Strong Designated Verifier Signatures with Non-delegatability from CSIDH[⋆]

Hiroki Minamide[1,2], Keisuke Tanaka[2], and Masayuki Tezuka[2]

[1] National Institute of Technology, Tokyo College
[2] Institute of Science Tokyo
minamide@tokyo-ct.ac.jp, keisuke@is.titech.ac.jp,
tezuka.m.eab3@m.isct.ac.jp

Abstract. Designated verifier signature allows a signer to designate a verifier who can verify the signature. Strong designated verifier signature (SDVS) enhances privacy by ensuring that the signature itself does not leak information about the signer's identity to anyone other than the designated verifier. Non-delegatability is a property, as it prevents the signer's ability to generate valid signatures from being delegated to others. This property is crucial for SDVS applications such as e-voting. To date, post-quantum SDVS schemes with non-delegatability have been proposed. These schemes are lattice-based or hash-based schemes. While isogeny-based SDVS schemes have been proposed, none of the existing works provide a proof of non-delegatability.

In this paper, we present the first isogeny-based SDVS scheme with a formal proof of non-delegatability. Our construction uses the quadratic twists of elliptic curves. The security of our scheme is proven under the commutative supersingular isogeny gap Diffie-Hellman assumption and the group action inversion problem assumption in the random oracle model.

Keywords: Strong designated verifier signature · Non-delegatability · Commutative supersingular isogeny Diffie-Hellman · Quadratic twist

1 Introduction

1.1 Background

Designated Verifier Signatures. Designated verifier signature (DVS) introduced by Jakobsson, Sako, and Impagliazzo [18] allows a signer to designate a verifier who can verify the signature. The designated verifier uses its own secret key to verify a signature. Typically, a DVS scheme satisfies unforgeability and non-transferability. Unforgeability guarantees that if a user has neither the signer's nor the designated verifier's secret key, it is computationally infeasible to

[⋆] A part of this work was supported by JSPS KAKENHI JP24H00071, JST CREST JPMJCR2113, and JST K Program JPMJKP24U2. This paper is the full version of the paper presented in ICISC 2025.

generate a valid signature for the designated verifier. Non-transferability allows the designated verifier to generate simulated signatures that are indistinguishable from those generated by the signer. Thanks to these properties, DVS has found applications in a variety of privacy-preserving contexts, including deniable authentication [39], e-voting [18], and undeniable signature [10].

Strong Designated Verifier Signatures. Non-transferability guarantees that one cannot distinguish whether a signature was generated by the signer or by the designated verifier. However, this property alone does not prevent a third party from identifying the two potential signers. That is, signatures generated by different signers may still be distinguishable to anyone other than the designated verifier. Saeednia, Kremer, and Markowitch [28] introduced strong designated verifier signatures (SDVS). Laguillaumie and Vergnaud [23] formalized the security notion called privacy of signer's identity for SDVS to enhance the signer's privacy. Privacy of signer's identity ensures that a signature does not reveal the identity of the signer to anyone other than the designated verifier.

Non-Delegatability. Lipmaa, Wang, and Bao [24] pointed out a weakness in the informal definition of DVS [18]. A weakness of DVS is that a signer can delegate the ability to generate valid signatures for a specific designated verifier to a third party, without disclosing their secret key. This weakness is undesirable in certain applications of SDVS. For example, we consider the application of SDVS for an e-voting system. A voter is a signer who sends a signed ballot, and an election authority is a verifier that verifies the ballot. If SDVS has this weakness, the signer could delegate their voting capability to a third party, making it impossible to guarantee that a ballot was generated by the real voter. To address such an issue, Lipmaa et al. [24] introduced non-delegatability for DVS. Non-delegatability requires that a valid signature be the proofs of knowledge of either the signer's secret key or the designated verifier's secret key. After the proposal by Lipmaa et al., non-delegatability has become a standard security notion for SDVS.

1.2 Motivation

Post-Quantum SDVS with Non-Delegatability. Upon until now, there have been a sequence of works for constructing SDVS under various assumptions, including pairing-free group [28,38,16,35,40], pairing-based [17,1,36,37], isogeny [33,27], lattice-based [7,26,41], code-based [30,2], hash-based [34] and generic construction [11,14]. Despite this extensive line of above works on SDVS, there exist only a few post-quantum constructions of SDVS with non-delegatability. To the best of our knowledge, only lattice-based SDVS schemes and [26,41] and a hash-based SDVS scheme [34] were proposed.

Current Open Question. In light of previous works [11,33,14,27], the following question remains open in isogeny-based cryptography:

Is it possible to construct an isogeny-based SDVS with non-delegatability?

1.3 Our Contributions.

Contributions. In this paper, we give an affirmative answer to the question. More precisely, we propose an SDVS with non-delegatability from the commutative supersingular isogeny Diffie-Hellman (CSIDH) class group action. The security of our scheme is proven under the group action inversion problem (GAIP) assumption and the commutative supersingular isogeny gap Diffie-Hellman (CSI-GDH) assumption in the random oracle model (ROM) [5].

Comparison in Previous Works. We provide comparisons among our scheme, generic SDVS constructions [11,14], and isogeny-based SDVS constructions [33,27] in Fig. 1.

Scheme	Assumption	Unf	NT	PSI	ND
[11]	Ring Signature + IND-CPA PKE	✓	✓	✓$^\sharp$	?$^\flat$
[11]	Deniable Authenticated Key Exchange	✓	✓	✓$^\sharp$	?$^\flat$
[14]	IND-CCA KEM + PRF	✓	✓	✓	×
[33] (insecure†)	SS-CDH + SS-DDH + ROM	–†	–†	–†	–†
[27]	MT-GAIP + ROM	✓	✓	✓	?$^\flat$
$\mathsf{SDVS}_{\mathsf{Ours}}$	GAIP + CSI-GDH + ROM	✓	✓	✓	✓

Fig. 1. Comparisons among our scheme, the generic construction, and isogeny-based SDVS schemes in previous works. In the column "Assumption", "IND-CCA KEM" represents an IND-CCA secure key encapsulation mechanism. "PRF" represents a pseudorandom function. "SS-DDH (resp. SS-CDH)" represents the supersingular decisional (resp. computational) Diffie-Hellman assumption. "MT-GAIP" represents the multi-target group action inversion problem assumption. In the column "Unf", ✓ represents that the corresponding scheme satisfies unforgeability. In the column "NT", ✓ represents that the corresponding scheme satisfies non-transferability. In the column "PSI", ✓ represents that the corresponding scheme satisfies the privacy of signer's identity. In the column "ND", ✓ represents that the corresponding scheme satisfies non-delegatability. $\sharp$: [11] proved privacy of signer's identity in a weak security model. $\flat$: [11,27] did not prove non-delegatability for the constructions. $\dagger$: The scheme [33] is insecure. The SS-CDH and SS-DDH assumptions are broken by the attack of Castryck and Decru [8].

There are three generic constructions of SDVS. Two of them were proposed by Feng, Xu, and Chen [11]: one based on a ring signature scheme combined with indistinguishability under chosen-plaintext attacks (IND-CPA) public-key encryption (PKE), and the other based on a deniable authenticated key exchange protocol. However, the privacy of signer's identity of these constructions was proven in a weak security model, where the adversary is not allowed to access the signing oracle or the verification oracle after receiving the challenge signature. Moreover, [11] did not provide a security analysis for non-delegatability.

The generic construction of SDVS proposed by Gong, Au, Xue [14], which is based on the indistinguishability under chosen ciphertext attacks (IND-CCA) secure key encapsulation mechanism (KEM) enables the derivation of several isogeny-based SDVS schemes.[3] However, their generic construction does not satisfy non-delegatability.[4]

There are two isogeny-based SDVS constructions. An isogeny-based SDVS scheme by Sun, Tian, and Wang [33] is insecure. This scheme relies on the supersingular Diffie-Hellman assumption, which was broken by an attack of Castryck and Decru [8].

Renan [27] proposed an isogeny-based SDVS based on a commutative group action and provided a security analysis with respect to unforgeability, non-transferability, and privacy of the signer's identity. However, no security analysis was given for non-delegatability.

Thus, to the best of our knowledge, the existing generic constructions [11,14] and isogeny-based constructions [33,27] either do not satisfy non-delegatability or lack a formal security proof for non-delegatability. Our scheme is the first isogeny-based SDVS scheme with a formal proof of non-delegatability.

1.4 How to Obtain Our Scheme.

DH Key Exchange + OR Proof Approach. Huang, Yang, Wong, and Susilo [16] proposed an SDVS scheme with non-delegatability, constructed from the combination of the Diffie-Hellman key exchange and a Σ-protocol for the OR relations.

We outline the idea of their gap Diffie-Hellman (GDH) based SDVS construction [16]. Let $\mathbb{G}$ be a cyclic group of prime order p and g be a generator of $\mathbb{G}$. Let $(\mathrm{pk}_s = g^{x_s}, \mathrm{sk}_s = x_s)$ be a public/secret key of the signer and $(\mathrm{pk}_v = g^{x_v}, \mathrm{sk}_v = x_v)$ be a public/secret key of the verifier. In their scheme, a signature σ is an OR proof derived via the Fiat-Shamir transformation [12] from the Σ-protocol for the OR relation, which proves knowledge of either sk_s or sk_v. To ensure that only the designated verifier can verify the signature, $K = \mathrm{pk}_v^{x_s}$ is included in the hash input when transforming the Σ-protocol to a non-interactive proof. As a result, the only designated verifier can compute $K = \mathrm{pk}_s^{x_v}$ and thus verify the signature. In contrast, any other party (excluding the signer and the designated verifier) cannot compute $K = g^{xy}$ from under the hardness of the CDH problem (more precisely, the hardness of the GDH problem). Therefore any other party cannot verify the signature.

Problem in Abstracting SDVS Scheme [16] from Group Actions. One might expect that their scheme could be directly abstracted using group actions. The GDH assumption can be replace the CSI-GDH assumption, which

[3] By applying the Fujisaki-Okamoto transformation [13,15] to a public key encryption scheme such as [25,3], an IND-CCA secure KEM can be obtained. The resulting KEM can be used in the generic construction.

[4] In scheme [14], the signer and the designated verifier share a pseudorandom function (PRF) key K via KEM, and a signature σ on a message m is computed as $\sigma = PRF_K(m)$. Any party to which K is delegated can generate signatures on behalf of the signer, which shows that the scheme does not satisfy non-delegatability.

is the analog of the GDH assumption in the commutative supersingular isogeny setting. However, the direct abstraction of their scheme is not possible due to the following reason. Their scheme needs the computation $g^z \cdot \mathrm{pk}^c$ where $z, c \in \mathbb{Z}_p$. Let us denote a group action as $* : G \times X \to X$. By using the group action, g^z can be represented as $z * g$ and pk^c as $c * \mathrm{pk}$. However, since both $z * g$ and $c * \mathrm{pk}$ belong to X, there is no way to define an operation between $z * g$ and pk^c. Consequently, we cannot directly abstract their SDVS scheme by group actions.

Our Solution. To obtain an isogeny-based SDVS, we employ the quadratic twist of an elliptic curve, which provides a richer structure than an abstract group action. Recently, Katsumata, Lai, LeGrow, and Qin [19,20] proposed the quadratic twist-based Σ-protocol of the OR relation. By using their Σ-protocol, we address the above problem. We follow the blueprint of the SDVS construction [16], adapting it to the commutative supersingular isogeny Diffie-Hellman (CSIDH) [9] based key exchange protocol and Σ-protocol by Katsumata et al. [19,20]. Consequently, we obtain our SDVS scheme $\mathsf{SDVS}_{\mathsf{Ours}}$.

Our Paper and Concurrent Work. As concurrent work, Khuc et al. [22] proposed isogeny-based SDVS constructions with non-delegatability. The one construction is obtained by modifying the isogeny-based signature scheme CSI-FiSh [6] and security is proven under the GAIP assumption in the ROM. Moreover, they gave the optimized variant of the construction by using the MT-GAIP assumption in the ROM.

1.5 Organization

In Section 2, we introduce notation and briefly review the background of elliptic curves and the forking lemma. In Section 3, we review the definition of SDVS and its security notions. In Section 4, we propose our SDVS scheme with non-delegatability and prove the security. In Appendix A, we review the Σ-protocol by Katsumata et al. [19,20], which is the basis protocol for our construction.

2 Preliminaries

In this section, we introduce notation, briefly review the background of elliptic curves, and recall the forking lemma.

2.1 Notation

Let 1^λ be the security parameter. A function $f(\lambda)$ is negligible in λ if $f(\lambda)$ tends to 0 faster than $\frac{1}{\lambda^c}$ for every constant $c > 0$. We write $f(\lambda) = \mathsf{negl}(\lambda)$ to denote that $f(\lambda)$ is a negligible function. We write $f(\lambda) = \mathsf{poly}(\lambda)$ to denote that $f(\lambda)$ is a polynomial function in λ.

For a finite set S, $s \xleftarrow{\$} S$ denotes that an element s is sampled uniformly at random from S, and $\#S$ denotes the number of elements in S. We denote the set of all finite binary strings by $\{0,1\}^*$. For a vector $\mathbf{v}$, $\mathbf{v}[i]$ denotes the i-th element of $\mathbf{v}$. For an algorithm A, we write $y \leftarrow \mathsf{A}(x)$ to indicate that A outputs y on input x. When explicitly showing that A uses randomness r, we write $y \leftarrow \mathsf{A}(x;r)$. We abbreviate probabilistic polynomial time as PPT.

In this paper, we use standard code-based security games. A game Game_A is a probabilistic experiment in which an adversary A interacts with an implicit challenger C that answers oracle queries issued by A. We denote the output $b \in \{0,1\}$ of the game Game between a challenger and an adversary A as $\mathsf{Game}_\mathsf{A} \Rightarrow b$. We say that A wins the game if $\mathsf{Game}_\mathsf{A} \Rightarrow 1$.

2.2 Elliptic Curves and Hard Homogeneous Spaces

We briefly review the ideal class group action on supersingular elliptic curves, hard homogeneous space, quadratic twist, and computational assumptions.

Prime and Finite Field. Let $p = 4\ell_1 \cdots \ell_n - 1$ be a cryptographically large prime satisfying $p \equiv 3 \pmod 8$, where $\ell_1, \dots, \ell_n$ are distinct small odd primes. Let $\mathbb{F}_p$ denote the finite field with p elements. All computations in our construction are carried out over $\mathbb{F}_p$. In this work, we consider the parameter setting the same as CSIDH [9]. For instance, the CSIDH-512 parameter set chooses $n = 74$, $\ell_1 = 3$, $\ell_{73} = 373$, and $\ell_{74} = 587$.

Supersingular Montgomery Curves. Let p be a prime that satisfies $p \equiv 3 \pmod 8$ and $p > 5$. We define the set of supersingular elliptic curves of Montgomery type as

$$\mathrm{Ell}_p := \left\{ E_A : y^2 = x^3 + Ax^2 + x \;\middle|\; A \in \mathbb{F}_p,\ A \neq \pm 2,\ \#E_A(\mathbb{F}_p) = p+1 \right\}.$$

Here, $\#E_A(\mathbb{F}_p) = p+1$ means that the equation of E_A has exactly p distinct solutions $(x,y) \in \mathbb{F}_p^2$, together with one distinguished point at infinity. A simple and concrete example of such a curve is $E_0 : y^2 = x^3 + x$.

Group Action. Rather than relying on advanced algebraic geometry, we emphasize a key fact: all curves in Ell_p form a single orbit under the action of the ideal class group $\mathrm{Cl}_p := \mathrm{Cl}\big(\mathbb{Z}[\sqrt{-p}]\big)$, associated with the quadratic imaginary field $\mathbb{Q}(\sqrt{-p})$. It is known that the group Cl_p is finite, abelian, and has cardinality asymptotically $\#\mathrm{Cl}_p \approx \sqrt{p}$.

$$\mathrm{Cl}_p \times \mathrm{Ell}_p \longrightarrow \mathrm{Ell}_p, \quad ([\mathfrak{a}], E_A) \longmapsto [\mathfrak{a}] * E_A,$$

where $E_A \in \mathrm{Ell}_p$ and $[\mathfrak{a}] \in \mathrm{Cl}_p$. This action is free and transitive:

- **Free:** The action is called free if the condition $[\mathfrak{a}] * E = E$ for some $E \in \mathrm{Ell}_p$ implies that $[\mathfrak{a}]$ is the identity element $[(1)] \in \mathrm{Cl}_p$. In other words, no non-trivial element of Cl_p fixes a point of Ell_p.
- **Transitive:** The action is called transitive if for any two elements $E, E' \in \mathrm{Ell}_p$, there exists a $[\mathfrak{a}] \in \mathrm{Cl}_p$ such that $[\mathfrak{a}] * E = E'$. This means that the entire set Ell_p forms a single orbit under the group action.

Hard Homogeneous Space. The pair $(\mathrm{Cl}_p, \mathrm{Ell}_p)$ is an example of a hard homogeneous space (HHS) [9]. While computing $[\mathfrak{a}] * E_A$ for a given ideal class is efficient, inverting the action, i.e., recovering $[\mathfrak{a}]$ from $(E_A, [\mathfrak{a}]*E_A)$, is conjectured to be hard, forming the basis of security. This inversion problem is called the group action inversion problem (GAIP).

Quadratic Twist. In the CSIDH setting, each supersingular Montgomery curve admits a unique quadratic twist. For a prime $p \equiv 3 \pmod 4$, the quadratic twist of a curve E_A is denoted by $(E_A)^{-1}$ and is explicitly given by

$$(E_A)^{-1} = E_{-A} : y^2 = x^3 - Ax^2 + x.$$

This twist plays a key role in enabling certain optimizations and symmetry arguments. The special curve E_0 is self-dual under this operation.

The following lemma formalizes the natural compatibility of this twist operation with the class group action.

Lemma 1. *Let $[\mathfrak{a}] \in \mathrm{Cl}_p$ be an ideal class and $E_A \in \mathrm{Ell}_p$ a supersingular Montgomery curve. Then:*

$$([\mathfrak{a}] * E_A)^{-1} = [\mathfrak{a}]^{-1} * E_A^{-1} = [\mathfrak{a}]^{-1} * E_{-A}.$$

This lemma states that twisting commutes with the class group action: applying an ideal class and then twisting is equivalent to twisting first and applying the inverse class.

Now, we introduce computational assumptions for the ideal class group action on supersingular elliptic curves.

Definition 1 (CSI-GDH Assumption [19,20]). *The commutative supersingular isogeny gap Diffie-Hellman (CSI-GDH) assumption holds on $(\mathrm{Cl}_p, \mathrm{Ell}_p)$, if for any PPT adversary A,*

$$\mathsf{Adv}_{\mathsf{A}}^{\mathsf{CSI\text{-}GDH}}(\lambda) := \Pr\left[E' = [\mathfrak{x}\mathfrak{y}] * E : \begin{array}{l} [\mathfrak{x}], [\mathfrak{y}] \xleftarrow{\$} \mathrm{Cl}_p, \\ E' \leftarrow \mathsf{A}^{\mathcal{O}^{\mathsf{CSI\text{-}DDH}}(\cdot,\cdot,\cdot)}](E, [\mathfrak{x}] * E, [\mathfrak{y}] * E) \end{array}\right]$$

is negl(λ) where $\mathcal{O}^{\mathsf{CSI\text{-}DDH}}$ is the oracle that takes a tuple $([\mathfrak{a}] * E, [\mathfrak{b}] * E, E')$ and returns 1 if $E' = [\mathfrak{a}\mathfrak{b}] * E$ and 0 otherwise.

Definition 2 (Group Action Inversion Problem Assumption [21]). *The group action inversion problem (GAIP) assumption holds on $(\mathrm{Cl}_p, \mathrm{Ell}_p)$, if for any PPT adversary A,*

$$\mathsf{Adv}_{\mathsf{A}}^{\mathsf{GAIP}}(\lambda) := \Pr[E' = [\mathfrak{x}'] * E_0 : [\mathfrak{x}] \xleftarrow{\$} \mathrm{Cl}_p, E' \leftarrow [\mathfrak{x}] * E_0, [\mathfrak{x}'] \leftarrow \mathsf{A}(E_0, E')]$$

is $\mathsf{negl}(\lambda)$.

2.3 Forking Lemma

We review the forking lemma [4], which is used in the security analysis of our SDVS scheme.

Lemma 2 (Forking Lemma [4]). *Fix an integer q and a set H of size $\#H \geq 2$. Let W be a randomized algorithm that takes as input a string x and elements $h_1, \ldots, h_q \in H$. It outputs an integer $j \in \{0, \ldots, q\}$ and a side output* out. *Let GI be a randomized algorithm called the input generator that takes a security parameter 1^λ and outputs a string x. The forking algorithm Fork^W associated with W is defined in Fig. 2.*

```
$\mathsf{Fork}^\mathsf{W}(x)$ :
  Pick random coin $\rho$ for $\mathsf{W}$ at random
  $h_1, \ldots, h_q \xleftarrow{\$} H$, $(j, \mathrm{out}) \leftarrow \mathsf{W}(x, h_1, \ldots, h_q; \rho)$
  If $j = 0$, return $(0, \perp, \perp)$
  $\widetilde{h}_1, \ldots, \widetilde{h}_q \xleftarrow{\$} H$, $(\widetilde{j}, \widetilde{\mathrm{out}}) \leftarrow \mathsf{W}(x, h_1, \ldots, h_{j-1}, \widetilde{h}_j, \ldots, \widetilde{h}_q; \rho)$
  If $j = \widetilde{j} \wedge h_j \neq \widetilde{h}_j$, return $(1, \mathrm{out}, \widetilde{\mathrm{out}})$
  Otherwise, return $(0, \perp, \perp)$
```

Fig. 2. The forking algorithm Fork^W associated to W.

Let $\mathsf{acc}(\lambda) := \Pr[j \geq 1 : x \leftarrow \mathsf{GI}(1^\lambda), h_1, \ldots, h_q \xleftarrow{\$} H, (j, \mathrm{out}) \leftarrow \mathsf{W}(x, h_1, \ldots, h_q)]$ *and* $\mathsf{frk}(\lambda) := \Pr[b = 1 : x \xleftarrow{\$} \mathsf{GI}(1^\lambda), (b, \mathrm{out}, \widetilde{\mathrm{out}}) \leftarrow \mathsf{Fork}^\mathsf{W}(x)]$. *Then* $\mathsf{frk}(\lambda) \geq \mathsf{acc}(\lambda) \cdot \left(\frac{\mathsf{acc}(\lambda)}{q} - \frac{1}{\#H}\right)$ *holds.*

3 Strong Designated Verifier Signatures

In this section, we review the definition of a strong designated verifier signature (SDVS) scheme and its security notions.

3.1 Syntax

We recall the definition of a SDVS scheme.

Definition 3. *A strong designated verifier signature scheme* SDVS *is a tuple of PPT algorithms* $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$.

- $\mathsf{KGen}(1^\lambda)$: *The key generation algorithm takes as input a security parameter 1^λ. It outputs a public key* pk *and a secret key* sk.
- $\mathsf{Sign}(\mathrm{sk}_s, \mathrm{pk}_s, \mathrm{pk}_v, m)$: *The signing algorithm takes as input a secret key of a signer* sk_s*, a public key of the signer* pk_s*, a public key of a verifier* pk_v*, and a message m. It outputs a signature σ.*
- $\mathsf{Ver}(\mathrm{sk}_v, \mathrm{pk}_s, \mathrm{pk}_v, m, \sigma)$: *The verification algorithm takes as input a secret key of a verifier* sk_v*, a public key of the verifier* pk_v*, a public key of a signer* pk_s*, a message m, and a signature σ. It outputs a bit $b \in \{0, 1\}$.*

Correctness. We require $\mathsf{SDVS} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Ver})$ to satisfy the following correctness. For any $\lambda \in \mathbb{N}$, $(\mathrm{pk}_s, \mathrm{sk}_s) \leftarrow \mathsf{KGen}(1^\lambda)$, $(\mathrm{pk}_v, \mathrm{sk}_v) \leftarrow \mathsf{KGen}(1^\lambda)$, for any $m \in \{0,1\}^*$, and $\sigma \leftarrow \mathsf{Sign}(\mathrm{sk}_s, \mathrm{pk}_s, \mathrm{pk}_v, m)$, $\mathsf{Ver}(\mathrm{sk}_v, \mathrm{pk}_s, \mathrm{pk}_v, m, \sigma) = 1$ holds.

3.2 Security

In this section, we review security notions for a SDVS scheme. First, we introduce the definition of forgeability. Our definition is obtained by modifying the unforgeability definition in [36] to capture strong unforgeability.

Definition 4 (Strong Unforgeability). *Strong unforgeability of* SDVS *is defined by the following strong unforgeability game* $\mathsf{Game}^{\mathsf{SUF}}_{\mathsf{SDVS},\mathsf{A}}(1^\lambda)$ *between a challenger* C *and an adversary* A*.*

– C *initializes a list* $L^{\mathsf{Sign}} \leftarrow \{\}$*, runs* $(\mathrm{pk}_s, \mathrm{sk}_s) \leftarrow \mathsf{KGen}(1^\lambda)$*,* $(\mathrm{pk}_v, \mathrm{sk}_v) \leftarrow \mathsf{KGen}(1^\lambda)$*, and gives* $(\mathrm{pk}_s, \mathrm{pk}_v)$ *to* A*.*
– A *is allowed to accesses oracles* $\mathcal{O}^{\mathsf{Sign}}(\cdot)$ *and* $\mathcal{O}^{\mathsf{Verify}}(\cdot,\cdot)$ *polynomially many times:*
  - $\mathcal{O}^{\mathsf{Sign}}(\cdot)$ *: For a query on* m*, it runs* $\sigma \leftarrow \mathsf{Sign}(\mathrm{sk}_s, \mathrm{pk}_s, \mathrm{pk}_v, m)$*, updates* $L^{\mathsf{Sign}} \leftarrow L^{\mathsf{Sign}} \cup \{(m, \sigma)\}$ *and returns* σ *to* A*.*
  - $\mathcal{O}^{\mathsf{Verify}}(\cdot,\cdot)$ *: For a query on* (m, σ)*, it computes* $b \leftarrow \mathsf{Ver}(\mathrm{sk}_v, \mathrm{pk}_s, \mathrm{pk}_v, m, \sigma)$ *and returns* b *to* A*.*

– *Finally,* A *outputs a forgery* (m^*, σ^*)*.*
– A *wins the game if the following condition holds:*

$$(m^*, \sigma^*) \notin L^{\mathsf{Sign}} \wedge \mathsf{Ver}(\mathrm{sk}_v, \mathrm{pk}_s, \mathrm{pk}_v, m^*, \sigma^*) = 1$$

The advantage of A *is defined as* $\mathsf{Adv}^{\mathsf{SUF}}_{\mathsf{SDVS},\mathsf{A}}(\lambda) := \Pr[\mathsf{Game}^{\mathsf{SUF}}_{\mathsf{SDVS},\mathsf{A}}(1^\lambda) \Rightarrow 1]$*. We say* SDVS *satisfies strongly unforgeability if for any PPT adversary* A*,* $\mathsf{Adv}^{\mathsf{SUF}}_{\mathsf{SDVS},\mathsf{A}}(\lambda) = \mathsf{negl}(\lambda)$*.*

Second, we review the definition of non-transferability. This security notion is formalized by [31,32]. We refer to the non-transferability definition in [29].

Definition 5 (Non-Transferability [29]). *We say* SDVS *satisfies perfect non-transferability if there exists a PPT simulation algorithm* Sim *such that for any two pairs of keys* $(\mathrm{pk}_s, \mathrm{sk}_s) \leftarrow \mathsf{KGen}(1^\lambda)$ *and* $(\mathrm{pk}_s, \mathrm{sk}_s) \leftarrow \mathsf{KGen}(1^\lambda)$*, and for any message* $m \in \{0,1\}^*$*, the following two random variables are the same distribution:*

$$\{\mathsf{Sign}(\mathrm{sk}_s, \mathrm{pk}_s, \mathrm{pk}_v, m)\} \text{ and } \{\mathsf{Sim}(\mathrm{sk}_v, \mathrm{pk}_s, \mathrm{pk}_v, m)\}.$$

Third, we review the definition of privacy of signer's identity. We refer to the definition of privacy of the signer's identity in [23].

Definition 6 (Privacy of Signer's Identity [23]). *Privacy of signer's identity (PSI) of* SDVS *is defined by the following privacy of signer's identity game* $\mathsf{Game}^{\mathsf{PSI}}_{\mathsf{SDVS},\mathsf{A}}(1^\lambda)$ *between a challenger* C *and an adversary* A*.*

– C *runs* $(\mathrm{pk}_{s_0}, \mathrm{sk}_{s_0}) \leftarrow \mathsf{KGen}(1^\lambda)$, $(\mathrm{pk}_{s_1}, \mathrm{sk}_{s_1}) \leftarrow \mathsf{KGen}(1^\lambda)$, $(\mathrm{pk}_v, \mathrm{sk}_v) \leftarrow \mathsf{KGen}(1^\lambda)$, *and gives* $(\mathrm{pk}_{s_0}, \mathrm{pk}_{s_1}, \mathrm{pk}_v)$ *to* A.
– A *is allowed to access oracles* $\mathcal{O}^{\mathsf{Sign}}(\cdot,\cdot)$ *and* $\mathcal{O}^{\mathsf{Verify}}(\cdot,\cdot,\cdot)$ *polynomially many times:*
  - $\mathcal{O}^{\mathsf{Sign}}(\cdot,\cdot)$: *For a query on* (d, m), *it runs* $\sigma \leftarrow \mathsf{Sign}(\mathrm{sk}_{s_d}, \mathrm{pk}_{s_d}, \mathrm{pk}_v, m)$ *and returns* σ *to* A.
  - $\mathcal{O}^{\mathsf{Verify}}(\cdot,\cdot,\cdot)$: *For a query on* (d, m, σ), *it computes* $\beta \leftarrow \mathsf{Ver}(\mathrm{sk}_v, \mathrm{pk}_{s_d}, \mathrm{pk}_v, m, \sigma)$ *and returns* β *to* A.
– A *sends a challenge* m^* *to* C.
– C *samples a bit* $b \xleftarrow{\$} \{0,1\}$, *and runs* $\sigma^* \leftarrow \mathsf{Sign}(\mathrm{sk}_{s_b}, \mathrm{pk}_{s_b}, \mathrm{pk}_v, m^*)$ *and gives* σ^* *to* A.
– A *is allowed to access oracles* $\mathcal{O}^{\mathsf{Sign}}(\cdot)$ *and* $\mathcal{O}^{\mathsf{Verify}^*}(\cdot,\cdot,\cdot)$ *polynomially many times where* $\mathcal{O}^{\mathsf{Verify}^*}$ *is the restricted oracle* $\mathcal{O}^{\mathsf{Verify}}$ *that queries* (d, m^*, σ^*) *not allowed for* $d \in \{0,1\}$.
– *Finally,* A *outputs a guess* b^*.
– A *wins the game if* $b^* = b$ *holds.*

The advantage of A *is defined as* $\mathsf{Adv}^{\mathsf{PSI}}_{\mathsf{SDVS},\mathsf{A}}(\lambda) := \left|\Pr[\mathsf{Game}^{\mathsf{PSI}}_{\mathsf{SDVS},\mathsf{A}}(1^\lambda) \Rightarrow 1] - \frac{1}{2}\right|$. *We say* SDVS *satisfies privacy of signer's identity if for any PPT adversary* A, $\mathsf{Adv}^{\mathsf{PSI}}_{\mathsf{SDVS},\mathsf{A}}(\lambda) = \mathsf{negl}(\lambda)$.

Finally, we review the definition of non-delegatability. This security is formalized by [24]. We refer to the equivalent non-delegatability definition in [16].

Definition 7 (Non-Delegatability [16]). *Let* $\kappa \in [0,1]$ *be the knowledge error and* F *a forger algorithm. Let* F_m *be* F *with* m *as its input, and oracle calls to* F_m *be counted as one step. We say* SDVS *satisfies non-delegatability with knowledge error* κ *if there exists a positive polynomial* $\mathsf{poly}(\lambda)$ *and a probabilistic oracle machine* Ext *such that for every PPT algorithm* F, Ext *satisfies the following condition:*

> *Let* ϵ *be the probability that* F *on the input* $m \in \{0,1\}^*$ *produces a valid signature on* m. *For any two pairs of keys* $(\mathrm{pk}_s, \mathrm{sk}_s) \leftarrow \mathsf{KGen}(1^\lambda)$ *and* $(\mathrm{pk}_s, \mathrm{sk}_s) \leftarrow \mathsf{KGen}(1^\lambda)$, *and for any message* $m \in \{0,1\}^*$, *if* $\epsilon > \kappa$, $\mathsf{Ext}^{\mathsf{F}_m}(m)$ *produces either* sk_s *or* sk_v *in expected polynomial time with probability at least* $(\epsilon - \kappa)/\mathsf{poly}(\lambda)$.

4 Our SDVS Scheme

In this section, we give our SDVS construction and prove security.

4.1 Our Construction

In this section, we describe our SDVS construction. Let $(p, N, \mathfrak{g}, E_0)$ be the public parameter specified as the underlying prime p, the order N of the ideal class group $\mathrm{Cl}_p \cong \mathbb{Z}_N$, a generator $\mathfrak{g}$, and the Montgomery type elliptic curves $E_0 : y^2 = x^3 + Ax^2 + x$. For a vector $\mathbf{r} \in \mathbb{Z}_N^\lambda$, $\mathbf{c} \in \{\pm 1\}$ and an elliptical curve $E \in \mathrm{Ell}_p$, $[\mathfrak{g}^{\mathbf{r}}] * E$ stands for a vector $([\mathfrak{g}^{\mathbf{r}[1]}] * E, \ldots, [\mathfrak{g}^{\mathbf{r}[\lambda]}] * E)$ and $[\mathfrak{g}^{\mathbf{r}}] * E^{\mathbf{c}}$ stands for a vector $([\mathfrak{g}^{\mathbf{r}[1]}] * E^{\mathbf{c}[1]}, \ldots, [\mathfrak{g}^{\mathbf{r}[\lambda]}] * E^{\mathbf{c}[\lambda]})$. We use $\odot$ to denote the component-wise multiplication of vectors in $\mathbb{R}$. Let $H : \{0,1\}^* \to \{0,1\}^\lambda$ be a hash function. Now, we give our SDVS scheme $\mathsf{SDVS}_{\mathsf{Ours}}$ as follows.

- $\mathsf{KGen}(1^\lambda)$:Sample $x \xleftarrow{\$} \mathbb{Z}_N$, set $(\mathrm{pk}, \mathrm{sk}) \leftarrow ([\mathfrak{g}^x] * E_0, x)$, and return $(\mathrm{pk}, \mathrm{sk})$.
- $\mathsf{Sign}(\mathrm{sk}_s = x_s, \mathrm{pk}_s, \mathrm{pk}_v, m)$: Sample $\mathbf{r}_s, \mathbf{z}_v \xleftarrow{\$} \mathbb{Z}_N^\lambda$, $\mathbf{c}_v \xleftarrow{\$} \{\pm 1\}^\lambda$. set $\mathbf{R}_s \leftarrow [\mathfrak{g}^{\mathbf{r}_s}] * E_0$, $\mathbf{R}_v \leftarrow [\mathfrak{g}^{\mathbf{z}_v}] * \mathrm{pk}_v^{\mathbf{c}_v}$, $K \leftarrow [\mathfrak{g}^{x_s}] * \mathrm{pk}_v$, $\mathbf{c} \leftarrow H(m, K, \mathbf{R}_s, \mathbf{R}_v) \in \{\pm 1\}^\lambda$, $\mathbf{c}_s \leftarrow \mathbf{c} \odot \mathbf{c}_v$, $\mathbf{z}_s \leftarrow \mathbf{r}_s - x_s \cdot \mathbf{c}_s$, and return $\sigma := (\mathbf{c}_s, \mathbf{z}_s, \mathbf{c}_v, \mathbf{z}_v)$
- $\mathsf{Ver}(\mathrm{sk}_v = x_v, \mathrm{pk}_s, \mathrm{pk}_v, m, \sigma = (\mathbf{c}_s, \mathbf{z}_s, \mathbf{c}_v, \mathbf{z}_v))$: Set $\mathbf{R}_s \leftarrow [\mathfrak{g}^{\mathbf{z}_s}] * \mathrm{pk}_s^{\mathbf{c}_s}$, $\mathbf{R}_v \leftarrow [\mathfrak{g}^{\mathbf{z}_v}] * \mathrm{pk}_v^{\mathbf{c}_v}$, $K \leftarrow [\mathfrak{g}^{x_v}] * \mathrm{pk}_s$. If $\mathbf{c}_s \odot \mathbf{c}_v = H(m, K, \mathbf{R}_s, \mathbf{R}_v)$, return 1. Otherwise, return 0.

Correctness: Let $(\mathrm{pk}_s = [\mathfrak{g}^{x_s}] * E_0, \mathrm{sk}_s = x_s) \leftarrow \mathsf{KGen}(1^\lambda)$, $(\mathrm{pk}_v = [\mathfrak{g}^{x_v}] * E_0, \mathrm{sk}_v = x_v) \leftarrow \mathsf{KGen}(1^\lambda)$, and $\sigma = (\mathbf{c}_s, \mathbf{z}_s, \mathbf{c}_v, \mathbf{z}_v) \leftarrow \mathsf{Sign}(\mathrm{sk}_s = x_s, \mathrm{pk}_s, \mathrm{pk}_v, m)$. Then $\mathbf{c} = H(m, K = [\mathfrak{g}^{x_s}] * \mathrm{pk}_v, \mathbf{R}_s = [\mathfrak{g}^{\mathbf{r}_s}] * E_0, \mathbf{R}_v = [\mathfrak{g}^{\mathbf{z}_v}] * \mathrm{pk}_v^{\mathbf{c}_v})$ holds. Since $K = [\mathfrak{g}^{x_s}] * \mathrm{pk}_v = [\mathfrak{g}^{x_v}] * \mathrm{pk}_s$, $\mathbf{R}_s = [\mathfrak{g}^{\mathbf{r}_s}] * E_0 = [\mathfrak{g}^{\mathbf{z}_s - x_s \cdot \mathbf{c}_s}] = [\mathfrak{g}^{\mathbf{z}_s}] * \mathrm{pk}_s^{\mathbf{c}_s}$, and $\mathbf{c} = \mathbf{c}_s \odot \mathbf{c}_v$ hold, $\mathbf{c}_s \odot \mathbf{c}_v = \mathbf{c} = H(m, [\mathfrak{g}^{x_v}] * \mathrm{pk}_s, [\mathfrak{g}^{\mathbf{z}_s}] * \mathrm{pk}_s^{\mathbf{c}_s}, [\mathfrak{g}^{\mathbf{z}_v}] * \mathrm{pk}_v^{\mathbf{c}_v})$ holds. Thus, we see that $\mathsf{SDVS}_{\mathsf{Ours}}$ satisfies correctness.

4.2 Non-Transferability Analysis

In this section, we prove non-transferability for our SDVS.

Theorem 1. $\mathsf{SDVS}_{\mathsf{Ours}}$ *satisfies perfectly non-transferability.*

Proof. We give the simulation algorithm Sim as follows.

- $\mathsf{Sim}(\mathrm{sk}_v = x_v, \mathrm{pk}_s, \mathrm{pk}_v, m)$: Sample $\mathbf{r}_v, \mathbf{z}_s \xleftarrow{\$} \mathbb{Z}_N^\lambda$, $\mathbf{c}_s \xleftarrow{\$} \{\pm 1\}^\lambda$, set $\mathbf{R}_v \leftarrow [\mathfrak{g}^{\mathbf{r}_v}] * E_0$, $\mathbf{R}_s \leftarrow [\mathfrak{g}^{\mathbf{z}_s}] * \mathrm{pk}_s^{\mathbf{c}_s}$, $K \leftarrow [\mathfrak{g}^{x_v}] * \mathrm{pk}_s$, $\mathbf{c} \leftarrow H(m, K, \mathbf{R}_s, \mathbf{R}_v) \in \{\pm 1\}^\lambda$, $\mathbf{c}_v \leftarrow \mathbf{c} \odot \mathbf{c}_s$, $\mathbf{z}_v \leftarrow \mathbf{r}_v - x_v \cdot \mathbf{c}_v$, and return $\sigma = (\mathbf{c}_s, \mathbf{z}_s, \mathbf{c}_v, \mathbf{z}_v)$.

Now, we discuss the distributions $\{\mathsf{Sign}(\mathrm{sk}_s, \mathrm{pk}_s, \mathrm{pk}_v, m)\}$ and $\{\mathsf{Sim}(\mathrm{sk}_v, \mathrm{pk}_s, \mathrm{pk}_v, m)\}$. Our signing algorithm is based on the Σ-protocol for the OR relation [19,20]. Their Σ-protocol proves the knowledge of a secret key either sk_s or sk_v. The signing algorithm Sign is obtained by transforming this Σ-protocol with running sk_s into a non-interactive form. The simulation algorithm Sim is obtained by transforming this Σ-protocol with running sk_v into a non-interactive form. Since Σ-protocol for the OR relation [19,20] satisfies perfect witness indistinguishability (WI), we see that $\{\mathsf{Sign}(\mathrm{sk}_s, \mathrm{pk}_s, \mathrm{pk}_v, m)\}$ and $\{\mathsf{Sim}(\mathrm{sk}_v, \mathrm{pk}_s, \mathrm{pk}_v, m)\}$ are identical.

(Theorem 1) □

4.3 Strong Unforgeability Analysis

In this section, we prove strong unforgeability for our SDVS.

Theorem 2. *If the GAIP assumption holds on* $(\mathrm{Cl}_p, \mathrm{Ell}_p)$*,* $\mathsf{SDVS}_{\mathsf{Ours}}$ *satisfies strong unforgeability in the ROM.*

Proof. We prove that if a PPT adversary A to break strong unforgeability, A can be used to break the GAIP assumption by giving a reduction R. Our reduction R rewinds the running of A by using the forking algorithm Fork in Lemma 2. We construct R into two steps. First, we construct an algorithm W for the forking algorithm Fork. Then we construct the reduction R that runs $\mathsf{Fork}^{\mathsf{W}}$.

Construction of W^{A}. $\mathsf{W}(x = (E_0, E'), \mathbf{c}_1, \ldots, \mathbf{c}_{Q_{\mathsf{Sign}}+Q_{\mathsf{Ver}}+Q_H})$ simulates the view of A without the signer's secret key sk_s or the verifier's secret key sk_v where Q_{Sign}, Q_{Ver} and Q_H the total number of queries from A to $\mathcal{O}^{\mathsf{Sign}}$, $\mathcal{O}^{\mathsf{Ver}}$, and $\mathcal{O}^H$, respectively.

- **First procedure of W:** Given an input $x = (E_0, E') \in \mathrm{Ell}_p^2$, $\mathbf{c}_1, \ldots, \mathbf{c}_{Q_{\mathsf{Sign}}+Q_{\mathsf{Ver}}+Q_H} \in (\{\pm 1\}^\lambda)^{Q_{\mathsf{Sign}}+Q_{\mathsf{Ver}}+Q_H})$, W initializes a hash table $\mathbb{T} \leftarrow \{\}$, a signature list L^{Sign} and a counter $\mathit{ctr} \leftarrow 0$. W samples $\alpha \xleftarrow{\$} \{0,1\}$. If $\alpha = 0$, W sets $\mathrm{pk}_s \leftarrow E'$, $(\mathrm{pk}_v, \mathrm{sk}_v) \leftarrow \mathsf{KGen}(1^\lambda)$. If $\alpha = 1$, W sets $\mathrm{pk}_v \leftarrow E'$, $(\mathrm{pk}_s, \mathrm{sk}_s) \leftarrow \mathsf{KGen}(1^\lambda)$. (If $\alpha = 0$, W simulates the game without sk_s. If $\alpha = 0$, W simulates the game without sk_v.) W gives an input $(\mathrm{pk}_s, \mathrm{pk}_v)$ to A.
- **Simulation of $\mathcal{O}^H$:** For a query $(m, K, \mathbf{R}_s, \mathbf{R}_v)$, if $\mathbb{T}[m, K, \mathbf{R}_s, \mathbf{R}_v] = \mathbf{c}$ for some $\mathbf{c}$, W returns $\mathbf{c}$. Otherwise, W sets $\mathit{ctr} \leftarrow \mathit{ctr} + 1$, defines $\mathbb{T}[m, K, \mathbf{R}_s, \mathbf{R}_v] := \mathbf{c}_{\mathit{ctr}}$, and returns $\mathbf{c}_{\mathit{ctr}}$.
- **Simulation of $\mathcal{O}^{\mathsf{Sign}}$:** For a query m, if $\alpha = 0$, W sets $\sigma \leftarrow \mathsf{Sim}(\mathrm{sk}_v, \mathrm{pk}_s, \mathrm{pk}_v, m)$ where Sim is the algorithm in the proof of Theorem 1. If $\alpha = 1$, W sets $\sigma \leftarrow \mathsf{Sign}(\mathrm{sk}_s, \mathrm{pk}_s, \mathrm{pk}_v, m)$. W updates $L^{\mathsf{Sign}} \leftarrow L^{\mathsf{Sign}} \cup \{(m, \sigma)\}$ and returns σ.
- **Simulation of $\mathcal{O}^{\mathsf{Ver}}$:** For a query (m, σ), if $\alpha = 0$, W returns $b = \mathsf{Ver}(\mathrm{sk}_v, \mathrm{pk}_s, \mathrm{pk}_v, m, \sigma)$. If $\alpha = 1$, W returns $b = \widetilde{\mathsf{Ver}}(\mathrm{sk}_s, \mathrm{pk}_s, \mathrm{pk}_v, m, \sigma)$ where $\widetilde{\mathsf{Ver}}$ is the modified algorithm of Ver by changing the calculation of K from $K \leftarrow [\mathfrak{g}^{x_v}] * \mathrm{pk}_s$ to $K \leftarrow [\mathfrak{g}^{x_s}] * \mathrm{pk}_v$.
- **Final procedure of W:** W receives the forgery $(m^*, \sigma^* = (\mathbf{c}_s^*, \mathbf{z}_s^*, \mathbf{c}_v^*, \mathbf{z}_v^*))$. Then W computes $\mathbf{R}_s^* \leftarrow [\mathfrak{g}^{\mathbf{z}_s^*}] * \mathrm{pk}_s^{\mathbf{c}_s^*}$, $\mathbf{R}_v^* \leftarrow [\mathfrak{g}^{\mathbf{z}_v^*}] * \mathrm{pk}_v^{\mathbf{c}_v^*}$. If $\alpha = 0$, W sets $K^* \leftarrow [\mathfrak{g}^{x_s}] * \mathrm{pk}_v$. If $\alpha = 1$, W sets $K^* \leftarrow [\mathfrak{g}^{x_v}] * \mathrm{pk}_s$. If there is no entry $((m^*, K^*, \mathbf{R}_s^*, \mathbf{R}_v^*), \mathbf{c})$ for any $\mathbf{c}$, then W aborts. If $(m^*, \sigma^*) \in L^{\mathsf{Sign}}$, W returns $(0, \perp)$. If $\alpha = 0 \wedge \mathsf{Ver}(\mathrm{sk}_v, \mathrm{pk}_s, \mathrm{pk}_v, m^*, \sigma^*) \neq 1$, W returns $(j, \mathrm{out}) \leftarrow (0, \perp)$. If $\alpha = 1 \wedge \mathsf{Ver}(\mathrm{sk}_s, \mathrm{pk}_s, \mathrm{pk}_v, m^*, \sigma^*) \neq 1$, W returns $(j, \mathrm{out}) \leftarrow (0, \perp)$. Otherwise, W returns $(j, \mathrm{out}) \leftarrow (\mathit{ctr}^*, (m^*, \sigma^*, \alpha))$ where ctr^* is the value of ctr at the moment when $\mathbb{T}[m^*, K^*, \mathbf{R}_s^*, \mathbf{R}_v^*] = c_{\mathit{ctr}}$ is defined and α is the bit sampled in the first procedure of W.

Construction of R. The reduction algorithm $\mathsf{R}(E_0, E')$ runs $\mathsf{Fork}^{\mathsf{W}}$ and extracts the GAIP solution. We give xR as follows.

- Given a GAIP instance $(E_0, E') \leftarrow \mathrm{Ell}_p$ as input, R runs $(d, \mathrm{out}, \widetilde{\mathrm{out}}) \leftarrow \mathsf{Fork}^{\mathsf{W}}(x = (E_0, E'))$. If $d = 0$, R aborts.
- R parses $(m^*, \sigma^* = (\mathbf{c}_s^*, \mathbf{z}_s^*, \mathbf{c}_v^*, \mathbf{z}_v^*), \alpha^*) \leftarrow \mathrm{out}, (\widetilde{m}^*, \widetilde{\sigma}^* = (\widetilde{\mathbf{c}}_s^*, \widetilde{\mathbf{z}}_s^*, \widetilde{\mathbf{c}}_v^*, \widetilde{\mathbf{z}}_v^*), \widetilde{\alpha}^*) \leftarrow \widetilde{\mathrm{out}}$.
- If $\alpha = 0 \land \mathbf{c}_s^* \neq \widetilde{\mathbf{c}}_s^*$, R finds an index i^* such that $\mathbf{c}_s^*[i^*] \neq \widetilde{\mathbf{c}}_s^*[i^*]$. Then R outputs a solution $\left[\mathfrak{g}^{\frac{\widetilde{\mathbf{z}}_s^*[i^*] - \mathbf{z}_s^*[i^*]}{\mathbf{c}_s^*[i^*] - \widetilde{\mathbf{c}}_s^*[i^*]}}\right]$. If $\alpha = 0 \land \mathbf{c}_s^* = \widetilde{\mathbf{c}}_s^*$, R aborts.
- If $\alpha = 1 \land \mathbf{c}_v^* \neq \widetilde{\mathbf{c}}_v^*$, R finds an index i^* such that $\mathbf{c}_v^*[i^*] \neq \widetilde{\mathbf{c}}_v^*[i^*]$. Then R outputs a solution $\left[\mathfrak{g}^{\frac{\widetilde{\mathbf{z}}_v^*[i^*] - \mathbf{z}_v^*[i^*]}{\mathbf{c}_v^*[i^*] - \widetilde{\mathbf{c}}_v^*[i^*]}}\right]$. If $\alpha = 1 \land \mathbf{c}_v^* = \widetilde{\mathbf{c}}_v^*$, R aborts.

Analysis of W. First, we confirm that perfectly simulates W the view of A. In the case of $\alpha = 0$, the difference from the original strong unforgeability game is the simulation of $\mathcal{O}^{\mathsf{Sign}}$. Since $\mathsf{Sim}(\mathrm{sk}_v, \mathrm{pk}_s, \mathrm{pk}_v, m)$ perfectly simulates $\mathsf{Sign}(\mathrm{sk}_s, \mathrm{pk}_s, \mathrm{pk}_v, m)$, W perfectly simulates the view of A. In the case of $\alpha = 1$, the difference from the original strong unforgeability game is the simulation of $\mathcal{O}^{\mathsf{Ver}}$. The verification procedure differs in the method of computing K. Since $[\mathfrak{g}^{x_v}] * \mathrm{pk}_s = [\mathfrak{g}^{x_s}] * \mathrm{pk}_v$ holds, we see that W perfectly simulates the view of A.

Next, we consider the probability that W outputs (j, out) such that $j \geq 1$. Let $\mathsf{Adv}^{\mathsf{SUF}}_{\mathsf{SDVS}_{\mathsf{Ours}},\mathsf{A}}$ be the advantage of the strong unforgeability game for our scheme $\mathsf{SDVS}_{\mathsf{Ours}}$. Since without the hash query on $(m^*, K^*, \mathbf{R}_s^*, \mathbf{R}_v^*)$, the probability that the success probability of $\mathsf{Ver}(\mathrm{sk}_v, \mathrm{pk}_s, \mathrm{pk}_v, m^*, \sigma^*) \neq 1$ is $\frac{1}{\#\{0,1\}^\lambda}$. Thus, we have $\mathsf{acc}(\lambda) = \mathsf{Adv}^{\mathsf{SUF}}_{\mathsf{SDVS}_{\mathsf{Ours}},\mathsf{A}} - \frac{1}{2^\lambda}$.

Analysis of R. First, we confirm that $d = 1 \land \alpha = 0 \land \mathbf{c}_s^* \neq \widetilde{\mathbf{c}}_s^*$ or $d = 1 \land \alpha = 1 \land \mathbf{c}_v^* \neq \widetilde{\mathbf{c}}_v^*$, R extracts a solution for the GAIP instance. Before considering each case, we note the important fact that if $d = 1$, $m^* = \widetilde{m}^*$, $[\mathfrak{g}^{\mathbf{z}_s^*}] * \mathrm{pk}_s^{\mathbf{c}_s^*} = \mathbf{R}_s^* = \widetilde{\mathbf{R}}_s^* = [\mathfrak{g}^{\widetilde{\mathbf{z}}_s^*}] * \mathrm{pk}_s^{\widetilde{\mathbf{c}}_s^*}$, and $[\mathfrak{g}^{\mathbf{z}_v^*}] * \mathrm{pk}_v^{\mathbf{c}_v^*} = \mathbf{R}_v^* = \widetilde{\mathbf{R}}_v^* = [\mathfrak{g}^{\widetilde{\mathbf{z}}_v^*}] * \mathrm{pk}_v^{\widetilde{\mathbf{c}}_v^*}$ hold. This fact comes from the fact that the forking algorithm Fork fixes the random coin ρ for W and runs W same input x, $\mathbf{c}_1, \ldots, \mathbf{c}_{j-1}$, and ρ. This implies that during the first and second executions of W, the behavior of A remains the same up to the point where the hash query is defined using c_j.

In the case of $d = 1 \land \alpha = 0 \land \mathbf{c}_s^* \neq \widetilde{\mathbf{c}}_s^*$, there is i^* such that $\mathbf{c}_s^*[i^*] \neq \widetilde{\mathbf{c}}_s^*[i^*]$. Since $[\mathfrak{g}^{\mathbf{z}_s^*}] * \mathrm{pk}_s^{\mathbf{c}_s^*} = [\mathfrak{g}^{\widetilde{\mathbf{z}}_s^*}] * \mathrm{pk}_s^{\widetilde{\mathbf{c}}_s^*}$ holds, we have $[\mathfrak{g}^{\mathbf{z}_s^*[i^*]}] * \mathrm{pk}_s^{\mathbf{c}_s^*[i^*]} = [\mathfrak{g}^{\widetilde{\mathbf{z}}_s^*[i^*]}] * \mathrm{pk}_s^{\widetilde{\mathbf{c}}_s^*[i^*]}$. Let $E' = [\mathfrak{g}^x] * E_0$. Then $\mathrm{pk}_s = E' = [\mathfrak{g}^x] * E_0$ holds. From these facts, we have $[\mathfrak{g}^{\mathbf{z}_s^*[i^*] + x\mathbf{c}_s^*[i^*]}] * E_0 = [\mathfrak{g}^{\widetilde{\mathbf{z}}_s^*[i^*] + x\widetilde{\mathbf{c}}_s^*[i^*]}] * E_0$. Thus, $E = \left[\mathfrak{g}^{\frac{\widetilde{\mathbf{z}}_s^*[i^*] - \mathbf{z}_s^*[i^*]}{\mathbf{c}_s^*[i^*] - \widetilde{\mathbf{c}}_s^*[i^*]}}\right] * E_0$ holds. The case of $d = 1 \land \alpha = 1 \land \mathbf{c}_v^* \neq \widetilde{\mathbf{c}}_v^*$ can be analyzed in the same way as discussed above.

Next, we analyze the probability that R outputs a GAIP solution. If $d = 0$, R failed to extract a solution. We consider the condition where $d = 1$ holds. In the case of $\mathbf{c}_s^* = \widetilde{\mathbf{c}}_s^*$, if $\alpha = 0$, R fails to extract a solution and $\alpha = 1$, R can extract a solution. From this fact, with probability one-half, R can extract a solution. The case of $\mathbf{c}_v^* = \widetilde{\mathbf{c}}_v^*$ can be analyzed in the same way as discussed above.

By the forking lemma (Lemma 2), we have

$$\begin{aligned}\mathsf{Adv}_{\mathsf{R}}^{\mathsf{GAIP}}(\lambda) &= \frac{1}{2} \cdot \mathsf{frk}(\lambda) \geq \frac{1}{2} \cdot \mathsf{acc}(\lambda) \cdot \left(\frac{\mathsf{acc}(\lambda)}{Q} - \frac{1}{2^\lambda} \right) \\ &\geq \frac{1}{2Q} \cdot \left(\mathsf{Adv}_{\mathsf{SDVS}_{\mathsf{Ours}},\mathsf{A}}^{\mathsf{SUF}} - \frac{1}{2^\lambda} \right) \cdot \left(\mathsf{Adv}_{\mathsf{SDVS}_{\mathsf{Ours}},\mathsf{A}}^{\mathsf{SUF}} - \frac{Q+1}{2^\lambda} \right)\end{aligned}$$

where $Q = Q_{\mathsf{Sign}} + Q_{\mathsf{Ver}} + Q_H$.

(Theorem 2) □

4.4 Privacy of Signer's Identity Analysis

In this section, we prove privacy of signer's identity for our SDVS.

Theorem 3. *If $\mathsf{SDVS}_{\mathsf{Ours}}$ satisfies strong unforgeability in the ROM and the CSI-GDH assumption holds on $(\mathrm{Cl}_p, \mathrm{Ell}_p)$, $\mathsf{SDVS}_{\mathsf{Ours}}$ satisfies privacy of signer's identity in the ROM.*

Proof. Let A be a PPT adversary for $\mathsf{Game}_{\mathsf{SDVS}_{\mathsf{Ours}}}^{\mathsf{PSI}}$. We introduce the sequence of games $\mathsf{Game}_{0,\mathsf{A}}, \ldots, \mathsf{Game}_{6,\mathsf{A}}$ as follows.

- $\mathsf{Game}_{0,\mathsf{A}}$: The original privacy of signer's identity security game $\mathsf{Game}_{\mathsf{SDVS}_{\mathsf{Ours}},\mathsf{A}}^{\mathsf{PSI}}$ in the ROM with some modification. We introduce a signing list L^{Sign}. For a query on input (d, m) to the signing oracle $\mathcal{O}^{\mathsf{Sign}}$, $\mathcal{O}^{\mathsf{Sign}}$ outputs a signature and records a tuple (d, m, σ) to L^{Sign}. We also introduce the hash table $\mathbb{T}$. These modifications do not change the view of an adversary A.
- $\mathsf{Game}_{1,\mathsf{A}}$: This game is identical to $\mathsf{Game}_{0,\mathsf{A}}$ except that we modify the generation of the signature σ^* for the challenge and the verification oracle $\mathcal{O}^{\mathsf{Ver}}$. For a challenge query on m^*, sample $b \xleftarrow{\$} \{0,1\}$ and query (b, m^*) the signing oracle $\mathcal{O}^{\mathsf{Sign}}$, obtain a signature σ^* and return σ^*.
  For a query on $(0, m, \sigma)$, search a tuple $(0, m, \sigma)$ from L^{Sign}. If there is tuple in L^{Sign} return 1 and 0 otherwise. For a query on $(1, m, \sigma)$, return $b = \widetilde{\mathsf{Ver}}(\mathrm{sk}_{s_1}, \mathrm{pk}_{s_1}, \mathrm{pk}_v, m, \sigma)$ where $\widetilde{\mathsf{Ver}}$ is the modified algorithm obtained from $\mathsf{Ver}(\mathrm{sk}_v, \mathrm{pk}_{s_1}, \mathrm{pk}_v, m, \sigma)$ by modifying the computation of K from $K = [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_1}$ to $K = [\mathfrak{g}^{x_{s_1}}] * \mathrm{pk}_v$.
- $\mathsf{Game}_{2,\mathsf{A}}$: This game is identical to $\mathsf{Game}_{1,\mathsf{A}}$ except that we modify the signing oracle $\mathcal{O}^{\mathsf{Sign}}$. For a query on $(0, m, \sigma)$, samples $\mathbf{c}_s, \mathbf{c}_v \xleftarrow{\$} \{\pm 1\}^\lambda$, $\mathbf{z}_s, \mathbf{z}_v \xleftarrow{\$} \mathbb{Z}_N^\lambda$, set $K \leftarrow [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_0}$, $\mathbf{R}_s \leftarrow [\mathfrak{g}^{\mathbf{z}_s}] * \mathrm{pk}_s^{\mathbf{c}_s}$, $\mathbf{R}_v \leftarrow [\mathfrak{g}^{\mathbf{z}_v}] * \mathrm{pk}_v^{\mathbf{c}_v}$, $\mathbf{c} \leftarrow \mathbf{c}_s \odot \mathbf{c}_v$. If there is $\mathbb{T}[m, K, \mathbf{R}_s, \mathbf{R}_v] \neq \perp$, then abort. Otherwise, update $\mathbb{T}[m, K, \mathbf{R}_s, \mathbf{R}_v] = \mathbf{c}$ and return $\sigma \leftarrow (\mathbf{c}_s, \mathbf{c}_v, \mathbf{z}_s, \mathbf{z}_v)$.
- $\mathsf{Game}_{3,\mathsf{A}}$: This game is identical to $\mathsf{Game}_{2,\mathsf{A}}$ except that we modify the hash oracle $\mathcal{O}^H$ and the signing oracle $\mathcal{O}^{\mathsf{Sign}}$. We modify the $\mathcal{O}^H$ as follows. For a query on $(m, K, \mathbf{R}_s, \mathbf{R}_v)$, check $K = [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_0}$ by using the CSI-DDH

oracle. If $K = [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_0}$ holds, then abort. Otherwise, behave the same as an ordinary random oracle and output a hash value.

We modify the $\mathcal{O}^{\mathsf{Sign}}$ as follows. For a query on $(0, m, \sigma)$, samples $\mathbf{c}_s, \mathbf{c}_v \xleftarrow{\$} \{\pm 1\}^\lambda$, $\mathbf{z}_s, \mathbf{z}_v \xleftarrow{\$} \mathbb{Z}_N^\lambda$, set $K \leftarrow \bot_0$, $\mathbf{R}_s \leftarrow [\mathfrak{g}^{\mathbf{z}_s}] * \mathrm{pk}_s^{\mathbf{c}_s}$, $\mathbf{R}_v \leftarrow [\mathfrak{g}^{\mathbf{z}_v^*}] * \mathrm{pk}_v^{\mathbf{c}_v^*}$, $\mathbf{c} \leftarrow \mathbf{c}_s \odot \mathbf{c}_v$. If there is $\mathbb{T}[m, \bot_0, \mathbf{R}_s, \mathbf{R}_v] \neq \bot$, then abort. Otherwise, update $\mathbb{T}[m, \bot_0, \mathbf{R}_s, \mathbf{R}_v] = \mathbf{c}$ and return $\sigma \leftarrow (\mathbf{c}_s, \mathbf{c}_v, \mathbf{z}_s, \mathbf{z}_v)$.

– $\mathsf{Game}_{4,\mathsf{A}}$: This game is identical to $\mathsf{Game}_{3,\mathsf{A}}$ except that we modify the verification oracle $\mathcal{O}^{\mathsf{Ver}}$. For a query on $(1, m, \sigma)$, search a tuple $(1, m, \sigma)$ from L^{Sign}. If there is tuple in L^{Sign} return 1 and 0 otherwise.
– $\mathsf{Game}_{5,\mathsf{A}}$: This game is identical to $\mathsf{Game}_{4,\mathsf{A}}$ except that we modify the signing oracle $\mathcal{O}^{\mathsf{Sign}}$. For a query on $(1, m, \sigma)$, samples $\mathbf{c}_s, \mathbf{c}_v \xleftarrow{\$} \{\pm 1\}^\lambda$, $\mathbf{z}_s, \mathbf{z}_v \xleftarrow{\$} \mathbb{Z}_N^\lambda$, set $K \leftarrow [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_1}$, $\mathbf{R}_s \leftarrow [\mathfrak{g}^{\mathbf{z}_s}] * \mathrm{pk}_s^{\mathbf{c}_s}$, $\mathbf{R}_v \leftarrow [\mathfrak{g}^{\mathbf{z}_v}] * \mathrm{pk}_v^{\mathbf{c}_v}$, $\mathbf{c} \leftarrow \mathbf{c}_s \odot \mathbf{c}_v$. If there is $\mathbb{T}[m, K, \mathbf{R}_s, \mathbf{R}_v] \neq \bot$, then abort. Otherwise, update $\mathbb{T}[m, K, \mathbf{R}_s, \mathbf{R}_v] = \mathbf{c}$ and return $\sigma \leftarrow (\mathbf{c}_s, \mathbf{c}_v, \mathbf{z}_s, \mathbf{z}_v)$.
– $\mathsf{Game}_{6,\mathsf{A}}$: This game is identical to $\mathsf{Game}_{5,\mathsf{A}}$ except that we modify the hash oracle $\mathcal{O}^H$ and the signing oracle $\mathcal{O}^{\mathsf{Sign}}$. We modify the $\mathcal{O}^H$ as follows. For a query on $(m, K, \mathbf{R}_s, \mathbf{R}_v)$, check $K = [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_0}$ by using the CSI-DDH oracle. If $K = [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_0}$ holds, then abort. Also, check $K = [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_1}$ by using the CSI-DDH oracle. If $K = [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_1}$ holds, then abort. Otherwise, behave the same as an ordinary random oracle and output a hash value.

We modify the $\mathcal{O}^{\mathsf{Sign}}$ as follows. For a query on $(1, m, \sigma)$, samples $\mathbf{c}_s, \mathbf{c}_v \xleftarrow{\$} \{\pm 1\}^\lambda$, $\mathbf{z}_s, \mathbf{z}_v \xleftarrow{\$} \mathbb{Z}_N^\lambda$, set $K \leftarrow \bot_1$, $\mathbf{R}_s \leftarrow [\mathfrak{g}^{\mathbf{z}_s}] * \mathrm{pk}_s^{\mathbf{c}_s}$, $\mathbf{R}_v \leftarrow [\mathfrak{g}^{\mathbf{z}_v}] * \mathrm{pk}_v^{\mathbf{c}_v}$, $\mathbf{c} \leftarrow \mathbf{c}_s \odot \mathbf{c}_v$. If there is $\mathbb{T}[m, \bot_1, \mathbf{R}_s, \mathbf{R}_v] \neq \bot$, then abort. Otherwise, update $\mathbb{T}[m, \bot_1, \mathbf{R}_s, \mathbf{R}_v] = \mathbf{c}$ and return $\sigma \leftarrow (\mathbf{c}_s, \mathbf{c}_v, \mathbf{z}_s, \mathbf{z}_v)$.

For a sequence of games $\mathsf{Game}_{0,\mathsf{A}}, \ldots, \mathsf{Game}_{6,\mathsf{A}}$, the following lemmas hold.

Lemma 3. *If the* $\mathsf{SDVS}_{\mathsf{Ours}}$ *satisfies strong unforgeability,*

$$|\Pr[\mathsf{Game}_{0,\mathsf{A}} \Rightarrow 1] - \Pr[\mathsf{Game}_{1,\mathsf{A}} \Rightarrow 1| = \mathsf{negl}(\lambda).$$

Proof. Let $\mathbf{BAD}_{V0}$ be the event that A queries $(0, m, \sigma)$ to $\mathcal{O}^{\mathsf{Ver}}$ such that $(0, m, \sigma) \notin L^{\mathsf{Sign}}$ and $\mathsf{Ver}(\mathrm{sk}_v, pk_{s0}, pk_v, m, \sigma) = 1$. If $\mathbf{BAD}_{V0}$ does not occur, the views of A are identical in $\mathsf{Game}_{0,\mathsf{A}}$ and $\mathsf{Game}_{1,\mathsf{A}}$. The probability that $\mathbf{BAD}_{V0}$ occurs is bounded by the following reduction R to strong unforgeability security of $\mathsf{SDVS}_{\mathsf{Ours}}$. The reduction R is given as follows.

– **First procedure of** R**:** Given an input $(\mathrm{pk}_s, \mathrm{pk}_v)$ from the challenger C of the strong unforgeability game, R sets $\mathrm{pk}_{s_0} = \mathrm{pk}_s$, runs $(\mathrm{pk}_{s_1}, \mathrm{sk}_{s_1}) \leftarrow \mathsf{KGen}(1^\lambda)$ and gives $(\mathrm{pk}_{s_0}, \mathrm{pk}_{s_1}, \mathrm{pk}_{s_v})$ to A. Then R simulates the view of A without sk_{s_0} and sk_v until $\mathbf{BAD}_{V0}$ occurs.
– **Simulation of** $\mathcal{O}^H$**:** For a query on $(m, K, \mathbf{R}_s, \mathbf{R}_v)$, R queries $(m, K, \mathbf{R}_s, \mathbf{R}_v)$ to $\mathcal{O}^H$ in the strong unforgeability game and obtains a hash value $\mathbf{c}$. Then R returns $\mathbf{c}$.
– **Simulation of** $\mathcal{O}^{\mathsf{Sign}}$**:** For a query on (d, m), if $d = 0$, R queries m to $\mathcal{O}^{\mathsf{Sign}}$ in the strong unforgeability game and obtains a signature σ. If $d = 1$, runs $\sigma \leftarrow \mathsf{Sign}(\mathrm{sk}_{s_1}, \mathrm{pk}_{s_1}, \mathrm{pk}_v, m)$. R records (d, m, σ) to L^{Sign} and returns σ.

- **Simulation of $\mathcal{O}^{\mathsf{Ver}}$:** For a query on (d, m, σ), if $d = 1$, R returns $b = \widetilde{\mathsf{Ver}}(\mathrm{sk}_{s_1}, \mathrm{pk}_{s_1}, \mathrm{pk}_v, m, \sigma)$ where $\widetilde{\mathsf{Ver}}$ is the algorithm in Game_1. In the case of $d = 0$, if there is an entry $(0, m, \sigma)$ in L^{Sign}, return 1. Otherwise, R queries (m, σ) to $\mathcal{O}^{\mathsf{Sign}}$ in the strong unforgeability game and obtains a bit β. If $\beta = 0$, return 0 to A. If $\beta = 1$ (i.e., $\mathbf{BAD}_{V0}$ occurs), W abort the simulation for A and returns a forgery (m, σ) to C.

If $\mathbf{BAD}_{V0}$ occurs, R outputs a valid forgery for the strong unforgeability game. Thus, we obtain the following bound:

$$|\Pr[\mathsf{Game}_{0,\mathsf{A}} \Rightarrow 1] - \Pr[\mathsf{Game}_{1,\mathsf{A}} \Rightarrow 1| \leq \mathsf{Adv}^{\mathsf{SUF}}_{\mathsf{SDVS}_{\mathsf{Ours}},\mathsf{R}} = \mathsf{negl}(\lambda).$$

(Lemma 3) □

Lemma 4.

$$|\Pr[\mathsf{Game}_{1,\mathsf{A}} \Rightarrow 1] - \Pr[\mathsf{Game}_{2,\mathsf{A}} \Rightarrow 1| = \mathsf{negl}(\lambda).$$

Proof. Let Q_{Sign}, Q_{Ver} and Q_H be the total number of queries from A to $\mathcal{O}^{\mathsf{Sign}}$, $\mathcal{O}^{\mathsf{Ver}}$, $\mathcal{O}^H$, respectively. Let $\mathbf{BAD}_{S0}$ be the event that the signing oracle aborts when $(0, m)$ is queried by A. If $\mathbf{BAD}_{S0}$ does not occur, the views of A are identical in $\mathsf{Game}_{1,\mathsf{A}}$ and $\mathsf{Game}_{2,\mathsf{A}}$.

We bound the probability that $\mathbf{BAD}_{S0}$ occurs. For each signing query $(0, m)$, the probability that $\mathcal{O}^{\mathsf{Sign}}$ aborts is at most $\frac{Q_{\mathsf{Sign}}+Q_{\mathsf{Ver}}+Q_H+1}{\#\mathrm{Cl}_p^\lambda}$. By taking the union bound, we have

$$\begin{aligned}|\Pr[\mathsf{Game}_{1,\mathsf{A}} \Rightarrow 1] - \Pr[\mathsf{Game}_{2,\mathsf{A}} \Rightarrow 1| &\leq (Q_{\mathsf{Sign}} + 1) \cdot \frac{Q_{\mathsf{Sign}} + Q_{\mathsf{Ver}} + Q_H + 1}{N^\lambda} \\ &= \mathsf{negl}(\lambda).\end{aligned}$$

(Lemma 4) □

Lemma 5. *If the CSI-GDH assumption holds,*

$$|\Pr[\mathsf{Game}_{2,\mathsf{A}} \Rightarrow 1] - \Pr[\mathsf{Game}_{3,\mathsf{A}} \Rightarrow 1| = \mathsf{negl}(\lambda).$$

Proof. Let $\mathbf{BAD}_{H0}$ be the event that the random oracle aborts. If $\mathbf{BAD}_{H0}$ does not occur, the views of A are identical in $\mathsf{Game}_{2,\mathsf{A}}$ and $\mathsf{Game}_{3,\mathsf{A}}$. The probability that $\mathbf{BAD}_{H0}$ occurs is bounded by the following reduction R to the CSI-GDH problem. The reduction R is given as follows.

- **First procedure of R:** Given a CSI-GDH problem instance $(E_0, [\mathfrak{x}] * E_0, [\mathfrak{y}] * E_0)$, R sets $\mathrm{pk}_{s_0} = [\mathfrak{x}] * E_0$, $\mathrm{pk}_v = [\mathfrak{y}] * E_0$, runs $(\mathrm{pk}_{s_1}, \mathrm{sk}_{s_1}) \leftarrow \mathsf{KGen}(1^\lambda)$ and gives $(\mathrm{pk}_{s_0}, \mathrm{pk}_{s_1}, \mathrm{pk}_{s_v})$ to A. Then R simulates the view of A without sk_{s_0} and sk_v until $\mathbf{BAD}_{H0}$ occurs.
- **Simulation of $\mathcal{O}^H$:** For a query on $(m, K, \mathbf{R}_s, \mathbf{R}_v)$, R queries $(E_0, [\mathfrak{x}] * E_0, [\mathfrak{y}] * E_0, K)$ to the $\mathcal{O}^{\mathsf{CSI\text{-}DDH}}$ oracle of the CSI-GDH problem and obtains the result β. If $\beta = 0$, R simulates $\mathcal{O}^H$ in the same way as $\mathcal{O}^{\mathsf{Sign}}$ in $\mathsf{Game}_{3,\mathsf{A}}$. If $\beta = 1$ (i.e., $\mathbf{BAD}_{H0}$ occurs), W abort the simulation for A and returns a solution K for the CSI-GDH problem.

- **Simulation of $\mathcal{O}^{\mathsf{Sign}}$:** For a query on (d, m), R simulates $\mathcal{O}^{\mathsf{Sign}}$ in the same way as $\mathcal{O}^{\mathsf{Sign}}$ in $\mathsf{Game}_{3,\mathsf{A}}$.
- **Simulation of $\mathcal{O}^{\mathsf{Ver}}$:** For a query on (d, m, σ), R simulates $\mathcal{O}^{\mathsf{Sign}}$ in the same way as $\mathcal{O}^{\mathsf{Sign}}$ in $\mathsf{Game}_{3,\mathsf{A}}$.

  If $\mathbf{BAD}_{H0}$ occurs, R outputs a valid solution for the CSI-GDH problem. Thus, we obtain the following bound:

$$|\Pr[\mathsf{Game}_{2,\mathsf{A}} \Rightarrow 1] - \Pr[\mathsf{Game}_{3,\mathsf{A}} \Rightarrow 1| \leq \mathsf{Adv}_{\mathsf{R}}^{\mathsf{CSI\text{-}GDH}}(\lambda) = \mathsf{negl}(\lambda).$$

(Lemma 5) □

Lemma 6. *If the* $\mathsf{SDVS}_{\mathsf{Ours}}$ *satisfies strong unforgeability,*

$$|\Pr[\mathsf{Game}_{3,\mathsf{A}} \Rightarrow 1] - \Pr[\mathsf{Game}_{4,\mathsf{A}} \Rightarrow 1| = \mathsf{negl}(\lambda).$$

The proof of Lemma 6 is obtained in a similar way to Lemma 3.

Proof. Let $\mathbf{BAD}_{V1}$ be the event that A queries $(1, m, \sigma)$ to $\mathcal{O}^{\mathsf{Ver}}$ such that $(1, m, \sigma) \notin L^{\mathsf{Sign}}$ and $\mathsf{Ver}(\mathrm{sk}_v, pk_s, pk_v, m, \sigma) = 1$. If the event $\mathbf{BAD}_{V1}$ does not occur, the views of A are identical in $\mathsf{Game}_{3,\mathsf{A}}$ and $\mathsf{Game}_{4,\mathsf{A}}$. The probability that $\mathbf{BAD}_{V1}$ occurs is bounded by the following reduction R to strong unforgeability security of $\mathsf{SDVS}_{\mathsf{Ours}}$. The reduction R is given as follows.

- **First procedure of R:** Given an input $(\mathrm{pk}_s, \mathrm{pk}_v)$ from the challenger C of the strong unforgeability game, R sets $\mathrm{pk}_{s_1} = \mathrm{pk}_s$, runs $(\mathrm{pk}_{s_0}, \mathrm{sk}_{s_0}) \leftarrow \mathsf{KGen}(1^\lambda)$ and gives $(\mathrm{pk}_{s_0}, \mathrm{pk}_{s_1}, \mathrm{pk}_{s_v})$ to A. Then R simulates the view of A in $\mathsf{Game}_{4,\mathsf{A}}$ without sk_{s_1} and sk_v until $\mathbf{BAD}_{V1}$ occurs.
- **Simulation of $\mathcal{O}^H$:** For a query on $(m, K, \mathbf{R}_s, \mathbf{R}_v)$, R simulates $\mathcal{O}^H$ in the same way as $\mathcal{O}^{\mathsf{Sign}}$ in $\mathsf{Game}_{4,\mathsf{A}}$ except that the check $K = [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_0}$ is changed to $K = [\mathfrak{g}^{x_{s_0}}] * \mathrm{pk}_v$.
- **Simulation of $\mathcal{O}^{\mathsf{Sign}}$:** For a query on (d, m), if $d = 0$, R simulates $\mathcal{O}^{\mathsf{Sign}}$ in the same way as $\mathcal{O}^{\mathsf{Sign}}$ in $\mathsf{Game}_{4,\mathsf{A}}$. If $d = 1$, R queries m to $\mathcal{O}^{\mathsf{Sign}}$ in the strong unforgeability game and obtains a signature σ. Then R records (d, m, σ) to L^{Sign} and returns σ.
- **Simulation of $\mathcal{O}^{\mathsf{Ver}}$:** For a query on (d, m, σ), if $d = 0$, R simulates $\mathcal{O}^{\mathsf{Sign}}$ in the same way as $\mathcal{O}^{\mathsf{Sign}}$ in $\mathsf{Game}_{4,\mathsf{A}}$. If $d = 1$, if there is an entry $(1, m, \sigma)$ in L^{Sign}, return 1. Otherwise, R queries (m, σ) to $\mathcal{O}^{\mathsf{Sign}}$ in the strong unforgeability game and obtains a bit β. If $\beta = 0$, return 0 to A. If $\beta = 1$ (i.e., $\mathbf{BAD}_{V1}$ occurs), abort the simulation for A and returns a forgery (m, σ) to C.

If $\mathbf{BAD}_{V1}$ occurs, R outputs a valid forgery for the strong unforgeability game. Thus, we obtain the bound:

$$|\Pr[\mathsf{Game}_{3,\mathsf{A}} \Rightarrow 1] - \Pr[\mathsf{Game}_{4,\mathsf{A}} \Rightarrow 1| \leq \mathsf{Adv}_{\mathsf{SDVS}_{\mathsf{Ours}},\mathsf{R}}^{\mathsf{SUF}} = \mathsf{negl}(\lambda).$$

(Lemma 6) □

Lemma 7.

$$|\Pr[\mathsf{Game}_{4,\mathsf{A}} \Rightarrow 1] - \Pr[\mathsf{Game}_{5,\mathsf{A}} \Rightarrow 1| = \mathsf{negl}(\lambda).$$

The proof of Lemma 7 can be obtained in a way similar to that of Lemma 4.

Proof. Let Q_{Sign}, Q_{Ver} and Q_H be the total number of queries from A to $\mathcal{O}^{\mathsf{Sign}}$, $\mathcal{O}^{\mathsf{Ver}}$, $\mathcal{O}^H$, respectively. Let $\mathbf{BAD}_{S1}$ be the event that the signing oracle aborts when $(1, m)$ is queried by A. If $\mathbf{BAD}_{S1}$ does not occur, the views of A are identical in $\mathsf{Game}_{4,\mathsf{A}}$ and $\mathsf{Game}_{5,\mathsf{A}}$.

We bound the probability that $\mathbf{BAD}_{S1}$ occurs. For each signing query $(1, m)$, the probability that $\mathcal{O}^{\mathsf{Sign}}$ aborts is at most $\frac{Q_{\mathsf{Sign}}+Q_{\mathsf{Ver}}+Q_H+1}{\#\mathrm{Cl}_p^\lambda}$. By taking the union bound, we have

$$\begin{aligned}|\Pr[\mathsf{Game}_{4,\mathsf{A}} \Rightarrow 1] - \Pr[\mathsf{Game}_{5,\mathsf{A}} \Rightarrow 1]| &\leq (Q_{\mathsf{Sign}} + 1) \cdot \frac{Q_{\mathsf{Sign}} + Q_{\mathsf{Ver}} + Q_H + 1}{N^\lambda} \\ &= \mathsf{negl}(\lambda).\end{aligned}$$

(Lemma 7) □

Lemma 8. *If the CSI-GDH assumption holds,*

$$|\Pr[\mathsf{Game}_{5,\mathsf{A}} \Rightarrow 1] - \Pr[\mathsf{Game}_{6,\mathsf{A}} \Rightarrow 1]| = \mathsf{negl}(\lambda).$$

The proof of Lemma 8 can be obtained in a way similar to that of Lemma 5.

Proof. Let $\mathbf{BAD}_{H1}$ be the event that the random oracle aborts when a tuple $(m, K, \mathbf{R}_s, \mathbf{R}_v)$ is queried such that $K = [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_1}$. If $\mathbf{BAD}_{H1}$ does not occur, the views of A are identical in $\mathsf{Game}_{5,\mathsf{A}}$ and $\mathsf{Game}_{6,\mathsf{A}}$. The probability that $\mathbf{BAD}_{H1}$ occurs is bounded by the following reduction R to the CSI-GDH problem. The reduction R is given as follows.

- **First procedure of** R**:** Given a CSI-GDH problem instance $(E_0, [\mathfrak{x}] * E_0, [\mathfrak{y}] * E_0)$, R sets $\mathrm{pk}_{s_1} = [\mathfrak{x}] * E_0$, $\mathrm{pk}_v = [\mathfrak{y}] * E_0$, runs $(\mathrm{pk}_{s_0}, \mathrm{sk}_{s_0}) \leftarrow \mathsf{KGen}(1^\lambda)$ and gives $(\mathrm{pk}_{s_0}, \mathrm{pk}_{s_1}, \mathrm{pk}_{s_v})$ to A. Then R simulates the view of A without sk_{s_1} and sk_v until $\mathbf{BAD}_{H1}$ occurs.
- **Simulation of** $\mathcal{O}^H$**:** For a query on $(m, K, \mathbf{R}_s, \mathbf{R}_v)$, R queries $(E_0, \mathrm{pk}_{s_0}, [\mathfrak{y}] * E_0, K)$ to the $\mathcal{O}^{\mathsf{CSI\text{-}DDH}}$ oracle of the CSI-GDH problem and obtains the result β. If $\beta = 1$, R aborts.
  R queries $(E_0, [\mathfrak{x}] * E_0, [\mathfrak{y}] * E_0, K)$ to the CSI-DDH oracle of the CSI-GDH problem and obtains the result β'. If $\beta' = 0$, R aborts. If $\beta' = 0$, R simulates $\mathcal{O}^H$ in the same way as $\mathcal{O}^{\mathsf{Sign}}$ in $\mathsf{Game}_{6,\mathsf{A}}$. If $\beta = 1$ (i.e., $\mathbf{BAD}_{H1}$ occurs), W abort the simulation for A and returns a solution K for the CSI-GDH problem.
- **Simulation of** $\mathcal{O}^{\mathsf{Sign}}$**:** For a query on (d, m), R simulates $\mathcal{O}^{\mathsf{Sign}}$ in the same way as $\mathcal{O}^{\mathsf{Sign}}$ in $\mathsf{Game}_{6,\mathsf{A}}$.
- **Simulation of** $\mathcal{O}^{\mathsf{Ver}}$**:** For a query on (d, m, σ), R simulates $\mathcal{O}^{\mathsf{Sign}}$ in the same way as $\mathcal{O}^{\mathsf{Sign}}$ in $\mathsf{Game}_{6,\mathsf{A}}$.
  If $\mathbf{BAD}_{H1}$ occurs, R outputs a valid solution for the CSI-GDH problem. Thus, we obtain the following bound:

$$|\Pr[\mathsf{Game}_{5,\mathsf{A}} \Rightarrow 1] - \Pr[\mathsf{Game}_{6,\mathsf{A}} \Rightarrow 1]| \leq \mathsf{Adv}_{\mathsf{R}}^{\mathsf{CSI\text{-}GDH}}(\lambda) = \mathsf{negl}(\lambda).$$

(Lemma 8) □

Lemma 9.

$$\Pr[\mathsf{Game}_{6,\mathsf{A}} \Rightarrow 1] = \frac{1}{2}.$$

Proof. Let m^* be a challenge and $\sigma^* = (\mathbf{c}_s^*, \mathbf{c}_v^*, \mathbf{z}_s^*, \mathbf{z}_v^*) \leftarrow \mathsf{Sign}(\mathrm{sk}_{s_b}, \mathrm{pk}_{s_b}, \mathrm{pk}_v, m^*)$ be its signature. Let $\mathbf{R}_s^* \leftarrow [\mathfrak{g}^{\mathbf{z}_s^*}] * \mathrm{pk}_s^{\mathbf{c}_s^*}$, $\mathbf{R}_v^* \leftarrow [\mathfrak{g}^{\mathbf{z}_v^*}] * \mathrm{pk}_v^{\mathbf{c}_v^*}$, $\mathbf{c}^* \leftarrow \mathbf{c}_s^* \odot \mathbf{c}_v^*$. If Game_6 does not abort at the end of the game, this implies that A does not query neither $(m^*, [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_0} = [\mathfrak{g}^{x_{s_0}}] * \mathrm{pk}_v, \mathbf{R}_s^*, \mathbf{R}_v^*)$ nor $(m^*, [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_1} = [\mathfrak{g}^{x_{s_1}}] * \mathrm{pk}_v, \mathbf{R}_s^*, \mathbf{R}_v^*)$ to $\mathcal{O}^H$. We see that A knows $\mathbf{c}^*$. However, A does not have any information to distinguish whether $H(m^*, [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_0} = [\mathfrak{g}^{x_{s_0}}] * \mathrm{pk}_v, \mathbf{R}_s^*, \mathbf{R}_v^*) = \mathbf{c}^*$ or $H(m^*, [\mathfrak{g}^{x_v}] * \mathrm{pk}_{s_1} = [\mathfrak{g}^{x_{s_1}}] * \mathrm{pk}_v, \mathbf{R}_s^*, \mathbf{R}_v^*) = \mathbf{c}^*$ at all. From this fact, we have

$$\Pr[\mathsf{Game}_{6,\mathsf{A}} \Rightarrow 1] = \frac{1}{2}.$$

(Lemma 9) □

From Lemma 3 to Lemma 9, we have

$$\begin{aligned}
\mathsf{Adv}^{\mathsf{PSI}}_{\mathsf{SDVS},\mathsf{A}}(\lambda) &= \left|\Pr[\mathsf{Game}^{\mathsf{PSI}}_{\mathsf{SDVS},\mathsf{A}}(1^\lambda) \Rightarrow 1] - \frac{1}{2}\right| \\
&= \left|\sum_{i=0}^{5} (\Pr[\mathsf{Game}_{i,\mathsf{A}} \Rightarrow 1] - \Pr[\mathsf{Game}_{i+1,\mathsf{A}} \Rightarrow 1]) - \frac{1}{2}\right| \\
&\leq \sum_{i=0}^{5} |\Pr[\mathsf{Game}_{i,\mathsf{A}} \Rightarrow 1] - \Pr[\mathsf{Game}_{i+1,\mathsf{A}} \Rightarrow 1]| + \left|\Pr[\mathsf{Game}_{6,\mathsf{A}} \Rightarrow 1] - \frac{1}{2}\right| \\
&= \mathsf{negl}(\lambda).
\end{aligned}$$

(Theorem 3) □

4.5 Non-Delegatability Analysis

In this section, we prove non-delegatability for our SDVS.

Theorem 4. *If there is an algorithm F_m that generates valid signatures on a message m in polynomial time and the probability ϵ with at most Q_H queries to the random oracle, then $\mathsf{SDVS}_{\mathsf{Ours}}$ satisfies non-delegatability with knowledge error $2^{-\lambda}$ in the ROM.*

Proof. We assume that $\epsilon > \kappa = 2^{-\lambda}$, where $2^{-\lambda}$ is the probability that the adversary guesses the value of $H(m, [\mathfrak{g}^{x_s + x_v}] * E_0, E_s, E_v)$ without querying $(m, [\mathfrak{g}^{x_s + x_v}] * E_0, E_s, E_v)$ to the random oracle. Let F_{m^*} be a forger with input message m^* and Q_H be the total number of hash queries from F_{m^*} to $\mathcal{O}^H$. To describe Ext, we introduce the following algorithm W.

Construction of $\mathsf{W}^{\mathsf{F}_{m^*}}$. $\mathsf{W}(m^*, \mathbf{c}_1, \ldots, \mathbf{c}_{Q_H})$ runs F_{m^*} and simulates hash queries from F_{m^*}.

- **First procedure of W:** W takes $(m^*, \mathbf{c}_1, \ldots, \mathbf{c}_{Q_H})$ as an input and initializes a hash table $\mathbb{T} \leftarrow \{\}$ and a counter $ctr \leftarrow 0$. Then W runs F_{m^*}.
- **Simulation of $\mathcal{O}^H$:** For a query $(m, K, \mathbf{R}_s, \mathbf{R}_v)$, if $\mathbb{T}[m, K, \mathbf{R}_s, \mathbf{R}_v] = c$ for some $\mathbf{c}$, W returns $\mathbf{c}$. Otherwise, W sets $ctr \leftarrow ctr + 1$, defines $\mathbb{T}[m, K, \mathbf{R}_s, \mathbf{R}_v] = \mathbf{c}_{ctr}$, and returns $\mathbf{c}_{ctr}$.
- **Final procedure of W:** W receives a signature $(m^*, \sigma^* = (\mathbf{c}_s^*, \mathbf{z}_s^*, \mathbf{c}_v^*, \mathbf{z}_v^*))$ from F_{m^*}. Then W computes $\mathbf{R}_s^* \leftarrow [\mathfrak{g}^{\mathbf{z}_s^*}] * \mathrm{pk}_s^{\mathbf{c}_s^*}$, $\mathbf{R}_v^* \leftarrow [\mathfrak{g}^{\mathbf{z}_v^*}] * \mathrm{pk}_v^{\mathbf{c}_v^*}$. If there is no entry $((m^*, K^*, \mathbf{R}_s^*, \mathbf{R}_v^*), \mathbf{c})$ for any $\mathbf{c}$, then W aborts. Otherwise, W returns $(j, \mathrm{out}) \leftarrow (ctr^*, (m^*, \sigma^*))$ where ctr^* is the values of ctr at the moment when $\mathbb{T}[m^*, K^*, \mathbf{R}_s^*, \mathbf{R}_v^*] = \mathbf{c}_{ctr}$ is defined.

Construction of $\mathsf{Ext}^{\mathsf{F}_{m^*}}(m^*)$. The extraction algorithm $\mathsf{Ext}^{\mathsf{F}_{m^*}}(m^*)$ runs $\mathsf{Fork}^{\mathsf{W}}$ and extracts the sk_s or sk_v.

- Given an input m^*, Ext runs $(d, \mathrm{out}, \widetilde{\mathrm{out}}) \leftarrow \mathsf{Fork}^{\mathsf{W}}(m^*)$. If $d = 0$, Ext aborts.
- Ext parses $(m^*, \sigma^* = (\mathbf{c}_s^*, \mathbf{z}_s^*, \mathbf{c}_v^*, \mathbf{z}_v^*)) \leftarrow \mathrm{out}, (\widetilde{m}^*, \widetilde{\sigma}^* = (\widetilde{\mathbf{c}}_s^*, \widetilde{\mathbf{z}}_s^*, \widetilde{\mathbf{c}}_v^*, \widetilde{\mathbf{z}}_v^*)) \leftarrow \widetilde{\mathrm{out}}$.
- If $\mathbf{c}_s^* \neq \widetilde{\mathbf{c}}_s^*$, Ext finds an index i^* such that $\mathbf{c}_s^*[i^*] \neq \widetilde{\mathbf{c}}_s^*[i^*]$. Then Ext outputs $\mathrm{sk}_s = \frac{\widetilde{\mathbf{z}}_s^*[i^*] - \mathbf{z}_s^*[i^*]}{\mathbf{c}_s^*[i^*] - \widetilde{\mathbf{c}}_s^*[i^*]}$.
- If $\beta = 1 \wedge \mathbf{c}_v^* \neq \widetilde{\mathbf{c}}_v^*$, Ext finds an index i^* such that $\mathbf{c}_v^*[i^*] \neq \widetilde{\mathbf{c}}_v^*[i^*]$. Then Ext outputs $\mathrm{sk}_v = \frac{\widetilde{\mathbf{z}}_v^*[i^*] - \mathbf{z}_v^*[i^*]}{\mathbf{c}_v^*[i^*] - \widetilde{\mathbf{c}}_v^*[i^*]}$.

Here, we briefly analyze Ext. If $\mathsf{Fork}^{\mathsf{W}}(m^*)$ outputs $(d, \mathrm{out} = (\mathbf{c}_s^*, \mathbf{z}_s^*, \mathbf{c}_v^*, \mathbf{z}_v^*),$ $\widetilde{\mathrm{out}} = (\widetilde{\mathbf{c}}_s^*, \widetilde{\mathbf{z}}_s^*, \widetilde{\mathbf{c}}_v^*, \widetilde{\mathbf{z}}_v^*))$ with $d = 1$, we have $\mathbf{c}^* = \mathbf{c}_s^* \odot \mathbf{c}_v^* \neq \widetilde{\mathbf{c}}_s^* \odot \widetilde{\mathbf{c}}_v^* = \widetilde{\mathbf{c}}^*$. This implies that $\mathbf{c}_s^* \neq \widetilde{\mathbf{c}}_s^*$ or $\mathbf{c}_s^* \neq \widetilde{\mathbf{c}}_s^*$ holds. If $\mathbf{c}_s^* \neq \widetilde{\mathbf{c}}_s^*$, there exists i^* such that $\mathbf{c}_s^*[i] \neq \widetilde{\mathbf{c}}_s^*[i]$. By the same argument as in the proof of Theorem 2, we have $[\mathfrak{g}^{\mathbf{z}_s^*[i^*]}] * \mathrm{pk}_s^{\mathbf{c}_s^*[i^*]} = [\mathfrak{g}^{\widetilde{\mathbf{z}}_s^*[i^*]}] * \mathrm{pk}_s^{\widetilde{\mathbf{c}}_s^*[i^*]}$. Let $\mathrm{pk}_s = [\mathfrak{g}^{x_s}] * E_0$. Then $[\mathfrak{g}^{\mathbf{z}_s^*[i^*] + x_s \mathbf{c}_s^*[i^*]}] * E_0 = [\mathfrak{g}^{\widetilde{\mathbf{z}}_s^*[i^*] + x_s \widetilde{\mathbf{c}}_s^*[i^*]}] * E_0$ holds. Thus, $x_s = \frac{\widetilde{\mathbf{z}}_s^*[i^*] - \mathbf{z}_s^*[i^*]}{\mathbf{c}_s^*[i^*] - \widetilde{\mathbf{c}}_s^*[i^*]}$ holds. In the case of $\mathbf{c}_v^* \neq \widetilde{\mathbf{c}}_v^*$ can be analyzed in the same way as discussed above.

Let ϵ_{Ext} be the probability that $\mathsf{Ext}^{\mathsf{F}_{m^*}}(m^*)$ produces either sk_s or sk_v. By the forking lemma (Lemma 2), we have

$$\begin{aligned}\epsilon_{\mathsf{Ext}} = \mathsf{frk} &\geq \frac{1}{2} \cdot \mathsf{acc} \cdot \left(\frac{\mathsf{acc}}{Q_{\mathsf{Sign}}} - \frac{1}{2^\lambda} \right) \\ &\geq \frac{1}{Q_{\mathsf{Sign}}} \cdot \left(\epsilon - \frac{1}{2^\lambda} \right) \cdot \left(\epsilon - \frac{Q_{\mathsf{Sign}} + 1}{2^\lambda} \right).\end{aligned}$$

(Theorem 4) □

Appendix

A Based $\varSigma$-protocol for OR Relation

A.1 $\varSigma$-protocol

A $\varSigma$-protocol for an NP relation $R \subseteq \{0,1\}^* \times \{0,1\}^*$ is a special type of public-coin three-move interactive protocol between a prover and a verifier.

Definition 8. *A $\varSigma$-protocol $\varPi$ for a NP relation R is a three-move public-coin interactive protocol with between PPT algorithms $\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2)$ and V. The $\varSigma$ is executed by the following procedure:*

- *The prover takes a statement x and witness w as an input, runs $(st, com) \leftarrow \mathsf{P}_1(x, w)$ and obtains a state st and a commitment com. The prover sends com to the verifier.*
- *The verifier samples $ch \xleftarrow{\$} CH$ where CH is a challenge space of $\varSigma$-protocol. The verifier sends ch to the prover.*
- *The prover runs $resp \leftarrow \mathsf{P}_2(st, ch)$. The prover sends the response $resp$ to the verifier.*
- *The verifier runs $b \leftarrow \mathsf{V}(x, com, ch, resp)$.*

A.2 $\varSigma$-Protocol for OR Relation by Katsumata et al. [19,20]

We consider the following NP relation R.

$$R = \{((E_0, E_0', E_1'), (\delta, x_\delta)) | E_\delta' = [\mathfrak{g}^{x_\delta}] * E_0, (\delta, x) \in \{0,1\} \times \mathbb{Z}_N\}$$

Katsumata et al. [19,20] gave the $\varSigma$-protocol for this relation R. We describe their $\varSigma$-protocol $\varPi = (\mathsf{P} = (\mathsf{P}_1, \mathsf{P}_2), \mathsf{V})$ with the challenge space $CH = \{0,1\}^\lambda$ as follows:

- $\mathsf{P}_1(x = (E_0, E_0', E_1'), w = (\delta, x_\delta))$: Sample $\mathbf{r}_\delta, \mathbf{z}_{1-\delta} \xleftarrow{\$} \mathbb{Z}_N^\lambda$, $\mathbf{c}_{1-\delta} \xleftarrow{\$} \{\pm 1\}^\lambda$, set $\mathbf{R}_\delta \leftarrow [\mathfrak{g}^{\mathbf{r}_\delta}] * E_0$, $\mathbf{R}_{1-\delta} \leftarrow [\mathfrak{g}^{\mathbf{z}_{1-\delta}}] * \mathrm{pk}_{1-\delta}^{\mathbf{c}_{1-\delta}}$, and return $(com, st) \leftarrow ((\mathbf{R}_0, \mathbf{R}_1), (\delta, x_\delta, \mathbf{c}_{1-\delta}, \mathbf{r}_\delta, \mathbf{z}_{1-\delta}))$
- $\mathsf{P}_2(st = (\delta, x_\delta, \mathbf{c}_{1-\delta}, \mathbf{r}_\delta, \mathbf{z}_{1-\delta}), ch = \mathbf{c})$: Set $\mathbf{c}_s \leftarrow \mathbf{c} \odot \mathbf{c}_v$, $\mathbf{z}_s \leftarrow \mathbf{r}_s - x_s \cdot \mathbf{c}_s$, and return $resp = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{z}_0, \mathbf{z}_1)$.
- $\mathsf{V}(x = (E_0, E_0', E_1'), com = (\mathbf{R}_0, \mathbf{R}_1), ch = \mathbf{c}, resp = (\mathbf{c}_0, \mathbf{c}_1, \mathbf{z}_0, \mathbf{z}_1))$: If $\mathbf{c} = \mathbf{c}_0 \odot \mathbf{c}_1 \wedge \mathbf{R}_0 = [\mathfrak{g}^{\mathbf{z}_0}] * {E_0'}^{\mathbf{c}_0} \wedge \mathbf{R}_1 = [\mathfrak{g}^{\mathbf{z}_1}] * {E_1'}^{\mathbf{c}_1}$, return 1. Otherwise, return 0.

Their protocol satisfies perfect honest verifier zero-knowldege (HVZK). Since perfect HVZK implies that perfect witness indistinguishability (WI). Our proposed SDVS scheme $\mathsf{SDVS}_{\mathsf{Ours}}$ is obtained by transforming this protocol to the non-interactive one via the Fiat-Shamir transformation. Thanks to the perfect WI property, our scheme SDVS satisfies perfect non-transferability.

References

1. M. R. Asaar and M. Salmasizadeh. A novel strong designated verifier signature scheme without random oracles. *IACR Cryptol. ePrint Arch.*, page 259, 2012.
2. H. Assidi and E. M. Souidi. Strong designated verifier signature based on the rank metric. In M. Laurent and T. Giannetsos, editors, *Information Security Theory and Practice - 13th IFIP WG 11.2 International Conference, WISTP 2019, Paris, France, December 11-12, 2019, Proceedings*, volume 12024 of *Lecture Notes in Computer Science*, pages 85–102. Springer, 2019.
3. A. Basso and L. Maino. Poké: A compact and efficient PKE from higher-dimensional isogenies. In S. Fehr and P. Fouque, editors, *Advances in Cryptology - EUROCRYPT 2025 - 44th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Madrid, Spain, May 4-8, 2025, Proceedings, Part II*, volume 15602 of *Lecture Notes in Computer Science*, pages 94–123. Springer, 2025.
4. M. Bellare and G. Neven. Multi-signatures in the plain public-key model and a general forking lemma. In A. Juels, R. N. Wright, and S. D. C. di Vimercati, editors, *Proceedings of the 13th ACM Conference on Computer and Communications Security, CCS 2006, Alexandria, VA, USA, October 30 - November 3, 2006*, pages 390–399. ACM, 2006.
5. M. Bellare and P. Rogaway. Random oracles are practical: A paradigm for designing efficient protocols. In D. E. Denning, R. Pyle, R. Ganesan, R. S. Sandhu, and V. Ashby, editors, *CCS '93, Proceedings of the 1st ACM Conference on Computer and Communications Security, Fairfax, Virginia, USA, November 3-5, 1993*, pages 62–73. ACM, 1993.
6. W. Beullens, T. Kleinjung, and F. Vercauteren. Csi-fish: Efficient isogeny based signatures through class group computations. In S. D. Galbraith and S. Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 227–247. Springer, 2019.
7. J. Cai, H. Jiang, P. Zhang, Z. Zheng, G. Lyu, and Q. Xu. An efficient strong designated verifier signature based on r-sis assumption. *IEEE Access*, 7:3938–3947, 2019.
8. W. Castryck and T. Decru. An efficient key recovery attack on SIDH. In C. Hazay and M. Stam, editors, *Advances in Cryptology - EUROCRYPT 2023 - 42nd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Lyon, France, April 23-27, 2023, Proceedings, Part V*, volume 14008 of *Lecture Notes in Computer Science*, pages 423–447. Springer, 2023.
9. W. Castryck, T. Lange, C. Martindale, L. Panny, and J. Renes. CSIDH: an efficient post-quantum commutative group action. In T. Peyrin and S. D. Galbraith, editors, *Advances in Cryptology - ASIACRYPT 2018 - 24th International Conference on the Theory and Application of Cryptology and Information Security, Brisbane, QLD, Australia, December 2-6, 2018, Proceedings, Part III*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018.
10. D. Chaum and H. V. Antwerpen. Undeniable signatures. In G. Brassard, editor, *Advances in Cryptology - CRYPTO '89, 9th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 1989, Proceedings*, volume 435 of *Lecture Notes in Computer Science*, pages 212–216. Springer, 1989.

11. D. Feng, J. Xu, and W. Chen. Generic constructions for strong designated verifier signature. *J. Inf. Process. Syst.*, 7(1):159–172, 2011.
12. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In A. M. Odlyzko, editor, *Advances in Cryptology - CRYPTO '86, Santa Barbara, California, USA, 1986, Proceedings*, volume 263 of *Lecture Notes in Computer Science*, pages 186–194. Springer, 1986.
13. E. Fujisaki and T. Okamoto. Secure integration of asymmetric and symmetric encryption schemes. In M. J. Wiener, editor, *Advances in Cryptology - CRYPTO '99, 19th Annual International Cryptology Conference, Santa Barbara, California, USA, August 15-19, 1999, Proceedings*, volume 1666 of *Lecture Notes in Computer Science*, pages 537–554. Springer, 1999.
14. B. Gong, M. H. Au, and H. Xue. Constructing strong designated verifier signatures from key encapsulation mechanisms. In *18th IEEE International Conference On Trust, Security And Privacy In Computing And Communications / 13th IEEE International Conference On Big Data Science And Engineering, TrustCom/BigDataSE 2019, Rotorua, New Zealand, August 5-8, 2019*, pages 586–593. IEEE, 2019.
15. D. Hofheinz, K. Hövelmanns, and E. Kiltz. A modular analysis of the fujisaki-okamoto transformation. In Y. Kalai and L. Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part I*, volume 10677 of *Lecture Notes in Computer Science*, pages 341–371. Springer, 2017.
16. Q. Huang, G. Yang, D. S. Wong, and W. Susilo. Efficient strong designated verifier signature schemes without random oracle or with non-delegatability. *Int. J. Inf. Sec.*, 10(6):373–385, 2011.
17. X. Huang, W. Susilo, Y. Mu, and F. Zhang. Short (identity-based) strong designated verifier signature schemes. In K. Chen, R. H. Deng, X. Lai, and J. Zhou, editors, *Information Security Practice and Experience, Second International Conference, ISPEC 2006, Hangzhou, China, April 11-14, 2006, Proceedings*, volume 3903 of *Lecture Notes in Computer Science*, pages 214–225. Springer, 2006.
18. M. Jakobsson, K. Sako, and R. Impagliazzo. Designated verifier proofs and their applications. In U. M. Maurer, editor, *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer, 1996.
19. S. Katsumata, Y. Lai, J. T. LeGrow, and L. Qin. CSI -otter: Isogeny-based (partially) blind signatures from the class group action with a twist. In H. Handschuh and A. Lysyanskaya, editors, *Advances in Cryptology - CRYPTO 2023 - 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20-24, 2023, Proceedings, Part III*, volume 14083 of *Lecture Notes in Computer Science*, pages 729–761. Springer, 2023.
20. S. Katsumata, Y. Lai, J. T. LeGrow, and L. Qin. Csi-otter: isogeny-based (partially) blind signatures from the class group action with a twist. *Des. Codes Cryptogr.*, 92(11):3587–3643, 2024.
21. T. Kawashima, K. Takashima, Y. Aikawa, and T. Takagi. An efficient authenticated key exchange from random self-reducibility on CSIDH. In D. Hong, editor, *Information Security and Cryptology - ICISC 2020 - 23rd International Conference, Seoul, South Korea, December 2-4, 2020, Proceedings*, volume 12593 of *Lecture Notes in Computer Science*, pages 58–84. Springer, 2020.

22. T. X. Khuc, W. Susilo, D. H. Duong, Y. Li, P. S. Roy, K. Fukushima, and S. Kiyomoto. Strong designated verifier signatures from isogeny assumptions. In Y. Kim, A. Miyaji, and M. Tibouchi, editors, *Cryptology and Network Security - 24th International Conference, CANS 2025, Osaka, Japan, November 17-20, 2025, Proceedings*, volume 16351 of *Lecture Notes in Computer Science*, pages 46–69. Springer, 2025.
23. F. Laguillaumie and D. Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In C. Blundo and S. Cimato, editors, *Security in Communication Networks, 4th International Conference, SCN 2004, Amalfi, Italy, September 8-10, 2004, Revised Selected Papers*, volume 3352 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 2004.
24. H. Lipmaa, G. Wang, and F. Bao. Designated verifier signature schemes: Attacks, new security notions and a new construction. In L. Caires, G. F. Italiano, L. Monteiro, C. Palamidessi, and M. Yung, editors, *Automata, Languages and Programming, 32nd International Colloquium, ICALP 2005, Lisbon, Portugal, July 11-15, 2005, Proceedings*, volume 3580 of *Lecture Notes in Computer Science*, pages 459–471. Springer, 2005.
25. K. Nakagawa and H. Onuki. QFESTA: efficient algorithms and parameters for FESTA using quaternion algebras. In L. Reyzin and D. Stebila, editors, *Advances in Cryptology - CRYPTO 2024 - 44th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2024, Proceedings, Part V*, volume 14924 of *Lecture Notes in Computer Science*, pages 75–106. Springer, 2024.
26. S. Poddar, S. Mishra, T. Mohanty, V. Srivastava, and S. Gangopadhyay. Lasdvs : A post-quantum secure compact strong-designated verifier signature. *CoRR*, abs/2504.16571, 2025.
27. F. Renan. A compact post-quantum strong designated verifier signature scheme from isogenies. *IACR Cryptol. ePrint Arch.*, page 1335, 2025.
28. S. Saeednia, S. Kremer, and O. Markowitch. An efficient strong designated verifier signature scheme. In J. I. Lim and D. H. Lee, editors, *Information Security and Cryptology - ICISC 2003, 6th International Conference, Seoul, Korea, November 27-28, 2003, Revised Papers*, volume 2971 of *Lecture Notes in Computer Science*, pages 40–54. Springer, 2003.
29. S. F. Shahandashti and R. Safavi-Naini. Construction of universal designated-verifier signatures and identity-based signatures from standard signatures. In R. Cramer, editor, *Public Key Cryptography - PKC 2008, 11th International Workshop on Practice and Theory in Public-Key Cryptography, Barcelona, Spain, March 9-12, 2008. Proceedings*, volume 4939 of *Lecture Notes in Computer Science*, pages 121–140. Springer, 2008.
30. M. K. Shooshtari, M. Ahmadian-Attari, and M. R. Aref. Provably secure strong designated verifier signature scheme based on coding theory. *Int. J. Commun. Syst.*, 30(7), 2017.
31. R. Steinfeld, L. Bull, H. Wang, and J. Pieprzyk. Universal designated-verifier signatures. In C. Laih, editor, *Advances in Cryptology - ASIACRYPT 2003, 9th International Conference on the Theory and Application of Cryptology and Information Security, Taipei, Taiwan, November 30 - December 4, 2003, Proceedings*, volume 2894 of *Lecture Notes in Computer Science*, pages 523–542. Springer, 2003.
32. R. Steinfeld, H. Wang, and J. Pieprzyk. Efficient extension of standard schnorr/rsa signatures into universal designated-verifier signatures. In F. Bao, R. H. Deng, and J. Zhou, editors, *Public Key Cryptography - PKC 2004, 7th International Workshop*

on Theory and Practice in Public Key Cryptography, Singapore, March 1-4, 2004, volume 2947 of *Lecture Notes in Computer Science*, pages 86–100. Springer, 2004.

33. X. Sun, H. Tian, and Y. Wang. Toward quantum-resistant strong designated verifier signature from isogenies. In F. Xhafa, L. Barolli, F. Pop, X. Chen, and V. Cristea, editors, *2012 Fourth International Conference on Intelligent Networking and Collaborative Systems, INCoS 2012, Bucharest, Romania, September 19-21, 2012*, pages 292–296. IEEE, 2012.
34. P. Thanalakshmi, R. Anitha, N. Anbazhagan, C. Park, G. P. Joshi, and C. Seo. A hash-based quantum-resistant designated verifier signature scheme. *Mathematics*, 10(10):1642, 2022.
35. H. Tian, X. Chen, Z. Jiang, and Y. Du. Non-delegatable strong designated verifier signature on elliptic curves. In H. Kim, editor, *Information Security and Cryptology - ICISC 2011 - 14th International Conference, Seoul, Korea, November 30 - December 2, 2011. Revised Selected Papers*, volume 7259 of *Lecture Notes in Computer Science*, pages 219–234. Springer, 2011.
36. H. Tian, X. Chen, and J. Li. A short non-delegatable strong designated verifier signature. In W. Susilo, Y. Mu, and J. Seberry, editors, *Information Security and Privacy - 17th Australasian Conference, ACISP 2012, Wollongong, NSW, Australia, July 9-11, 2012. Proceedings*, volume 7372 of *Lecture Notes in Computer Science*, pages 261–279. Springer, 2012.
37. H. Tian, Z. Jiang, Y. Liu, and B. Wei. A non-delegatable strong designated verifier signature without random oracles. In F. Xhafa, L. Barolli, F. Pop, X. Chen, and V. Cristea, editors, *2012 Fourth International Conference on Intelligent Networking and Collaborative Systems, INCoS 2012, Bucharest, Romania, September 19-21, 2012*, pages 237–244. IEEE, 2012.
38. R. Tso, T. Okamoto, and E. Okamoto. Practical strong designated verifier signature schemes based on double discrete logarithms. In D. Feng, D. Lin, and M. Yung, editors, *Information Security and Cryptology, First SKLOIS Conference, CISC 2005, Beijing, China, December 15-17, 2005, Proceedings*, volume 3822 of *Lecture Notes in Computer Science*, pages 113–127. Springer, 2005.
39. B. Wang and Z. Song. A non-interactive deniable authentication scheme based on designated verifier proofs. *Inf. Sci.*, 179(6):858–865, 2009.
40. B. Yang, Y. Sun, Y. Yu, and Q. Xia. A strong designated verifier signature scheme with secure disavowability. In F. Xhafa, L. Barolli, F. Pop, X. Chen, and V. Cristea, editors, *2012 Fourth International Conference on Intelligent Networking and Collaborative Systems, INCoS 2012, Bucharest, Romania, September 19-21, 2012*, pages 286–291. IEEE, 2012.
41. Y. Zhang, W. Susilo, and F. Guo. Lattice-based strong designated verifier signature with non-delegatability. *Comput. Stand. Interfaces*, 92:103904, 2025.