

Design and Analysis of Isogeny-Based Strong Designated Verifier Signature

Abhinav Sharma*

Indian Institute of Technology Hyderabad, India
abhisha8055@gmail.com

Vikas Srivastava†

Department of Mathematics
National Institute of Technology Warangal, India
vikas.math123@gmail.com, vsv@nitw.ac.in

Abstract

Strong designated-verifier signatures provide authentication while restricting verification to a chosen verifier and protecting the signer from transferable evidence. Designing such signatures in the post-quantum setting is challenging because authentication, signer privacy, simulation, and efficiency must be achieved simultaneously. Recently, Renan proposed CSI-SDVS, a compact post-quantum strong designated-verifier signature scheme built from CSIDH-style commutative isogeny class-group actions. We show that its response design, $z_i = b_i - s_i$, breaks privacy of the signer's identity: because the PSI experiment reveals both candidate signer secret keys, an adversary can reconstruct the signing randomness and identify the actual signer with overwhelming probability. We validate the attack over 30,000 executions, obtaining 100% signer identification in the main 128-bit experiment and for $\eta \in \{1, 2, 4, 8\}$. In the following, we propose an isogeny-based strong designated verifier signature. We prove correctness, non-transferability, signer privacy, and strong unforgeability under Gap Parallelization in the random-oracle model. For $\eta = 1$, the redesigned signature is 113 bytes compared with 49 bytes in CSI-SDVS, while signer key sizes remain unchanged.

Keywords: Isogeny-based Cryptography, Post-Quantum Cryptography, Strong Designated Verifier Signatures, Cryptanalysis, Quantum Security

1 Introduction

Digital signatures are a fundamental tool of modern cryptography and provide authentication, message integrity, and non-repudiation in applications such as secure communication, electronic transactions, software distribution, and digital identity systems. A conventional digital signature is publicly verifiable i.e. anyone possessing the signer's public key can verify whether a signature is valid. Although this property is essential in many applications, there are also situations in which a signer wishes to convince only a particular recipient and does not want the resulting signature

*This work was carried out as a part of a one-year remote research internship undertaken by Abhinav Sharma when he was doing his Masters at RIE Mysore under the supervision of Vikas Srivastava

†Corresponding author.

to be transferable as evidence to an arbitrary third party. This requirement motivates the notion of a *designated-verifier signature* (DVS), where a signature is intended for a particular verifier and the verifier can also generate indistinguishable transcripts, preventing a valid signature from serving as convincing evidence to an external party [1]. A stronger form, called a *strong designated-verifier signature* (SDVS), additionally aims to provide stronger privacy guarantees. In particular, some SDVS constructions consider *privacy of the signer’s identity* (PSI), which requires that an unauthorized observer should not be able to determine which of two possible signers generated a given signature [2].

The security of designated-verifier signatures becomes more important in the post-quantum setting. Many classical public-key constructions rely on integer factorization or discrete-logarithm-type assumptions, both of which can be efficiently solved by Shor’s quantum algorithm on a sufficiently powerful quantum computer [3]. This has motivated the development of post-quantum cryptography based on mathematical problems for which no efficient quantum algorithms are currently known. While post-quantum encryption, key encapsulation, and ordinary digital signatures have received considerable attention, relatively fewer works consider post-quantum designated-verifier signatures, where authentication, restricted verification, simulation, and signer privacy must be achieved simultaneously.

Isogeny-based cryptography provides one possible framework for constructing such protocols. In particular, commutative class-group actions, as used in CSIDH, offer an algebraic structure in which independently generated secret actions can be combined through commutativity [4]. These ideas have also been used in isogeny-based signature constructions such as CSI-FiSh [5]. Recently, Renan proposed CSI-SDVS, a compact post-quantum strong designated-verifier signature scheme based on such a commutative class-group action [6]. The construction is appealing because of its simple algebraic structure and compact signatures, and it claims, among other properties, privacy of the signer’s identity.

In this work, we revisit the signer-privacy property of CSI-SDVS and show that the PSI experiment defined for the scheme admits a simple polynomial-time attack. The weakness comes from the relation $z_i = b_i - s_i \pmod{N}$, where s_i is a signer secret and b_i is fresh signing randomness. In the PSI experiment, the adversary is given the secret keys of both candidate signers. Consequently, for a candidate signer S_j , it can compute $b_i^{(j)} = z_i + s_i^{(j)} \pmod{N}$. For the actual signer, this value is exactly the randomness used during signature generation. The adversary can then apply the recovered value to the public key of the designated verifier, reconstruct the curve that was used as input to the hash function, and test whether the resulting hash matches the challenge signature. In this way, the signer can be identified with overwhelming probability.

Motivated by this, we propose a modified isogeny-based SDVS in which the response is protected using a one-time masking value derived from an ephemeral class-group shared curve between the signer and the designated verifier.

Our contributions. The main contributions of this work are as follows:

1. We revisit the privacy guarantees of CSI-SDVS and identify a direct weakness in the way the response values are formed. In the PSI experiment, the adversary is given the secret keys of both candidate signers. Since each response has the form $z_i = b_i - s_i \pmod{N}$, the adversary can combine z_i with a candidate secret key and reconstruct the corresponding signing randomness. We show that this allows the adversary to recompute the hash input and identify the actual signer with overwhelming probability. The attack is polynomial-time and does not require solving GAIP, MT-GAIP, parallelization, or any other underlying isogeny problem.

2. We provide a formal analysis of the distinguisher and derive its success probability in the random-oracle model, including the negligible contributions arising from signer-key collisions and accidental hash collisions. We also implemented the attack. Across 30,000 challenge executions, including experiments for $\eta \in \{1, 2, 4, 8\}$, the signer was identified correctly in every tested instance.
3. In the following, we propose a new isogeny-based SDVS. Instead of publishing z_i directly, the proposed construction masks each response using a one-time value derived from an ephemeral class-group shared curve between the signer and the designated verifier. The verifier can recover this masking value using its secret key, whereas an outsider must solve a parallelization-type problem to obtain the corresponding shared curve. This prevents the direct reconstruction of the signing randomness from the candidate signer secret keys.
4. We prove correctness and non-transferability of the modified scheme and analyze signer privacy and strong unforgeability in the random-oracle model under an explicit Gap Parallelization assumption. We also evaluate the storage, communication, and computational overhead introduced by the modification. The signer key sizes remain unchanged, while the verifier stores one additional secret group element and one additional public curve. Each signature contains only one additional ephemeral curve, so the extra communication cost is independent of the number of parallel instances η .

2 Background and Preliminaries

This section introduces the concepts required to understand both our security analysis of CSI-SDVS and the modified construction proposed later in the paper. We first recall strong designated-verifier signatures and their security requirements. We then summarize the isogeny-based class-group-action framework and the computational assumptions used in the analysis.

2.1 Strong Designated-Verifier Signatures

A conventional digital-signature scheme consists of three main algorithms, namely **KeyGen**, **Sign**, and **Verify**. The signer uses a secret key to generate a signature on a message, while any party knowing the corresponding public key can verify it. This public verifiability is useful in many applications, but it is not appropriate when a signature is intended only for one particular recipient.

A designated-verifier signature (DVS) addresses this requirement [1]. A DVS involves a signer S and a designated verifier V . The signer generates a signature specifically for V , and the verification process requires information related to the verifier's secret key. Therefore, an arbitrary third party cannot independently verify the signature in the same way as the designated verifier. A DVS also provides a simulation property: the designated verifier can generate transcripts that are indistinguishable from genuine signatures. Consequently, a valid transcript cannot be transferred to another party as convincing evidence that the signer actually produced it.

A strong designated-verifier signature (SDVS) strengthens this setting because here, verification requires the designated verifier's secret key because of which no other party can verify it, and the identity of the signer is also hidden from every party except the designated verifier. A typical SDVS scheme consists of **Setup**, signer and verifier key-generation algorithms, **Sign**, **Verify**, and a simulation algorithm **Sim**. If (sk_S, pk_S) and (sk_V, pk_V) denote the signer and verifier key pairs, respectively, then a genuine signature is generated as $\sigma \leftarrow \text{Sign}(sk_S, pk_V, m)$ and checked using $\text{Verify}(pk_S, sk_V, m, \sigma)$. The verifier can additionally generate a simulated transcript as $\tilde{\sigma} \leftarrow \text{Sim}(pk_S, sk_V, m)$.

The security properties considered in this paper are correctness, strong unforgeability, non-transferability, and privacy of the signer’s identity. Correctness requires that every honestly generated signature is accepted by the designated verifier except with negligible probability. Strong unforgeability under chosen-message attacks (SUF-CMA) requires that an adversary cannot produce a fresh valid signature even after obtaining signatures on messages of its choice. In the strong notion, producing a different valid signature for an already signed message also counts as a forgery.

Non-transferability requires that genuine and simulated transcripts be identical or computationally indistinguishable. Hence, an external party should not be able to determine whether a valid transcript was generated by the signer or simulated by the designated verifier. This prevents the verifier from using the signature as transferable evidence.

Some SDVS constructions additionally provide privacy of the signer’s identity (PSI) [2]. A typical PSI experiment considers two candidate signers S_0 and S_1 . The challenger selects a random bit $\beta \in \{0, 1\}$ and generates a challenge signature using S_β . The adversary outputs a guess β' , and its advantage is measured as

$$\text{Adv}_{\text{PSI}} = \left| \Pr[\beta' = \beta] - \frac{1}{2} \right|.$$

A secure scheme requires this advantage to be negligible. The exact information given to the adversary is important. In particular, the PSI experiment used by CSI-SDVS gives the adversary the secret keys of both candidate signers while keeping the verifier’s secret key hidden. This feature becomes central in our cryptanalysis.

2.2 Isogeny-Based Group Actions and Security Assumptions

CSI-SDVS is based on a commutative class-group action over elliptic curves, following the general framework used in CSIDH [4]. Only a small amount of isogeny theory is required for our analysis. An elliptic curve over a finite field $\mathbb{F}_q$ can, under suitable conditions, be represented by an equation of the form $E : y^2 = x^3 + Ax + B$. An isogeny $\phi : E \rightarrow E'$ is a non-constant rational map between elliptic curves that preserves the identity element. Isogeny-based cryptography derives computational problems from the difficulty of recovering hidden relationships between suitable curves.

Let $\mathcal{G}$ be a finite abelian group acting on a set $\mathcal{X}$ of suitable elliptic curves. For $g \in \mathcal{G}$ and $E \in \mathcal{X}$, we write $g * E$ for the result of the group action. The action satisfies $1 * E = E$ and $(g_1 g_2) * E = g_1 * (g_2 * E)$. Since $\mathcal{G}$ is abelian, the actions commute.

CSI-SDVS represents group elements using exponents. Let E_0 be the public base curve. For $a \in \mathbb{Z}_N$, the notation $[a]E$ denotes the action of the corresponding group element on E . The main algebraic identity is

$$[a][b]E = [a + b]E. \tag{1}$$

Thus, $[a]([b]E_0) = [a + b]E_0$, and the order in which the two known actions are applied does not matter.

We work with the cyclic class-group representation used by CSI-SDVS. Fix a generator $\mathbf{g}$ of order N , and let $[a]E$ denote the action of $\mathbf{g}^a$. This action is free and transitive on the relevant set of curve isomorphism classes, and every class has a fixed canonical encoding. Consequently, $[a]E = [b]E$ holds exactly when $a = b$ in $\mathbb{Z}_N$. This freeness property is the hypothesis used below to bound the success probability of the privacy attack, and it must be stated explicitly rather than assumed for an arbitrary abelian action.

A secret key is represented by an exponent $s \in \mathbb{Z}_N$, while the corresponding public key is $E_s = [s]E_0$. When the exponent a is known, computing $[a]E$ is an efficient public operation. The

assumed hard direction is to recover an unknown action from two related curves. The Group Action Inverse Problem (GAIP) asks, given E and $E' = [a]E$, to recover an action element mapping E to E' . The Multi-Target Group Action Inverse Problem (MT-GAIP) considers several related instances simultaneously [7, 6].

A second important problem is parallelization [7, 4]. Given public curves $E_a = [a]E_0$ and $E_b = [b]E_0$, where a and b are unknown, the goal is to compute

$$[a + b]E_0. \quad (2)$$

By commutativity, this value can also be written as $[a]E_b = [b]E_a$. If GAIP can be solved, then parallelization can also be solved by recovering one of the hidden actions. The converse does not automatically follow, since an algorithm might compute the shared curve without recovering either exponent. This distinction is important in our analysis of the unforgeability proof of CSI-SDVS.

For signer secret s and verifier secret v , let $E_S = [s]E_0$ and $E_V = [v]E_0$. Their combined action defines the shared curve

$$K = [s + v]E_0. \quad (3)$$

Using commutativity, $K = [s]E_V = [v]E_S$. Hence, if either s or v is known, K can be efficiently computed from the other party's public key. A party that knows neither secret must solve a parallelization-type problem to compute K . For the parallel form of CSI-SDVS, we similarly write $K_i = [s_i + v_i]E_0$.

CSI-SDVS also uses a cryptographic hash function $H : \{0, 1\}^* \rightarrow \{0, 1\}^\lambda$ and analyzes security in the random-oracle model. In this model, every previously unseen query receives an independent random output, while repeated queries return the same value. Thus, if a signer computes $h = H(Y_1 \| \dots \| Y_\eta \| m)$, then an adversary that reconstructs the same values Y_i can recompute the same hash value.

The cryptanalysis presented later does not solve GAIP, MT-GAIP, or parallelization. It uses only the public evaluation of known group actions together with the information explicitly provided by the PSI experiment. Nevertheless, the above hard problems are needed to understand the claimed SUF-CMA proof of CSI-SDVS and the security assumptions used by our modified construction.

3 The CSI-SDVS Scheme

We now describe the CSI-SDVS scheme proposed by Renan [6]. The construction uses the commutative class-group action underlying CSIDH and the cyclic representation adopted in CSI-FiSh [4, 5]. Let E_0 be the public base curve, let N denote the order of the cyclic class group, and let $[a]E$ denote the action of $a \in \mathbb{Z}_N$ on a curve E . The construction uses η parallel instances and relies on the relation $[a][b]E = [a + b]E$. It consists of setup, signer key generation, verifier key generation, signing, verification, and simulation algorithms, summarized in Algorithm 1. Correctness follows immediately from the commutative action. For a genuine signature, $z_i = b_i - s_i$ and $E_i = [s_i]E_0$, so the verifier reconstructs

$$Y'_i = [v_i + z_i]E_i = [v_i + b_i - s_i][s_i]E_0 = [v_i + b_i]E_0 = [b_i]\hat{E}_i = Y_i.$$

Hence the verifier obtains the same hash input as the signer. Similarly, for a simulated signature, $z_i = r_i - v_i$, and therefore $[v_i + z_i]E_i = [r_i]E_i = Y_i$. Thus, genuine and simulated signatures have the same format and are accepted by the same verification algorithm.

For later analysis, it is useful to define the shared curve $K_i = [s_i + v_i]E_0$. By commutativity, $K_i = [s_i]\hat{E}_i = [v_i]E_i$. Since $z_i = b_i - s_i$, a genuine signature satisfies $Y_i = [z_i]K_i$, and therefore its

Algorithm 1 CSI-SDVS scheme [6]

```
1: procedure SETUP( $1^\lambda$ )
2:   Choose CSIDH parameters  $p$ ,  $E_0$ , cyclic class group of order  $N$ , generator  $\mathbf{g}$ , hash function  $\mathcal{H}$ , and number of parallel instances  $\eta$ .
3:   return  $\text{pp} = (p, \mathbf{g}, N, E_0, \mathcal{H}, \eta)$ .
4: end procedure
5: procedure SIGKEYGEN( $\text{pp}$ )
6:   for  $i = 1, \dots, \eta$  do
7:     Sample  $s_i \leftarrow \mathbb{Z}_N$  and compute  $E_i = [s_i]E_0$ .
8:   end for
9:   return  $\text{sk}_S = (s_1, \dots, s_\eta)$  and  $\text{pk}_S = (E_1, \dots, E_\eta)$ .
10: end procedure
11: procedure VERKEYGEN( $\text{pp}$ )
12:   for  $i = 1, \dots, \eta$  do
13:     Sample  $v_i \leftarrow \mathbb{Z}_N$  and compute  $\hat{E}_i = [v_i]E_0$ .
14:   end for
15:   return  $\text{sk}_V = (v_1, \dots, v_\eta)$  and  $\text{pk}_V = (\hat{E}_1, \dots, \hat{E}_\eta)$ .
16: end procedure
17: procedure SIGN( $m, \text{sk}_S, \text{pk}_V$ )
18:   for  $i = 1, \dots, \eta$  do
19:     Sample  $b_i \leftarrow \mathbb{Z}_N$ .
20:     Compute  $Y_i = [b_i]\hat{E}_i$  and  $z_i = b_i - s_i \pmod{N}$ .
21:   end for
22:   Compute  $h = \mathcal{H}(Y_1 \parallel \dots \parallel Y_\eta \parallel m)$ .
23:   return  $\sigma = (h, z_1, \dots, z_\eta)$ .
24: end procedure
25: procedure VERIFY( $m, \sigma, \text{pk}_S, \text{sk}_V$ )
26:   Parse  $\sigma = (h, z_1, \dots, z_\eta)$ .
27:   for  $i = 1, \dots, \eta$  do
28:     Compute  $Y'_i = [v_i + z_i]E_i$ .
29:   end for
30:   Compute  $h' = \mathcal{H}(Y'_1 \parallel \dots \parallel Y'_\eta \parallel m)$ .
31:   return 1 if  $h' = h$ ; otherwise return 0.
32: end procedure
33: procedure SIMUL( $m, \text{pk}_S, \text{sk}_V$ )
34:   for  $i = 1, \dots, \eta$  do
35:     Sample  $r_i \leftarrow \mathbb{Z}_N$ .
36:     Compute  $Y_i = [r_i]E_i$  and  $z_i = r_i - v_i \pmod{N}$ .
37:   end for
38:   Compute  $h = \mathcal{H}(Y_1 \parallel \dots \parallel Y_\eta \parallel m)$ .
39:   return  $\sigma = (h, z_1, \dots, z_\eta)$ .
40: end procedure
```

hash can equivalently be written as

$$h = \mathcal{H}([z_1]K_1 \parallel \cdots \parallel [z_\eta]K_\eta \parallel m).$$

The same representation also holds for simulated signatures. This equivalent form follows directly from the original CSI-SDVS equations and will be useful in our security analysis. Renan claims that CSI-SDVS provides strong unforgeability under chosen-message attacks, non-transferability, and privacy of the signer’s identity in the random-oracle model [6]. In the next section, we analyze these claims and show that the stated PSI property admits a direct polynomial-time attack.

4 Cryptanalysis of CSI-SDVS

CSI-SDVS claims strong unforgeability under chosen-message attacks (SUF-CMA), non-transferability, and privacy of the signer’s identity (PSI) [6]. In this section, we analyze these security claims. We first present a direct attack on PSI. We then explain why the published PSI proof does not rule out this attack. Our analysis finds a concrete break of PSI.

4.1 Attack on Signer Privacy

The PSI experiment of CSI-SDVS considers two possible signers S_0 and S_1 . Let their secret keys be $\mathbf{s}^{(0)} = (s_1^{(0)}, \dots, s_\eta^{(0)})$ and $\mathbf{s}^{(1)} = (s_1^{(1)}, \dots, s_\eta^{(1)})$. The designated verifier has secret key $\mathbf{v} = (v_1, \dots, v_\eta)$ and public key $\hat{E}_i = [v_i]E_0$ for each i . An important feature of the PSI experiment is that the adversary is given the secret keys of both candidate signers, while the verifier’s secret key remains hidden [6]. The challenger selects a random bit $\beta \in \{0, 1\}$ and generates a challenge signature $\sigma^* = (h^*, \mathbf{z}^*)$ using signer S_β .

During signature generation, the signer chooses random values b_i and computes $z_i^* = b_i - s_i^{(\beta)} \pmod{N}$. Since the adversary knows the secret key of each candidate signer, it can test a candidate S_j by computing $\tilde{b}_i^{(j)} = z_i^* + s_i^{(j)} \pmod{N}$. It then evaluates $\tilde{Y}_i^{(j)} = [\tilde{b}_i^{(j)}]\hat{E}_i$ and computes

$$\tilde{h}_j = \mathcal{H}(\tilde{Y}_1^{(j)} \parallel \cdots \parallel \tilde{Y}_\eta^{(j)} \parallel m^*).$$

If exactly one candidate hash equals h^* , the adversary outputs that candidate’s index. If both candidate hashes match, or if neither matches, it outputs a uniform random bit. For an honestly generated challenge under the specified algorithms, the correct candidate always matches, so the “neither matches” case does not arise, and the “both match” case arises only through identical candidate keys or a hash collision. For the actual signer S_β , we have $\tilde{b}_i^{(\beta)} = z_i^* + s_i^{(\beta)} = b_i$. Therefore, $\tilde{Y}_i^{(\beta)} = [b_i]\hat{E}_i = Y_i$, which is exactly the curve used by the signer in the challenge hash. Hence $\tilde{h}_\beta = h^*$. For the other candidate, the reconstructed curve is

$$\tilde{Y}_i^{(1-\beta)} = [b_i - s_i^{(\beta)} + s_i^{(1-\beta)}]\hat{E}_i.$$

Except when the two candidate secret-key components coincide, this curve differs from the genuine curve. Since the action is free, equality for all coordinates requires $s_i^{(0)} = s_i^{(1)}$ for all i , which occurs with probability $N^{-\eta}$ for independently generated keys. Otherwise, the two hash inputs are different, and a false match occurs only with the negligible probability of a hash collision.

Theorem 1. *In the random-oracle model, with independent uniform signer keys and the free cyclic action representation of Section 2, CSI-SDVS does not satisfy the PSI security definition stated in*

[6]. The adversary of the previous subsection uses at most 2η known group-action evaluations and two signature-hash queries, makes no signing, simulation, or verification queries, and succeeds with probability

$$\Pr[\beta' = \beta] = 1 - \frac{d}{2}, \quad \text{where } d = N^{-\eta} + (1 - N^{-\eta})2^{-\lambda}.$$

Its PSI advantage is therefore $(1 - N^{-\eta})(1 - 2^{-\lambda})/2$, which is overwhelming rather than negligible.

Proof. For the correct signer, the adversary reconstructs all the original signing values b_i and therefore exactly reproduces the hash input. Hence the correct candidate always satisfies $\tilde{h}_\beta = h^*$. For the wrong signer, all reconstructed curves coincide with the genuine curves only when the two signer secret keys are identical in every coordinate, an event of probability $N^{-\eta}$. Otherwise, a false acceptance can occur only through a hash collision. Thus, up to negligible terms, the adversary identifies the signer with probability close to one. Therefore its PSI advantage is close to $1/2$, rather than negligible. $\square$

The PSI proof in CSI-SDVS considers the value $X_i^{(j)} = [z_i]E_i^{(j)}$ and argues that an adversary cannot compute $[v_i]X_i^{(j)}$ because v_i is secret [6]. Since $E_i^{(j)} = [s_i^{(j)}]E_0$, we have

$$[v_i]X_i^{(j)} = [v_i][z_i][s_i^{(j)}]E_0 = [v_i + z_i + s_i^{(j)}]E_0.$$

The same curve can be computed as

$$[z_i + s_i^{(j)}]\hat{E}_i = [z_i + s_i^{(j)}][v_i]E_0 = [z_i + s_i^{(j)} + v_i]E_0.$$

Therefore,

$$[v_i]X_i^{(j)} = [z_i + s_i^{(j)}]\hat{E}_i.$$

The expression on the right is completely known to the adversary: z_i is contained in the signature, $s_i^{(j)}$ is supplied in the PSI experiment, and $\hat{E}_i$ is public. Thus the verifier secret v_i is not needed. The proof considers only the route that applies the unknown action $[v_i]$ to $X_i^{(j)}$, but commutativity provides another route in which a known action is applied to the verifier's public key.

4.2 Experimental Validation of the PSI Attack

To validate the signer-privacy attack, we implemented CSI-SDVS together with the proposed PSI distinguisher. The purpose of the experiment is to confirm the algebraic attack described above. The implementation uses an abstract hard homogeneous space in which the cyclic group $\mathbb{Z}_N$ acts freely and transitively on an opaque set of “curves.” The action is represented internally as $[a]E_b = E_{a+b \bmod N}$, while the distinguisher accesses curves only through the public group-action interface. It never inverts the group action and never uses the verifier's secret key. This abstraction preserves the compatibility, commutativity, and freeness properties used by CSI-SDVS and by our attack, while avoiding the unnecessary cost of implementing complete CSIDH isogeny walks. For every challenge signature $\sigma = (h, \mathbf{z})$, the implementation tests the two candidate signers exactly as described in Section 4. For candidate S_j , it computes $\tilde{b}_i^{(j)} = z_i + s_i^{(j)} \pmod{N}$, evaluates $\tilde{Y}_i^{(j)} = [\tilde{b}_i^{(j)}]\hat{E}_i$, and checks whether

$$\mathcal{H}(\tilde{Y}_1^{(j)} \parallel \dots \parallel \tilde{Y}_\eta^{(j)} \parallel m) = h.$$

Table 1: Experimental validation of the PSI distinguisher.

Experiment	η	λ	Trials	Success
Main experiment	1	128	10,000	100.00%
Parallel instances	1	64	5,000	100.00%
Parallel instances	2	64	5,000	100.00%
Parallel instances	4	64	5,000	100.00%
Parallel instances	8	64	5,000	100.00%

For the actual signer, the reconstructed value satisfies $\tilde{b}_i^{(\beta)} = b_i$, and hence the original hash input is reproduced exactly. The theoretical success probability is therefore

$$\Pr[\beta' = \beta] = 1 - \frac{1}{2} \left(N^{-\eta} + (1 - N^{-\eta})2^{-\lambda} \right),$$

which is overwhelmingly close to one for cryptographic parameters. Our main experiment used $\eta = 1$, $N = 2^{127} - 1$, a 128-bit hash output, and 10,000 independently generated challenge signatures. The distinguisher identified the correct signer in all 10,000 trials. We additionally studied the effect of the number of parallel instances by testing $\eta \in \{1, 2, 4, 8\}$ with 5,000 trials for each value. The correct signer was identified in every trial for all four values of η .

Overall, the reported experiments contain 30,000 challenge executions. The results agree with the theoretical analysis: the signer was identified in every tested instance, and increasing the number of parallel components did not mitigate the attack. It only increases the distinguisher’s computational cost linearly with η . Since the attack is already established algebraically by Theorem 1, these experiments should be viewed as implementation-level validation rather than as the basis of the security claim. They confirm that the PSI weakness follows directly from the exposed relation $z_i = b_i - s_i$ and does not depend on any weakness of a particular isogeny implementation.

5 Proposed CSI-SDVS Design

The cryptanalysis in the previous section shows that the main privacy weakness of CSI-SDVS comes from the response value $z_i = b_i - s_i$. In the PSI experiment, the adversary knows the secret key $s_i^{(j)}$ of each candidate signer. Hence, from a public value z_i , it can recover a candidate signing randomness $b_i^{(j)} = z_i + s_i^{(j)}$ and test which candidate reproduces the hash contained in the signature. This observation suggests a simple design principle: the response z_i should not be available in the clear. At the same time, the designated verifier must be able to recover it in order to perform the original verification procedure. We achieve this by masking every response value with a one-time pseudorandom value derived from an ephemeral class-group shared secret. The shared secret can be computed by the signer using the verifier’s public key and by the verifier using its secret key, while an outsider is required to solve a parallelization-type problem to obtain it. The resulting construction preserves the main algebraic structure of CSI-SDVS while removing the direct signer-identification mechanism identified in Section 4.

5.1 Modified Construction

The public parameters are the same as in CSI-SDVS, together with three domain-separated hash functions. We use $\mathcal{H}$ for the signature hash, $\mathcal{H}_K$ for key derivation, and $\mathcal{H}_M$ for deriving masking values in $\mathbb{Z}_N$. The signer key generation algorithm remains unchanged. For every $i \in [\eta]$, the signer

chooses $s_i \leftarrow \mathbb{Z}_N$ and computes $E_i = [s_i]E_0$. Thus, $\mathbf{sk}_S = (s_1, \dots, s_\eta)$ and $\mathbf{pk}_S = (E_1, \dots, E_\eta)$. The verifier also generates the original CSI-SDVS verification key. It chooses $v_i \leftarrow \mathbb{Z}_N$ and computes $\hat{E}_i = [v_i]E_0$ for every i . In addition, it generates one independent masking key by choosing $u \leftarrow \mathbb{Z}_N$ and computing $U = [u]E_0$. The verifier secret and public keys are therefore

$$\mathbf{sk}_V = (v_1, \dots, v_\eta, u), \quad \mathbf{pk}_V = (\hat{E}_1, \dots, \hat{E}_\eta, U).$$

The additional pair (u, U) is used only to protect the response values. Keeping it separate from the verification exponents v_i simplifies both the construction and its security analysis.

Signature generation. To sign a message m , the signer first creates an ephemeral class-group value. It chooses $e \leftarrow \mathbb{Z}_N$, computes $C = [e]E_0$, and evaluates the shared curve $Q = [e]U$. Since $U = [u]E_0$, we have $Q = [e + u]E_0$. The signer then derives a temporary masking key

$$k = \mathcal{H}_K(C \| Q \| m \| \mathbf{pk}_S).$$

The signer next performs the original CSI-SDVS signing computation. For every i , it chooses $b_i \leftarrow \mathbb{Z}_N$, computes $Y_i = [b_i]\hat{E}_i$, and forms the ordinary response $z_i = b_i - s_i \pmod{N}$. Instead of including z_i directly in the signature, the signer derives $\rho_i = \mathcal{H}_M(k \| i \| C \| m) \in \mathbb{Z}_N$ and computes the masked response.

$$\bar{z}_i = z_i + \rho_i \pmod{N}.$$

Finally, it computes

$$h = \mathcal{H}(Y_1 \| \dots \| Y_\eta \| m \| C)$$

and outputs

$$\sigma = (C, h, \bar{z}_1, \dots, \bar{z}_\eta).$$

Including C in the hash binds the ephemeral masking component to the signature and prevents it from being replaced independently of the remaining transcript.

Verification. Given $\sigma = (C, h, \bar{z}_1, \dots, \bar{z}_\eta)$, the designated verifier first reconstructs the ephemeral shared curve using its secret value u . It computes $Q' = [u]C$. Since $C = [e]E_0$, commutativity gives $Q' = [u + e]E_0 = Q$. The verifier derives $k = \mathcal{H}_K(C \| Q' \| m \| \mathbf{pk}_S)$, using the signer public key $\mathbf{pk}_S$ supplied to VERIFY, and recomputes $\rho_i = \mathcal{H}_M(k \| i \| C \| m)$ for every i . It then removes the mask by computing $z_i = \bar{z}_i - \rho_i \pmod{N}$. After recovering the original responses, verification proceeds exactly as in CSI-SDVS. The verifier computes $Y'_i = [v_i + z_i]E_i$ and accepts if

$$h = \mathcal{H}(Y'_1 \| \dots \| Y'_\eta \| m \| C).$$

Simulation. The designated verifier must also be able to generate a valid transcript without knowing the signer's secret key. To simulate a signature, it first chooses $e \leftarrow \mathbb{Z}_N$, computes $C = [e]E_0$ and $Q = [u]C$, and derives $k = \mathcal{H}_K(C \| Q \| m \| \mathbf{pk}_S)$, using the public key $\mathbf{pk}_S$ of the signer being simulated. For each i , it chooses $r_i \leftarrow \mathbb{Z}_N$, computes $Y_i = [r_i]E_i$, and sets $z_i = r_i - v_i \pmod{N}$, exactly as in the original CSI-SDVS simulation algorithm. It then derives $\rho_i = \mathcal{H}_M(k \| i \| C \| m)$ and outputs the masked value $\bar{z}_i = z_i + \rho_i \pmod{N}$. Finally, it computes $h = \mathcal{H}(Y_1 \| \dots \| Y_\eta \| m \| C)$ and returns $\sigma = (C, h, \bar{z}_1, \dots, \bar{z}_\eta)$.

5.2 Correctness and Design Rationale

Correctness follows directly from the commutative action. The signer computes $Q = [e]U = [e+u]E_0$, while the verifier computes $Q' = [u]C = [u+e]E_0$. Hence $Q = Q'$, so both parties derive the same temporary key k and the same masking values ρ_i . The verifier therefore recovers the original response z_i exactly. Using $z_i = b_i - s_i$ and $E_i = [s_i]E_0$, the verifier then obtains

$$[v_i + z_i]E_i = [v_i + b_i - s_i][s_i]E_0 = [v_i + b_i]E_0 = [b_i]\hat{E}_i = Y_i.$$

Thus the reconstructed hash input is identical to the one used during signing, and an honestly generated signature is always accepted. The important difference from CSI-SDVS is that the value z_i is no longer visible to an outsider. Consider again the PSI adversary that knows both candidate signer secret keys. In the original scheme, it could compute $b_i^{(j)} = z_i + s_i^{(j)}$ and immediately test candidate j . In the modified construction, the adversary sees only $\bar{z}_i = z_i + \rho_i$. To recover z_i , it must first obtain ρ_i , which requires the temporary key k .

The key k is derived from the shared curve $Q = [e+u]E_0$. An outsider is given only $C = [e]E_0$ and $U = [u]E_0$. Computing Q from these two public curves is exactly a parallelization-type computation. Therefore, under the assumed hardness of parallelization and the pseudorandomness of the key-derivation hash, the values ρ_i are hidden from an outsider. This removes the direct PSI attack. In particular, knowledge of a candidate signer secret $s_i^{(j)}$ is no longer sufficient to reconstruct the signing randomness because

$$b_i^{(j)} = \bar{z}_i - \rho_i + s_i^{(j)} \pmod{N}$$

still contains the unknown mask ρ_i . The hash value h alone also does not reveal the signer. For fresh uniformly sampled b_i , each $Y_i = [b_i]\hat{E}_i$ is distributed independently of the long-term signer secret s_i . In the original construction, signer information became visible only through the correlation between h and the exposed response z_i . Masking z_i removes this correlation from the adversary's view. The simulation property is preserved as well. Before masking, the genuine and simulated CSI-SDVS responses have the same algebraic verification relation. Both are subsequently shifted by the same type of pseudorandom mask derived from an independently generated ephemeral shared secret. Consequently, the designated verifier can still create valid transcripts without knowing the signer's secret key. The modified construction introduces only one additional ephemeral curve C in each signature and one additional class-group key U in the verifier public key. All η response components are protected using a single ephemeral shared secret, so no additional class-group encapsulation is required for each coordinate.

The next section gives a formal security analysis of the modified construction. In particular, we show that the signer-identification attack against CSI-SDVS no longer applies and relate signer privacy to the hardness of computing the ephemeral parallelization value $[e+u]E_0$.

6 Security Analysis of the Modified Scheme

We now analyze the security of the modified construction. We consider the same correctness, non-transferability, PSI, and SUF-CMA requirements used for CSI-SDVS [6]. The hash functions $\mathcal{H}$, $\mathcal{H}_K$, and $\mathcal{H}_M$ are modeled as independent random oracles. For the formal analysis, we use the following version of the parallelization assumption. Let $X = [x]E_0$ and $Y = [y]E_0$. The parallelization problem is to compute $[x+y]E_0$ from (E_0, X, Y) . We additionally consider a decision oracle

$$\mathcal{D}_{\text{Par}}(X, Y, Z) = \begin{cases} 1, & \text{if } Z = [x + y]E_0, \\ 0, & \text{otherwise.} \end{cases}$$

The *Gap Parallelization Problem* asks an adversary to compute $[x + y]E_0$ even when it is allowed to use $\mathcal{D}_{\text{Par}}$. We assume that this problem remains computationally hard for the underlying class-group action. This assumption is stronger than ordinary parallelization hardness, but it allows us to handle the simulation and verification oracles appearing in the SDVS security experiments without making an implicit assumption inside the proof.

Theorem 2 (Correctness). *The modified CSI-SDVS construction is correct. Every honestly generated signature, as well as every honestly simulated signature, is accepted by the designated verifier.*

Proof. Let the signer choose an ephemeral value e and compute $C = [e]E_0$ and $Q = [e]U$, where $U = [u]E_0$ is the verifier's masking public key. The designated verifier computes $Q' = [u]C$. By commutativity of the class-group action,

$$Q' = [u][e]E_0 = [u + e]E_0 = [e][u]E_0 = Q.$$

Therefore, the signer and verifier obtain the same temporary key $k = \mathcal{H}_K(C \| Q \| m \| \text{pk}_S)$ and consequently the same masking values $\rho_i = \mathcal{H}_M(k \| i \| C \| m)$, since both parties use the same signer public key pk_S . Hence, from $\bar{z}_i = z_i + \rho_i$, the verifier correctly recovers $z_i = \bar{z}_i - \rho_i$. For a genuine signature, $z_i = b_i - s_i$. Since $E_i = [s_i]E_0$, the verifier computes

$$[v_i + z_i]E_i = [v_i + b_i - s_i][s_i]E_0 = [v_i + b_i]E_0 = [b_i]\hat{E}_i = Y_i.$$

Thus, the verifier reconstructs exactly the curves used by the signer in the hash $\mathcal{H}(Y_1 \| \dots \| Y_\eta \| m \| C)$ and accepts. For a simulated signature, $z_i = r_i - v_i$. Therefore,

$$[v_i + z_i]E_i = [r_i]E_i = Y_i,$$

which is exactly the curve used by the simulation algorithm. Hence simulated signatures are also accepted. $\square$

Theorem 3 (Non-Transferability). *Real and simulated signatures in the modified construction have identical distributions. Consequently, the scheme satisfies non-transferability.*

Proof. Consider first a genuine signature. Since b_i is chosen uniformly from $\mathbb{Z}_N$ and $z_i = b_i - s_i$, the value z_i is also uniformly distributed over $\mathbb{Z}_N$. Define the shared signer-verifier curve $K_i = [s_i + v_i]E_0$. The curve hashed in a genuine signature can then be written as

$$Y_i = [b_i]\hat{E}_i = [b_i + v_i]E_0 = [z_i + s_i + v_i]E_0 = [z_i]K_i.$$

Now consider a simulated signature. The simulator chooses r_i uniformly and sets $z_i = r_i - v_i$, so z_i is again uniform over $\mathbb{Z}_N$. Moreover,

$$Y_i = [r_i]E_i = [r_i + s_i]E_0 = [z_i + v_i + s_i]E_0 = [z_i]K_i.$$

Thus, before masking, both genuine and simulated signatures are distributed as a uniform z_i together with the same curve $Y_i = [z_i]K_i$. The ephemeral value e , the curve C , the shared masking curve Q , and the masks ρ_i are generated according to the same distribution in both algorithms. In both cases, the transmitted response is $\bar{z}_i = z_i + \rho_i$. Therefore, the complete tuples $(C, h, \bar{z}_1, \dots, \bar{z}_\eta)$

produced by the signing and simulation algorithms have identical distributions. Hence, an external party cannot determine whether a valid transcript was generated by the signer or by the designated verifier. $\square$

Theorem 4 (Privacy of the Signer’s Identity). *Assume that the Gap Parallelization Problem is hard for the underlying class-group action and that $\mathcal{H}_K$ and $\mathcal{H}_M$ are random oracles. Then, with the signer-bound key derivation $k = \mathcal{H}_K(C \| Q \| m \| \text{pk}_S)$ of Section 5, the modified construction satisfies privacy of the signer’s identity in the PSI experiment, including verification queries made under either candidate key both before and after the challenge.*

Proof. Consider the two candidate signers S_0 and S_1 . As in the CSI-SDVS PSI experiment, the adversary is given both signer secret keys $\mathbf{s}^{(0)}$ and $\mathbf{s}^{(1)}$, their corresponding public keys, and the verifier public key. The verifier secrets $\mathbf{v}$ and u remain hidden. Let the challenge signature be $\sigma^* = (C^*, h^*, \bar{\mathbf{z}}^*)$, generated by signer S_β for a random bit β . Write $C^* = [e^*]E_0$. The masking shared curve is

$$Q^* = [e^*]U = [e^* + u]E_0.$$

Suppose first that the adversary does not obtain Q^* . Then it cannot query $\mathcal{H}_K$ at the correct input $(C^*, Q^*, m^*, \text{pk}_S^{(\beta)})$, except by guessing the missing parallelization value. Consequently, the key $k^* = \mathcal{H}_K(C^* \| Q^* \| m^* \| \text{pk}_S^{(\beta)})$ is a fresh random value from the adversary’s point of view. Each corresponding mask $\rho_i^* = \mathcal{H}_M(k^* \| i \| C^* \| m^*)$ is therefore independent and uniformly distributed in $\mathbb{Z}_N$. This holds regardless of when the adversary queries the verification oracle: since k^* is keyed to the specific pair $(Q^*, \text{pk}_S^{(\beta)})$, a query submitted under the other candidate key $\text{pk}_S^{(1-\beta)}$ — whether made before or after the challenge, and whether or not it reuses $\bar{\mathbf{z}}^*$ — addresses the independently random point $k' = \mathcal{H}_K(C^* \| Q^* \| m^* \| \text{pk}_S^{(1-\beta)})$ unless that exact point has itself been queried, in which case the reduction below extracts the Gap Parallelization solution from it exactly as it would from a query at $(C^*, Q^*, m^*, \text{pk}_S^{(\beta)})$, because $\mathcal{D}_{\text{Par}}$ is evaluated only on (C, U, Q) and does not depend on which pk_S accompanies the query. Consequently the argument below covers verification queries under either candidate key without a separate case. For the challenge signer,

$$\bar{z}_i^* = b_i - s_i^{(\beta)} + \rho_i^* \pmod{N}.$$

Since ρ_i^* is uniform and hidden, $\bar{z}_i^*$ is uniform independently of b_i and independently of $s_i^{(\beta)}$. In particular, even though the adversary knows both candidate signing secrets, it cannot recover b_i from $\bar{z}_i^*$. The remaining challenge value is

$$h^* = \mathcal{H}(Y_1^* \| \dots \| Y_\eta^* \| m^* \| C^*),$$

where $Y_i^* = [b_i]\widehat{E}_i$. The values b_i are fresh uniform randomness and are chosen independently of the identity bit β . Hence, once the correlation through z_i is hidden by the random masks, the challenge transcript has the same distribution for S_0 and S_1 .

It remains to consider the possibility that the adversary learns the correct masking value. To obtain Q^* from $C^* = [e^*]E_0$ and $U = [u]E_0$, it must compute $[e^* + u]E_0$, which is exactly a parallelization instance. More formally, suppose there exists a PPT adversary $\mathcal{A}$ that distinguishes the challenge signer with non-negligible advantage. We construct an algorithm $\mathcal{B}$ for Gap Parallelization. The algorithm $\mathcal{B}$ receives (C^*, U) and must compute their parallelization. It generates all signer keys and the verification exponents v_i itself and uses U as the masking public key. For ordinary simulation queries, $\mathcal{B}$ chooses the ephemeral exponent e itself and can therefore compute the corresponding

shared curve as $[e]U$ without knowing u . Verification queries are simulated using the random-oracle tables together with $\mathcal{D}_{\text{Par}}$, which allows $\mathcal{B}$ to recognize whether a candidate curve supplied in an $\mathcal{H}_K$ query is the correct parallelization of C and U .

For the challenge, $\mathcal{B}$ uses C^* and initially replaces k^* with an independent random value. If $\mathcal{A}$ ever queries $\mathcal{H}_K$ on $(C^*, Q, m^*, \text{pk}_S^{(j)})$, for either candidate $j \in \{0, 1\}$, and $\mathcal{D}_{\text{Par}}(C^*, U, Q) = 1$, then $Q = Q^*$ and $\mathcal{B}$ immediately obtains the required solution to its Gap Parallelization instance, regardless of which candidate's key accompanied the query. If no such query occurs, the challenge masking key remains information-theoretically independent of $\mathcal{A}$'s view, and the challenge signature is independent of β . Thus $\mathcal{A}$ can do no better than random guessing, apart from negligible random-oracle guessing probabilities. Therefore, a non-negligible PSI advantage would imply a non-negligible algorithm for Gap Parallelization, contradicting the assumption. $\square$

Remark 1 (Why the key derivation must bind pk_S). A variant of the construction that derives the masking key as $k = \mathcal{H}_K(C \| Q \| m)$, independent of which signer public key is supplied to **VERIFY**, remains vulnerable even though it resists the direct PSI attack of Section 4. Given the challenge $\sigma^* = (C^*, h^*, \bar{z}^*)$ and the two signer secret vectors, an adversary with access to a verification oracle after the challenge can set $\delta_i = s_i^{(0)} - s_i^{(1)} \bmod N$ and $\bar{z}'_i = \bar{z}^*_i + \delta_i \bmod N$, and query verification of $(m^*, \sigma' = (C^*, h^*, \bar{z}'))$ under candidate S_1 . Because that k does not depend on the target public key, verification recovers the same masks regardless of the candidate queried and checks the curves $[v_i + z'_i]E_i^{(1)} = [v_i + z_i^* + s_i^{(0)}]E_0$, which coincide with the genuine hash-input curves exactly when the challenge came from S_0 and generically differ when it came from S_1 (with success probability $(1 - N^{-\eta})(1 - 2^{-\lambda-1}) + N^{-\eta}/2$). Since $\sigma' \neq \sigma^*$ whenever the signer vectors differ, this query survives a ban on the exact challenge pair. Binding k to pk_S , as adopted in Section 5, closes this gap: a query under S_1 then derives an independently random $k' \neq k^*$, so $[v_i + z'_i]E_i^{(1)}$ is uncorrelated with the genuine curves regardless of which candidate actually signed, and Theorem 4 holds without restricting the timing or target of verification queries.

Theorem 5 (Strong Unforgeability). *Assume that the Gap Parallelization Problem is hard and that $\mathcal{H}$, $\mathcal{H}_K$, and $\mathcal{H}_M$ are independent random oracles. Then the modified construction is strongly unforgeable under chosen-message attacks, up to negligible random-oracle guessing and collision probabilities.*

Proof. Suppose that a PPT adversary $\mathcal{A}$ produces a valid fresh signature with non-negligible probability. We construct an algorithm $\mathcal{B}$ that solves an externally supplied Gap Parallelization instance. Let the challenge given to $\mathcal{B}$ be

$$E^* = [s^*]E_0, \quad \hat{E}^* = [v^*]E_0.$$

The goal of $\mathcal{B}$ is to compute

$$K^* = [s^* + v^*]E_0.$$

The reduction chooses one coordinate, say j , and sets $E_j = E^*$ and $\hat{E}_j = \hat{E}^*$. Therefore, it knows neither s_j nor v_j . All remaining signer and verifier key components are generated normally by $\mathcal{B}$. Importantly, $\mathcal{B}$ also generates the masking key u itself. It can therefore compute $[u]C$ and remove the response masks for every signature submitted by $\mathcal{A}$. To understand the oracle simulation, recall that every valid unmasked CSI-SDVS transcript satisfies

$$Y_i = [z_i]K_i, \quad K_i = [s_i + v_i]E_0.$$

For the challenge coordinate, this becomes $Y_j = [z_j]K^*$. When answering a signing or simulation query, $\mathcal{B}$ chooses the response values z_i uniformly. For all non-challenge coordinates, it can compute

$Y_i = [z_i]K_i$ directly. It cannot initially compute Y_j because it does not know K^* . Instead, it chooses the signature hash h uniformly and records the corresponding hidden random-oracle point. This simulation is indistinguishable from a real signature until the adversary queries $\mathcal{H}$ at the corresponding hidden point, because the output of a random oracle at an unqueried point is itself uniformly random. Suppose $\mathcal{A}$ later makes a hash query containing a candidate curve Y_j . Since z_j is known to $\mathcal{B}$, it computes

$$K' = [-z_j]Y_j.$$

If the query is the genuine hidden point, then

$$K' = [-z_j][z_j]K^* = K^*.$$

The reduction can recognize this case using $\mathcal{D}_{\text{Par}}(E^*, \hat{E}^*, K')$. Once the test succeeds, $\mathcal{B}$ has obtained the solution K^* to the external parallelization instance. This also allows it to program the random oracle consistently with the previously returned signature. The same idea allows the verification oracle to be simulated. Because $\mathcal{B}$ knows the masking secret u , it first recovers $\mathbf{z}$ from any submitted signature. If the signature is one previously returned by the signing or simulation oracle, it is accepted. Otherwise, $\mathcal{B}$ examines the relevant $\mathcal{H}$ queries. A valid query must contain curves satisfying $Y_i = [z_i]K_i$. For the challenge coordinate, the value $[-z_j]Y_j$ must therefore equal K^* . If such a query exists, the reduction obtains the required parallelization value. If no corresponding random-oracle query exists, the adversary can make the verification oracle accept only by correctly guessing a fresh λ -bit hash value, which occurs with negligible probability.

Eventually, suppose $\mathcal{A}$ outputs a fresh valid forgery $\sigma^* = (C^*, h^*, \bar{\mathbf{z}}^*)$ on m^* . Since $\mathcal{B}$ knows u , it computes $Q^* = [u]C^*$, derives the masks, and recovers $\mathbf{z}^*$. For the forgery to verify,

$$h^* = \mathcal{H}(Y_1^* \parallel \cdots \parallel Y_\eta^* \parallel m^* \parallel C^*),$$

where $Y_j^* = [z_j^*]K^*$. Except with probability $2^{-\lambda}$, the adversary must previously have queried $\mathcal{H}$ at this exact input. From the challenge coordinate of that query, $\mathcal{B}$ computes

$$K^* = [-z_j^*]Y_j^*.$$

Thus, a successful fresh forgery yields the solution to the externally supplied parallelization instance. The only remaining cases are direct random-oracle guessing, accidental hash collisions, or collisions among the polynomially many simulated transcripts. All occur with negligible probability. Hence, a non-negligible SUF-CMA advantage would contradict the assumed hardness of Gap Parallelization. $\square$

7 Efficiency Analysis

We now analyze the storage, communication, and computational costs of the modified construction and compare them with the original CSI-SDVS scheme. We keep the number of parallel instances η explicit. Following CSI-SDVS, let the security parameter be λ , let an element of $\mathbb{Z}_N$ require $\ell_N = \lceil \log_2 N \rceil$ bits, and let a curve be represented by a Montgomery coefficient in $\mathbb{F}_p$, requiring $\ell_p = \lceil \log_2 p \rceil$ bits [6]. The original efficiency analysis writes $\log_2 N = 2\lambda$, which is only an approximation and not an identity, since the concrete CSIDH-512 class group order has $\log_2 N \approx 257.137$.

Table 2: Key and signature sizes of CSI-SDVS and the modified construction.

Component	CSI-SDVS	Modified scheme
Signer secret key	$\eta\ell_N$	$\eta\ell_N$
Signer public key	$\eta\ell_p$	$\eta\ell_p$
Verifier secret key	$\eta\ell_N$	$(\eta + 1)\ell_N$
Verifier public key	$\eta\ell_p$	$(\eta + 1)\ell_p$
Signature	$\lambda + \eta\ell_N$	$\lambda + \eta\ell_N + \ell_p$

Signer keys. The signer key generation algorithm is unchanged. The secret key contains η elements $s_i \in \mathbb{Z}_N$, while the public key contains η curves $E_i = [s_i]E_0$. Therefore,

$$|\text{sk}_S| = \eta\ell_N, \quad |\text{pk}_S| = \eta\ell_p$$

bits. Since $\ell_N \approx 2\lambda$ in the CSI-SDVS parameterization, the signer secret key has size approximately $2\eta\lambda$ bits. Thus, our modification introduces no additional storage at the signer.

Verifier keys. In CSI-SDVS, the designated verifier stores η secret values $v_i \in \mathbb{Z}_N$ and publishes the corresponding η curves $\hat{E}_i = [v_i]E_0$ [6]. Hence, the original verifier key sizes are $\eta\ell_N$ and $\eta\ell_p$ bits for the secret and public keys, respectively. Our construction introduces one additional masking secret $u \in \mathbb{Z}_N$ and the corresponding public curve $U = [u]E_0$. Therefore,

$$|\text{sk}_V| = (\eta + 1)\ell_N, \quad |\text{pk}_V| = (\eta + 1)\ell_p.$$

Thus, compared with CSI-SDVS, our verifier secret key increases by exactly one element of $\mathbb{Z}_N$, while the verifier public key increases by exactly one curve.

Signature size. An original CSI-SDVS signature has the form $\sigma = (h, z_1, \dots, z_\eta)$. It consists of one λ -bit hash value and η elements of $\mathbb{Z}_N$. Its size is therefore

$$|\sigma_{\text{CSI}}| = \lambda + \eta\ell_N.$$

Using $\ell_N \approx 2\lambda$, this becomes

$$|\sigma_{\text{CSI}}| \approx (2\eta + 1)\lambda$$

bits, which agrees with the efficiency analysis of CSI-SDVS [6].

Our modified signature has the form $\sigma = (C, h, \bar{z}_1, \dots, \bar{z}_\eta)$. The masked responses $\bar{z}_i$ are still elements of $\mathbb{Z}_N$ and therefore have exactly the same size as the original responses z_i . The only additional transmitted object is the ephemeral curve $C = [e]E_0$. Consequently,

$$|\sigma_{\text{mod}}| = \ell_p + \lambda + \eta\ell_N.$$

Equivalently,

$$|\sigma_{\text{mod}}| = |\sigma_{\text{CSI}}| + \ell_p.$$

Thus, the cost of obtaining signer privacy is exactly one additional curve per signature. Importantly, the communication overhead is independent of η . Table 2 summarizes the storage costs. For the parameter convention used in CSI-SDVS, $\ell_N \approx 2\lambda$. Hence, the same expressions can be written as

$$|\text{sk}_S| = 2\eta\lambda, \quad |\text{sk}_V| = 2(\eta + 1)\lambda,$$

while the signature sizes are $(2\eta + 1)\lambda$ bits for CSI-SDVS and $(2\eta + 1)\lambda + \ell_p$ bits for our construction.

Table 3: Approximate sizes for $\lambda = 128$, $\eta = 1$, $\ell_N = 258$, and $\ell_p = 512$.

Component	CSI-SDVS	Modified scheme
Signer secret key	33 B	33 B
Signer public key	64 B	64 B
Verifier secret key	33 B	66 B
Verifier public key	64 B	128 B
Signature	49 B	113 B

Table 4: Number of class-group action evaluations.

Operation	CSI-SDVS	Modified scheme
Signer key generation	η	η
Verifier key generation	η	$\eta + 1$
Signing	η	$\eta + 2$
Verification	η	$\eta + 1$
Simulation	η	$\eta + 2$

Concrete sizes. CSI-SDVS adopts the CSIDH-512 setting and considers $\lambda = 128$ [6]. In this setting, $\ell_N = \lceil \log_2 N \rceil$ is 258 bits, because $\log_2 N \approx 257.137$, and a curve represented by its Montgomery coefficient requires approximately 512 bits, or 64 bytes. CSIDH-512 public curves are conventionally represented using 64 bytes [4]. For $\eta = 1$, which the current CSI-SDVS parameter discussion considers sufficient, the approximate sizes are shown in Table 3. The original CSI-SDVS signature contains a 16-byte hash and one 33-byte response when $\eta = 1$, giving 49 bytes. Our scheme adds one 64-byte curve C , resulting in a signature of approximately 113 bytes. Although this more than doubles the signature size for $\eta = 1$, the absolute size remains small. More importantly, the additional cost is a fixed 64 bytes and does not grow with the number of parallel instances. For general η , the approximate signature sizes under the same parameter set are

$$|\sigma_{\text{CSI}}| = 16 + 33\eta \text{ bytes}, \quad |\sigma_{\text{mod}}| = 80 + 33\eta \text{ bytes}.$$

For example, if $\eta = 16$, the corresponding sizes are approximately 544 bytes and 608 bytes, respectively. In this case, the relative communication overhead of the additional curve is much smaller.

Computational cost. Let GA denote one evaluation of the class-group action. In the original CSI-SDVS signing algorithm, the signer computes $Y_i = [b_i]\hat{E}_i$ for each i , requiring η group-action evaluations. Verification similarly requires η evaluations to reconstruct the curves Y'_i [6]. Our signing algorithm performs the same η evaluations and additionally computes $C = [e]E_0$ and $Q = [e]U$. Hence, signing requires $\eta + 2$ group-action evaluations. Verification computes $Q = [u]C$ together with the original η verification actions, requiring $\eta + 1$ group-action evaluations. Simulation requires $\eta + 2$ actions: one to compute C , one to derive the ephemeral shared curve, and η to compute the simulated curves $Y_i = [r_i]E_i$. Verifier key generation also requires one additional group action to generate $U = [u]E_0$. The computational comparison is summarized in Table 4.

The modified construction also requires additional lightweight hash evaluations. In addition to the final signature hash $\mathcal{H}$, signing and verification evaluate $\mathcal{H}_K$ once and derive η response masks using $\mathcal{H}_M$. Thus, the modified scheme requires approximately $\eta + 2$ hash evaluations instead of the single main hash evaluation in CSI-SDVS. These operations are inexpensive compared with class-group-action evaluation and can be efficiently implemented using domain-separated calls to

a suitable extendable-output hash function. Binding pk_S into k (Section 6) enlarges the input to the single $\mathcal{H}_K$ evaluation in signing and verification by up to $\eta\ell_p$ bits, but adds no further hash or group-action evaluation, so Table 4 is unaffected.

8 Conclusion

In this paper, we revisited the security of CSI-SDVS, a recent isogeny-based strong designated-verifier signature scheme based on commutative class-group actions. Our analysis focused on the three main properties claimed by the construction: privacy of the signer’s identity, non-transferability, and strong unforgeability. We first showed that the PSI property of CSI-SDVS does not hold under the security experiment stated for the scheme. Motivated by the signer-privacy failure, we proposed a modified CSI-SDVS construction. We proved correctness and non-transferability of the modified scheme and analyzed PSI and SUF-CMA security under an explicit Gap Parallelization assumption in the random-oracle model. In contrast to the proof analyzed for CSI-SDVS, the unforgeability reduction in our construction embeds an externally supplied parallelization instance and makes the required computational assumption explicit. The modification introduces a limited efficiency overhead. The signer key sizes remain unchanged, while the designated verifier stores one additional secret group element and one additional public curve. Each signature contains one additional ephemeral curve, and the masking procedure requires only a small number of extra class-group actions and hash evaluations. Thus, the proposed construction retains the compact structure of CSI-SDVS while addressing the identified signer-privacy weakness.

Acknowledgments

Vikas Srivastava acknowledges the support received from the ANRF-PMECRG project with Ref. ANRF/ECRG/2025/002808/PMS and NIT Warangal Research Seed Grant.

References

- [1] Markus Jakobsson, Kazue Sako, and Russell Impagliazzo. Designated verifier proofs and their applications. In *Advances in Cryptology—EUROCRYPT ’96*, volume 1070 of *Lecture Notes in Computer Science*, pages 143–154. Springer, 1996. https://doi.org/10.1007/3-540-68339-9_13.
- [2] Fabien Laguillaumie and Damien Vergnaud. Designated verifier signatures: Anonymity and efficient construction from any bilinear map. In *Security in Communication Networks (SCN 2004)*, volume 3352 of *Lecture Notes in Computer Science*, pages 105–119. Springer, 2004. https://doi.org/10.1007/978-3-540-30598-9_8.
- [3] Peter W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *Proceedings of the 35th Annual Symposium on Foundations of Computer Science*, pages 124–134. IEEE, 1994. <https://doi.org/10.1109/SFCS.1994.365700>.
- [4] Wouter Castryck, Tanja Lange, Chloe Martindale, Lorenz Panny, and Joost Renes. CSIDH: An efficient post-quantum commutative group action. In *Advances in Cryptology—ASIACRYPT 2018*, volume 11274 of *Lecture Notes in Computer Science*, pages 395–427. Springer, 2018. https://doi.org/10.1007/978-3-030-03332-3_15.

- [5] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. CSI-FiSh: Efficient isogeny based signatures through class group computations. In *Advances in Cryptology—ASIACRYPT 2019*, volume 11921 of *Lecture Notes in Computer Science*, pages 227–247. Springer, 2019. https://doi.org/10.1007/978-3-030-34578-5_9.
- [6] Farzin Renan. A compact post-quantum strong designated verifier signature scheme from isogenies. arXiv:2507.14893v3; IACR Cryptology ePrint Archive, Report 2025/1335, 2025. <https://arxiv.org/abs/2507.14893>; <https://eprint.iacr.org/2025/1335>.
- [7] Jean-Marc Couveignes. Hard homogeneous spaces. IACR Cryptology ePrint Archive, Report 2006/291, 2006. <https://eprint.iacr.org/2006/291>.