinod Vaikuntanathan. Trapdoors for hard lattices and new cryptographic constructions. Cryptology ePrint Archive, Paper 2007/432, 2007.

[5] Jeffrey Hoffstein, Jill Pipher, , and Joseph H. Silverman. *An Introduction to Mathematical Cryptography*. Springer New York, NY, New York, 2nd edition, 2014.

[6] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In Ueli Maurer, editor, *Advances in Cryptology — EUROCRYPT '96*, pages 143–154, Berlin, Heidelberg, 1996. Springer Berlin Heidelberg.

[7] Jonathan Katz. *Digital signatures*, volume 1. Springer, 2010.

[8] Fabien Laguillaumie and Damien Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In Carlo Blundo and Stelvio Cimato, editors, *Security in Communication Networks*, pages 105–119, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[9] Russell W. F. Lai, Henry K. F. Cheung, and Sherman S. M. Chow. Trapdoors for ideal lattices with applications. In Dongdai Lin, Moti Yung, and Jianying Zhou, editors, *Information Security and Cryptology*, pages 239–256, Cham, 2015. Springer International Publishing.

[10] Helger Lipmaa, Guilin Wang, and Feng Bao. Designated verifier signature schemes: Attacks, new security notions and a new construction. In Luís Caires, Giuseppe F. Italiano, Luís Monteiro, Catuscia Palamidessi, and Moti Yung, editors, *Automata, Languages and Programming*, pages 459–471, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[11] Daniele Micciancio and Chris Peikert. Trapdoors for lattices: Simpler, tighter, faster, smaller. Cryptology ePrint Archive, Paper 2011/501, 2011.

[12] Sweta Mishra, Bhaskar Mondal, and Rishi Kumar Jha. A survey on isogeny-based cryptographic protocols. *Wireless Networks*, 31(3):2993–3024, 2025.

[13] National Institute of Standards and Technology. Nist announces report on post-quantum cryptography (pqc). https://csrc.nist.gov/pubs/ir/8105/final, 2016. Accessed: April, 2016.

[14] Corporate Nist. The digital signature standard. *Communications of the ACM*, 35(7):36–40, 1992.

[15] Geontae Noh and Ik Rae Jeong. Strong designated verifier signature scheme from lattices in the standard model. *Security and Communication Networks*, 9(18):6202–6214, 2016.

[16] Raphael Overbeck and Nicolas Sendrier. Code-based cryptography. In *Post-quantum cryptography*, pages 95–145. Springer, 2009.

[17] Oded Regev. On lattices, learning with errors, random linear codes, and cryptography. In *Proceedings of the Thirty-Seventh Annual ACM Symposium on Theory of Computing*, STOC '05, page 84–93, New York, NY, USA, 2005. Association for Computing Machinery.

[18] Peter W. Shor. Polynomial-time algorithms for prime factorization and discrete logarithms on a quantum computer. *SIAM Journal on Computing*, 26(5):1484–1509, 1997.

[19] Vikas Srivastava, Anubhab Baksi, and Sumit Kumar Debnath. An overview of hash based signatures. Cryptology ePrint Archive, Paper 2023/411, 2023.

[20] Fenghe Wang, Yupu Hu, and Baocang Wang. Lattice-based strong designate verifier signature and its applications. *Malaysian Journal of Computer Science*, 25(1):11–22, Mar. 2012.

[21] Yanhua Zhang, Willy Susilo, and Fuchun Guo. Lattice-based strong designated verifier signature with non-delegatability. *Computer Standards & Interfaces*, 92:103904, 2025.