Relaxed Vector Commitment for Shorter Signatures

Seongkwang Kim[1], Byeonghak Lee[1], and Mincheol Son[2]

[1] Samsung SDS, Seoul, Korea
sk39.kim@samsung.com,byghak.lee@samsung.com
[2] KAIST, Daejeon, Korea
encrypted.def@kaist.ac.kr

Abstract. The MPC-in-the-Head (MPCitH) paradigm has recently gained traction as a foundation for post-quantum signature schemes, offering robust security without the need for trapdoors. Despite its strong security profile, MPCitH-based schemes suffer from high computational overhead and large signature sizes, limiting their practical application.
This work addresses these inefficiencies by enhancing vector commitments within MPCitH-based schemes. We introduce the concept of *vector semi-commitment*, which relaxes traditional vector commitment requirements without compromising security, thus reducing signature size while maintaining performance. We instantiate vector semi-commitment schemes in both the random oracle model and the ideal cipher model, leveraging recent optimizations such as the Half-tree technique. Additionally, we propose a key injection technique that further minimizes signature size by embedding the secret key into the Half-GGM tree.
We apply these improvements to the BN++ signature scheme and prove it fully secure in the ideal cipher model. Implementing these improvements in the AIMer v2.0 signature scheme, we achieve up to 18% shorter signatures and up to 112% faster signing and verification speeds, setting new benchmarks for MPCitH-based schemes.

Keywords: MPC-in-the-Head, vector commitment, GGM tree, zero-knowledge proof, digital signature, commitment scheme

1 Introduction

Recently, the MPC-in-the-Head (MPCitH) paradigm [19] has emerged as a promising approach for designing post-quantum signature schemes. This paradigm leverages the concept of multi-party computation (MPC) to perform computations within a single entity's "head" and has been applied to zero-knowledge proofs and signature schemes. MPCitH-based signature schemes enable a signer to generate a signature without relying on a trapdoor, making their security depend solely on the one-way function used in key generation. This advantage allows primitives without trapdoors, such as block ciphers [28,11,23], unstructured multivariate quadratic (MQ) problem [13], or unstructured syndrome decoding problem [14] to base the security of signature schemes. This makes them more reliable compared to schemes whose security is based on artificially constructed hardness assumptions with potential gaps in the security reduction.

Despite their promising security feature, MPCitH-based signature schemes have been hindered by relatively high computational overhead and large signature sizes, making them less efficient compared to their lattice-based counterparts. The inherent complexity of simulating multi-party computations results in quadratic time and signature size with respect to the security parameter, which can be detrimental to practical adoption. In response, many studies have focused on optimizing the efficiency of MPC-in-the-Head-based signature schemes through protocol optimization [22,6,27,1] and improved cryptographic primitives [11,23].

One notable line of work for improving the efficiency of MPCitH-based signature schemes is to optimize the GGM (Goldreich-Goldwasser-Micali) tree or vector commitment in the context of VOLE-in-the-Head (VOLEitH) [4] which includes the GGM tree and subsequent commitments afterward, a component used to generate shares of the virtual parties. The GGM tree enables the secure distribution of shares among the virtual parties. However, traditional GGM tree construction can be computationally expensive, contributing significantly to the overall inefficiency of MPCitH-based schemes. To address this, researchers developed more efficient GGM tree constructions, such as double-length PRG instantiated by fixed-key block cipher [9,8] and the application of Half-tree technique [10,9].

1.1 Our Contribution

In this work, we focus on improving vector commitments used in MPCitH-based signature schemes. Our primary enhancement is to relax the vector commitment requirements. A vector commitment scheme, a crucial component of our MPCitH-based signature scheme, must satisfy two key properties: hiding and (extractable) binding. The hiding property ensures that the commitment conceals the committed values, safeguarding the data's secrecy. The binding property ensures that once a commitment is made, it is computationally infeasible to alter the committed values without detection. Notably, a violation of the binding property does not directly result in a signature forgery.

We introduce a relaxed version of vector commitment, called *vector semi-commitment*, which ensures the hardness of finding a preimage and finding "many" collisions of a commitment. Under a plausible assumption, we prove that replacing a vector commitment with a vector semi-commitment does not compromise security. Performance-related parameters, such as the number of repetitions, remain unchanged, thereby reducing the signature size while maintaining performance.

We then instantiate vector semi-commitment schemes in both the random oracle model and the ideal cipher model. For the latter, we fully instantiate all the random primitives using ideal cipher calls and incorporate recent optimizations of the GGM tree construction [9,8]. By slightly modifying the Davies-Mayer construction provided in [8], our vector semi-commitment scheme enjoys faster speed of both the fixed-key block cipher and the Half-tree technique.

Furthermore, we introduce a key injection technique that can be applied to the Half-tree technique. By injecting the secret key of the signature scheme as the root of the Half-GGM tree, the sum of the leaf nodes (which are seeds) always equals the secret key. While MPCitH-based signatures usually include the correction of the secret key shares, the key injection method eliminates this requirement, leading to a reduction in signature size.

Finally, we apply all the improvements to BN++ [21], an MPCitH-based signature scheme. We prove its full security[3] in the ideal cipher model. We also implement our improvements in AIMer v2.0 [24], achieving up to 18% shorter signatures and up to 112% faster signing and verification speeds compared to AIMer v2.0. Compared to other MPCitH-based signature schemes such as SDitH [26] and FAEST [3], it also offers the fastest performance and the shortest signature size. Detailed performance figures are summarized in Table 2.

1.2 Related Work

In MPCitH-based signature schemes, a prover emulates an MPC protocol among N parties "in her head" and then opens the views of $(N-1)$ parties except one. The verifier accepts if all the views are consistent with an honest execution of the MPC protocol. Katz et al. employed the GGM tree to reduce the number of opened random seeds from $N-1$ to $\log N$, subsequently applying it to the MPCitH-based signature scheme Picnic [22]. Since then, the GGM tree has been used as a core technique to reduce the signature size in the MPCitH-based signature schemes [11,23,14,26,7].

There have been efforts to improve GGM trees. Guo et al. proposed a correlated GGM tree [17] in the context of correlated oblivious transfer and distributed point function. In a correlated GGM tree, the sum of all nodes at the same level is fixed. This reduces the number of random permutation calls by half. The correlated GGM tree was recently applied to a VOLE-in-the-Head [10]. Using the correlated GGM tree, the number of random permutation calls for generating seeds is halved. Concurrently, Bui and Cong proposed applying the correlated GGM tree to MPC-in-the-Head and VOLE-in-the-Head [9]. They also replaced the random oracle calls used for commitment with random permutation calls, which can be implemented using efficient primitives such as fixed-key AES. However, the provable security of their proposal did not exceed the birthday bound. Independent with the Half-tree technique, Bui et al. proposed a fast salted GGM tree which is fully secure in the ideal cipher model [8]. Using this salted tree, tree evaluation is as fast as the best unsalted version while preventing multi-target attacks.

[3] This means the scheme is secure against $O(2^\lambda)$ queries to the signing oracle or ideal primitives.

2 Preliminaries

2.1 Notations

For two vectors or strings a and b, their concatenation is denoted by $a\|b$. For integers a and b, we denote the bitwise XOR of a and b by $a \oplus b$. Bitwise right shift by i of a is denoted by $a \gg i$.

We denote $[n] = \{1, \cdots, n\}$. Unless stated otherwise, all logarithms are to the base 2. For an integer $a \in \{0, 1, \ldots, 255\}$, $\langle a \rangle_B$ is the canonical binary representation of a, which is an 8-bit string. For a positive integer n and $k < n$, the falling factorial is denoted by $(n)_k = n \cdot (n-1) \cdot \cdots \cdot (n-k+1)$.

For a set S, we write $a \leftarrow_\$ S$ to denote that a is chosen uniformly at random from S. For a probability distribution $\mathcal{D}$, $a \leftarrow_\$ \mathcal{D}$ denotes that a is sampled according to the distribution $\mathcal{D}$. We denote the binomial distribution with n trials and probability p by $B(n, p)$.

Throughout this paper, the security parameter is denoted by λ. In the multiparty computation setting, $x^{(i)}$ denotes the i-th party's additive share of x, implying that $\sum_i x^{(i)} = x$.

For $x \in \{0,1\}^\lambda$, $2 \cdot x$ denotes multiplying x by X over $\mathbb{F}_{2^\lambda}$ for some irreducible polynomial $f(X)$. Similarly, $3 \cdot x$ denotes multiplying x by $X + 1$.

2.2 Chernoff Bound

Chernoff bound is a well-known upper bound on the tail of a random variable. As our proof relies on this upper bound, we briefly introduce the Chernoff bound in its multiplicative form.

Let X be a random variable following the binomial distribution $B(Q, p)$. Then, the probability of the tail of X is upper bounded by

$$\Pr[X > (1+\delta)\mu] < \left(\frac{e^\delta}{(1+\delta)^{1+\delta}} \right)^\mu$$

for any $\delta > 0$, where e is Euler's number and μ is the expectation of X.

In our security proof, we use $p = 1/2^\lambda$. Then, the multiplicative Chernoff bound is of the form

$$\Pr[X > c] < \left(\frac{eQ}{c2^\lambda} \right)^c$$

where $c > 1$. Given a multiset $S = \{x_1, x_2, \ldots, x_Q\}$ with $x_i \leftarrow_\$ \{0,1\}^\lambda$, we can bound the probability that the maximal multi-collision mcoll in S is greater than $2\lambda/\log\lambda$ as follows. With $\lambda \geq 16$ and $Q \leq 2^{\lambda-1}$, the probability is bounded by

$$\begin{aligned}
\Pr[\mathsf{mcoll} > 2\lambda/\log\lambda] &< \left(\frac{eQ \cdot \log\lambda}{2\lambda \cdot 2^\lambda} \right)^{2\lambda/\log\lambda} \cdot 2^\lambda = \left(\frac{eQ_c \cdot \log\lambda}{2\lambda^{1/2} \cdot 2^\lambda} \right)^{2\lambda/\log\lambda} \\
&\leq \left(\frac{eQ}{2 \cdot 2^\lambda} \right)^{2\lambda/\log\lambda} \qquad (\because \log\lambda \leq \lambda^{1/2}) \\
&\leq \frac{eQ}{2^{\lambda+1}} \quad (\because eQ \leq 2^{\lambda+1} \leq \frac{2Q}{2^\lambda}.
\end{aligned} \tag{1}$$

2.3 GGM Tree

GGM tree is a binary tree proposed by Goldreich, Goldwasser, and Micali [16]. For a power-of-two integer N, one can send $N-1$ out of N random strings with $\log N$ communication by using a GGM tree. For a pseudorandom generator $G : \{0,1\}^n \to \{0,1\}^{2n}$ and a root node_0, the nodes in a GGM tree are defined recursively as follows.

$$\mathsf{node}_{1,1} \| \mathsf{node}_{1,2} = G(\mathsf{node}_0)$$
$$\mathsf{node}_{i,2j-1} \| \mathsf{node}_{i,2j} = G(\mathsf{node}_{i-1,j}) \quad \text{for } i \geq 2 \text{ and } 0 < j \leq 2^{i-1}$$

Let a GGM tree $\mathcal{T}$ has 2^d leaf nodes. If one wants to send all leaf nodes except k-th leaf node (i.e., $\mathsf{node}_{d,k}$), she can send a Merkle path

$$\left(\mathsf{node}_{1,(((k-1)\gg(d-1))\oplus 1)+1}, \mathsf{node}_{2,(((k-1)\gg(d-2))\oplus 1)+1}, \ldots, \mathsf{node}_{d,((k-1)\oplus 1)+1}\right).$$

We will call this Merkle path an *associate path* of the unopened node $\mathsf{node}_{d,k}$. Conversely, we will call the unopened node $\mathsf{node}_{d,k}$ an *associate node* of the Merkle path described above. The depth of a node is defined by the length of the shortest upward path to the root (e.g., the depth of $\mathsf{node}_{1,1}$ is 1), and the height of a node is defined by the length of the longest downward path to a leaf node.

2.4 BN++ Zero-knowledge Protocol

In this section, we briefly review the BN++ proof system [21], one of the state-of-the-art MPCitH-based zero-knowledge protocols. At a high level, BN++ is a variant of the BN protocol [5] with several optimization techniques applied to reduce the signature size.

Protocol Overview. BN++ essentially simulates multiparty computation of *triple checking protocol*, which verifies that all the multiplication triples are honestly generated. To check C multiplication triples $(x_j, y_j, z_j = x_j \cdot y_j)_{j=1}^C$ over a finite field $\mathbb{F}$ in the multiparty computation setting with N parties, *helping values* $((a_j, b_j)_{j=1}^C, c)$ are required, where $a_j \leftarrow \mathbb{F}, b_j = y_j$, and $c = \sum_{j=1}^C a_j \cdot b_j$. Each party holds secret shares of the multiplication triples $(x_j, y_j, z_j)_{j=1}^C$ and helping values $((a_j, b_j)_{j=1}^C, c)$. Then the protocol proceeds as follows.

- A prover is given random challenges $\epsilon_1, \cdots, \epsilon_C \in \mathbb{F}$.
- For $i \in [N]$, the i-th party locally sets $\alpha_1^{(i)}, \cdots, \alpha_C^{(i)}$ where $\alpha_j^{(i)} = \epsilon_j \cdot x_j^{(i)} + a_j^{(i)}$.
- The parties open $\alpha_1, \cdots, \alpha_C$ by broadcasting their shares.
- For $i \in [N]$, the i-th party locally sets

$$v^{(i)} = \sum_{j=1}^{C} \epsilon_j \cdot z_j^{(i)} - \sum_{j=1}^{C} \alpha_j \cdot b_j^{(i)} + c^{(i)}.$$

- The parties open v by broadcasting their shares and output `Accept` if $v = 0$.

By Lemma 1, the probability that there exist incorrect triples and the parties output `Accept` in a single run of the above steps is upper bounded by $1/|\mathbb{F}|$.

Lemma 1 ([21]). *If the secret-shared input $(x_j, y_j, z_j)_{j\in[C]}$ contains an incorrect multiplication triple, or if the shares of $((a_j, y_j)_{j\in[C]}, c)$ form an incorrect dot product, then the parties output `Accept` in the sub-protocol with probability at most $1/|\mathbb{F}|$.*

Signature Size. By applying the Fiat-Shamir transform [12], one can obtain a signature scheme from the BN++ proof system. In this signature scheme, the signature size is given as

$$6\lambda + \tau \cdot (3\lambda + \lambda \cdot \lceil \log_2(N) \rceil + \mathcal{M}(C)),$$

where λ is the security parameter, C is the number of multiplication gates in the underlying symmetric primitive, and $\mathcal{M}(C) = (2C+1)\cdot \log_2(|\mathbb{F}|)$. In particular, $\mathcal{M}(C)$ is defined from the observation that sharing the secret share offsets for $(z_j)_{j=1}^{C}$ and c, and opening shares for $(\alpha_j)_{j=1}^{C}$ occurs for each repetition, using C, 1, and C elements of $\mathbb{F}$, respectively.

2.5 H-coefficient Technique

The H-coefficient technique is a powerful method used in the analysis of cryptographic algorithms, particularly in the context of provable security. Introduced by Patarin, this technique provides a systematic way to bound the distinguishing advantage of an adversary interacting with an idealized cryptographic system and a real implementation. The core idea of the H-coefficient technique is to partition the set of possible transcripts (i.e., sequences of queries and responses) into two subsets: "good" and "bad" transcripts. The probability of bad transcripts can be shown to be negligible, while the probability of distinguishing between the distributions of good transcripts can be tightly bounded.

The technique is especially useful in scenarios involving pseudorandom functions (PRFs), encryption schemes, and protocols where interactions can be modeled as sequences of random variables. By leveraging the H-coefficient technique, one can obtain strong security guarantees with clear, quantifiable bounds.

To illustrate the H-coefficient technique, we present the following lemma, which is a fundamental component of the technique:

Lemma 2 (H-Coefficient Lemma). *Let $\mathcal{A}$ be an algorithm that interacts with either an ideal world $\mathcal{I}$ or a real world $\mathcal{R}$ and tries to distinguish two worlds. Let T_{id} and T_{re} be the distribution of transcript in the ideal world and the real world, respectively, and $\mathcal{T}$ denote the set of all attainable transcripts in the ideal world. Suppose there exist partition $\mathcal{T}_{\mathsf{Good}}$ (good transcripts) and $\mathcal{T}_{\mathsf{Bad}}$ (bad transcripts) of $\mathcal{T}$, and constants ϵ and δ such that for any $\gamma \in \mathcal{T}_{\mathsf{Good}}$,*

$$\Pr[T_{\mathsf{id}} \in \mathcal{T}_{\mathsf{Bad}}] \leq \epsilon, \qquad \frac{\Pr[\gamma = T_{\mathsf{re}}]}{\Pr[\gamma = T_{\mathsf{id}}]} \geq 1 - \delta.$$

Then, the distinguishing advantage of $\mathcal{A}$ in distinguishing $\mathcal{I}$ from $\mathcal{R}$ is bounded by:

$$\mathbf{Adv}^{\mathsf{dist}}_{\mathcal{I},\mathcal{R}}(\mathcal{A}) \leq \epsilon + \delta.$$

This lemma provides a clear framework for analyzing the security of cryptographic protocols. By carefully defining the sets of good and bad transcripts and bounding their probabilities, one can apply the H-coefficient technique to obtain rigorous security proofs.

3 Vector Semi-commitment

In the context of MPCitH-based signature schemes, vector commitment (VC) abstracts the process of generating views and corresponding commitments. A vector commitment scheme typically consists of four sub-algorithms: Commit, Open, Recon, and Verify. The Commit algorithm generates the views of virtual parties and commits to these views. When a verifier challenges which views to reveal, the prover uses Open to disclose a subset of the views. Open produces partial decommitment information. Using this partial decommitment information, the verifier runs Recon to reconstruct the revealed messages and the corresponding commitments. Finally, the verifier uses Verify to check the validity of the partial decommitment information.

A vector semi-commitment scheme shares the same interface as vector commitments, with one of its properties relaxed compared to traditional vector commitments. We introduce the interface of vector semi-commitment in the random oracle model. Although the following is described in the context of the random oracle model, it can be easily adapted to the ideal cipher model.

Definition 1 (Vector Semi-commitment). *Let H be a random oracle. An IV-based vector commitment scheme VSC with message space $\mathcal{M}$ in a random oracle model is defined by the following PPT algorithms.*

- $\mathsf{Commit}^H(\mathsf{salt}, \mathsf{root}) \to (\mathsf{com}, \mathsf{decom}, (m_1, \ldots, m_N))$*: given an IV* $(= \mathsf{salt})$ *and* root*, output a commitment* com *with opening information* decom *for messages* $m = (m_1, \ldots, m_N) \in \mathcal{M}^N$*.*
- $\mathsf{Open}^H(\mathsf{salt}, \mathsf{decom}, I) \to \mathsf{pdecom}$*: given an IV* $(= \mathsf{salt})$*, opening information* decom *and a subset* $I \subseteq [N]$ *of indices, output a partial opening information* pdecom *for* I*.*
- $\mathsf{Recon}^H(\mathsf{salt}, \mathsf{pdecom}, I) \to ((m_i)_{i \in I}, \mathsf{com})$*: given an IV* $(= \mathsf{salt})$*, a partial opening information* pdecom *for a subset* I*, output partially reconstructed messages and the full commitment* com*.*
- $\mathsf{Verify}^H(\mathsf{salt}, \mathsf{com}, \mathsf{pdecom}, I) \to \{(m_i)_{i \in I}\} \cup \{\perp\}$*: given an IV* $(= \mathsf{salt})$*, commitment* com*, a partial opening information* pdecom*, and a subset* I*, either output the messages* $(m_i)_{i \in I}$ *(accept) or* $\perp$ *(reject).*

Although the interface is written for a general subset I, we will primarily consider I as a subset missing a single element unless otherwise specified. We will refer to this opening as *all-but-one*, and I as an all-but-one subset. If the context is clear, we will omit the random oracle indication $(^H)$.

The messages of vector semi-commitments are seeds in the signature scheme. If a prover can fix the sum of these seeds, it can be used to distribute shares without correction. Specifically, if a λ-bit substring of the secret key sk is injected as root, the correction of the secret key shares (usually denoted by $\Delta\mathsf{sk}$) is always zero. The Half-tree technique [17] enables this in vector semi-commitments, and the following property is inspired by this technique.

Definition 2 (Correlated Vector Semi-commitment). *A vector commitment scheme* $(\mathsf{Commit}^H, \mathsf{Open}^H, \mathsf{Verify}^H)$ *is called* correlated *vector commitment if*

$$\mathsf{Commit}^H(\mathsf{salt}, \mathsf{root}) = (\mathsf{com}, \mathsf{decom}, (m_1, \ldots, m_N))$$

implies $\mathsf{msb}_\lambda(m_1 \oplus \cdots \oplus m_N) = \mathsf{root}$

The major difference between vector semi-commitment and vector commitment is the binding property. Informally, the binding property of a commitment implies that it is hard to find multiple messages corresponding to the same commitment. A vector semi-commitment has a relaxed version of this property: the extractable semi-binding property. This property implies that while an adversary may find a small number of messages corresponding to the same commitment, it is hard to find a large number of such messages.

Definition 3 (Extractable Semi-binding). *Let* VSC *be an IV-based vector semi-commitment scheme in the random oracle model with random oracle H. Let* $\mathsf{Ext}(\mathcal{Q}, \mathsf{com}) \to (m_i)_{i\in[N]}$ *be a PPT algorithm that, given a set of query-response pairs of random oracle queries $\mathcal{Q}$ and a commitment* com*, outputs the committed messages $(m_i)_{i\in[N]}$. The u-extractable semi-binding game for* VSC*, denoted as* u-ESB*, with $N = poly(\lambda)$ and Q queries to the random oracle and stateful $\mathcal{A}$, is defined as follows.*

1. $(\mathsf{salt}, \mathsf{com}, \mathsf{pdecom}_I, (m_i)_{i\in I}, I) \leftarrow \mathcal{A}^H(1^\lambda, Q)$
2. $((m_i^{(j)})_{i\in[N]})_{j\in J} \leftarrow \mathsf{Ext}(\mathcal{Q}, \mathsf{com})$*, where $\mathcal{Q}$ is the set $\{(x_i, H(x_i))\}$ of query-response pairs of queries $\mathcal{A}$ made to H and $|J| \leq u$.*
3. *Output 1 if* $\mathsf{Verify}^H(\mathsf{salt}, \mathsf{com}, \mathsf{pdecom}_I, I) \to (m_i^*)_{i\in I}$ *but $m_i^* \neq m_i$ for any $i \in I$; otherwise, output 0.*

We define $\mathcal{A}$'s u-extractable semi-binding advantage by

$$\mathbf{Adv}^{u\text{-}\mathsf{ESB}}_{\mathsf{VSC}}(\mathcal{A}) = \Pr[\mathcal{A} \textit{ wins } u\text{-}\mathsf{ESB}]$$

Although the extractor Ext extracts the messages $(m_i^{(j)})_{i,j}$, the purpose of the extractable semi-binding game is to bound the number of valid pdecom for the same com. Therefore, the extractor should be programmed to extract messages only from valid pdecom's. We note that Ext may output $m_i = \perp$ if the committed value at index i is invalid.

The next property of vector semi-commitment is hiding, but the following definition is in the multi-instance form. The intuition behind the multi-instance hiding property is that given multiple puncturable PRF instances and their commitments, the punctured messages are indistinguishable from random.

Definition 4 (Multi-Instance Hiding). *Let* VSC *be an IV-based vector semi-commitment scheme in the random oracle model with random oracle* H*. The multi-instance hiding game for vector semi-commitments,* Q_I-MIH*, with* $N = poly(\lambda)$ *and* Q *queries to the random oracle and stateful* $\mathcal{A}$ *is defined as follows.*

1. $\mathsf{root} \leftarrow_\$ \{0,1\}^\lambda$, $b^* \leftarrow_\$ \{0,1\}$
2. *For* $j \in [Q_I]$*, do the following:*
   (a) $\mathsf{salt}_j \leftarrow_\$ \{0,1\}^{2\lambda}$
   (b) $(\mathsf{com}_j, \mathsf{decom}_j, (m^*_{j,1}, \ldots, m^*_{j,N})) \leftarrow \mathsf{Commit}^H(\mathsf{salt}_j, \mathsf{root})$
   (c) $\bar{i}_j \leftarrow_\$ [N]$, $I_j = [N] \setminus \{\bar{i}_j\}$
   (d) $\mathsf{pdecom}_{I_j} \leftarrow \mathsf{Open}^H(\mathsf{salt}_j, \mathsf{decom}, I_j)$
   (e) $m_{j,i} \leftarrow m^*_{j,i}$ *for* $i \in I_j$.
3. $(x_1, \ldots, x_q) \leftarrow \mathcal{A}^H((\mathsf{salt}_j, \mathsf{pdecom}_{I_j}, I_j)_{j\in[Q_I]})$
4. *For* $j \in [Q_I]$*, set* $m_{j,\bar{i}_j} \leftarrow \begin{cases} m^*_{j,\bar{i}_j} & \text{if } b^* = 0, \\ \leftarrow_\$ \mathcal{M} & \text{otherwise.} \end{cases}$
5. $b \leftarrow \mathcal{A}((m_{j,i})_{i\in[N],j\in[Q_I]}, (x_i, H(x_i))_{i\in[q]})$.
6. *Output 1 if* $b = b^*$*, else 0.*

We define $\mathcal{A}$*'s multi-instance hiding advantage by*

$$\mathbf{Adv}^{Q_I\text{-}\mathsf{MIH}}_{\mathsf{VSC}}(\mathcal{A}) = \left|\Pr[\mathcal{A} \text{ wins } Q_I\text{-}\mathsf{MIH}] - \frac{1}{2}\right|.$$

We call VSC *is multi-instance hiding if advantage of any* Q_I-MIH *adversary for polynomially bounded* Q_I *is negligible.*

3.1 Instantiation from Random Oracle

In this section, we instantiate VSC from a random oracle, dubbed $\mathsf{RO\text{-}VSC}$, and prove the extractable semi-binding property and multi-instance hiding property. Let $H_{\mathsf{com}} : \{0,1\}^* \to \{0,1\}^\lambda$, $H_{\mathsf{tree}} : \{0,1\}^* \to \{0,1\}^\lambda$, and $H_{\mathsf{exp}} : \{0,1\}^* \to \mathbb{F}^{2C+1}$ be random oracles. Then, $\mathsf{RO\text{-}VSC}$ is constructed as Figure 1.

The commitment process Commit^H involves an evaluation of the Half-GGM tree to generate seeds and tapes, culminating in the output of a commitment com and opening information decom. The opening algorithm Open^H extracts path information for a given index set, while the reconstruction algorithm Recon^H rebuilds messages and commitments from path data and verifies the integrity of the commitment. Finally, the verification algorithm Verify^H ensures that the

- **Parameters**: a triple of random oracles $H = (H_{\mathsf{com}}, H_{\mathsf{tree}}, H_{\mathsf{exp}})$, an integer C, a power-of-two integer $N = 2^d$.
- **Inputs**: $\mathsf{salt} \in \{0,1\}^{2\lambda}$, and $\mathsf{root} \in \{0,1\}^{\lambda}$.

- $\mathsf{Commit}^H(\mathsf{salt}, \mathsf{root})$:
  1. Set $\mathsf{node}_{1,1} \leftarrow \mathsf{root}$.
  2. For each level $e \in [d-1]$ and $i \in [2^e]$, set
  $$\begin{aligned} \mathsf{node}_{e+1,2i-1} &\leftarrow H_{\mathsf{tree}}(\mathsf{salt}, \mathsf{node}_{e,i}) \\ \mathsf{node}_{e+1,2i} &\leftarrow H_{\mathsf{tree}}(\mathsf{salt}, \mathsf{node}_{e,i}) \oplus \mathsf{node}_{e,i}. \end{aligned}$$
  3. For $i \in [N]$, set
  $$\begin{aligned} \mathsf{seed}_i &\leftarrow \mathsf{node}_{d,i} \\ \mathsf{com}_i &\leftarrow H_{\mathsf{com}}(\mathsf{salt}, i, \mathsf{seed}_i) \\ \mathsf{tape}_i &\leftarrow H_{\mathsf{exp}}(\mathsf{salt}, i, \mathsf{seed}_i) \\ m_i &= \mathsf{seed}_i \parallel \mathsf{tape}_i \end{aligned}$$
  4. Output a commitment $\mathsf{com} = (\mathsf{com}_1, \ldots, \mathsf{com}_N)$ with opening information $\mathsf{decom} = ((\mathsf{node}_{e,i})_{e \in [d-1], i \in [2^e]}, \mathsf{com})$, and messages $(m_1, \ldots, m_N)$.

- $\mathsf{Open}^H(\mathsf{salt}, \mathsf{decom}, I = [N] \setminus \{\bar{i}\})$:
  1. Set $\mathsf{path}_I \leftarrow (\mathsf{node}_{d-e+1, i_e})_{e \in [d-1]}$ where $i_e = (\lfloor (\bar{i} - 1)/2^{e-1} \rfloor \oplus 1) + 1$ for $e \in [d-1]$;
  2. Output $\mathsf{pdecom} = (\mathsf{path}_I, \mathsf{com}_{\bar{i}})$

- $\mathsf{Recon}^H(\mathsf{salt}, \mathsf{pdecom}, I = [N] \setminus \{\bar{i}\})$:
  1. Similarly as Step 2 in Commit, expand each node in path_I and get $(\mathsf{node}_{d,i})_{i \in I}$.
  2. For $i \in I$, do Step 3 in Commit.
  3. Output $((m_i)_{i \in I}, \mathsf{com})$.

- $\mathsf{Verify}^H(\mathsf{salt}, \mathsf{com} = (\mathsf{com}_i^*)_{i \in [N]}, \mathsf{pdecom}, I = [N] \setminus \{\bar{i}\})$:
  1. Similarly as Step 2 in Commit, expand each node in path_I and get $(\mathsf{node}_{d,i})_{i \in I}$.
  2. For $i \in I$, do Step 3 in Commit.
  3. Output $(m_i)_{i \in I}$ if $\mathsf{com} = \mathsf{com}_i^*$ for all $i \in I$, or output $\perp$ otherwise.

Fig. 1: RO-VSC

reconstructed messages match the original commitment. The integer C corresponds to the number of multiplication gates, and the integer N corresponds to the number of parties in the signature scheme.

Before proving the extractable semi-binding property, we bound the collision probability as follows.

Lemma 3. *Let $H_{\mathsf{com}} : \{0,1\}^* \to \{0,1\}^\lambda$, $H_{\mathsf{tree}} : \{0,1\}^* \to \{0,1\}^\lambda$ be random oracles. Let $\mathcal{A}$ be arbitrary adversary that makes Q_c queries to H_{com} and Q_t queries to H_{tree}. Then, the probability that $\mathcal{A}$ finds $\mathsf{salt} \in \{0,1\}^{2\lambda}$, $i \in \mathbb{N}$, and distinct $n, n' \in \{0,1\}^\lambda$ such that*

$$\begin{cases} H_{\mathsf{com}}(\mathsf{salt}, i, H_{\mathsf{tree}}(\mathsf{salt}, n)) = H_{\mathsf{com}}(\mathsf{salt}, i, H_{\mathsf{tree}}(\mathsf{salt}, n')), \\ H_{\mathsf{com}}(\mathsf{salt}, i, (H_{\mathsf{tree}}(\mathsf{salt}, n) \oplus n)) = H_{\mathsf{com}}(\mathsf{salt}, i, (H_{\mathsf{tree}}(\mathsf{salt}, n') \oplus n')) \end{cases} \tag{2}$$

is at most $9Q/2^\lambda$.

Proof. Without loss of generality, assume that $\mathcal{A}$ queries to the random oracles with fixed salt and i, and we omit the salt and i input for each random oracle query. At the end of the game, we define $H_{\mathsf{tree}}(x) = \bot$ (resp. $H_{\mathsf{com}}(x) = \bot$) for non-queried input x to H_{tree} (resp. H_{com}), and we consider that two $\bot$'s are not identical for simplicity. We define

$$\begin{aligned} \mathcal{L}_1 &= \left\{(n, l, l') \in \{0,1\}^{3\lambda} : H_{\mathsf{tree}}(n) = l, H_{\mathsf{com}}(l) = H_{\mathsf{com}}(l'), l \neq l'\right\}, \\ \mathcal{L}_2 &= \left\{(n, r, r') \in \{0,1\}^{3\lambda} : H_{\mathsf{tree}}(n) \oplus n = r, H_{\mathsf{com}}(r) = H_{\mathsf{com}}(r'), r \neq r'\right\}, \\ \mathcal{L}_3 &= \left\{(n, l, l', r, r') \in \{0,1\}^{5\lambda} : (n, l, l') \in \mathcal{L}_1, (n, r, r') \in \mathcal{L}_2\right\} \end{aligned}$$

and auxiliary events Aux_i for $i \in [3]$, where

$$\mathsf{Aux}_i \Leftrightarrow |\mathcal{L}_i| > Q_c$$

and let $\mathsf{Aux} = \mathsf{Aux}_1 \vee \mathsf{Aux}_2 \vee \mathsf{Aux}_3$. Then, by Markov's inequality, we have

$$\begin{aligned} \Pr[\mathsf{Aux}_1] &\leq \frac{\mathsf{Ex}[|\mathcal{L}_1|]}{Q_c} \leq \frac{1}{Q_c}\left(\frac{Q_t Q_c}{2^\lambda} + \frac{Q_t Q_c^2}{2^{2\lambda}}\right), \\ \Pr[\mathsf{Aux}_2] &\leq \frac{\mathsf{Ex}[|\mathcal{L}_2|]}{Q_c} \leq \frac{1}{Q_c}\left(\frac{Q_t Q_c}{2^\lambda} + \frac{Q_t Q_c^2}{2^{2\lambda}}\right), \\ \Pr[\mathsf{Aux}_3] &\leq \frac{\mathsf{Ex}[|\mathcal{L}_3|]}{Q_c} \leq \frac{1}{Q_c}\left(\frac{Q_t Q_c^2}{2^{2\lambda}} + \frac{2 Q_t Q_c^3}{2^{3\lambda}} + \frac{Q_t Q_c^4}{2^{4\lambda}}\right). \end{aligned}$$

For each query $H_{\mathsf{tree}}(n)$, we say Bad occurs if there exists $n' \neq n$ satisfies (2). Let $H_{\mathsf{tree}}(n) = l$, $n \oplus l = r$, $H_{\mathsf{tree}}(n') = l'$, and $n' \oplus l' = r'$. We classify the Bad into sub-events, up to the freshness of l and r.

- $\mathsf{L}_1 \Leftrightarrow l = l'$ or l is fresh. Observe that n should satisfies

$$(H_{\mathsf{tree}}(n) = l') \vee (H_{\mathsf{tree}}(n) \neq l' \wedge H_{\mathsf{com}}(l) = H_{\mathsf{com}}(l')).$$

– $\mathsf{L}_2 \Leftrightarrow l$ is not fresh and $l \neq l'$. Observe that n should satisfies

$$(n', l', H_{\mathsf{tree}}(n)) \in \mathcal{L}_1.$$

– $\mathsf{R}_1 \Leftrightarrow r = r'$ or r is fresh. Observe that n should satisfies

$$(H_{\mathsf{tree}}(n) \oplus n = r') \vee (H_{\mathsf{tree}}(n) \oplus n \neq r' \wedge H_{\mathsf{com}}(r) = H_{\mathsf{com}}(r')).$$

– $\mathsf{R}_2 \Leftrightarrow r$ is not fresh and $r \neq r'$. Observe that n should satisfies

$$(n', r', H_{\mathsf{tree}}(n) \oplus n) \in \mathcal{L}_2.$$

Note that one cannot have $l = l'$ and $r = r'$ at the same time, since it implies $n = n'$. Then, we have

$$\Pr[\mathsf{L}_1 \wedge \mathsf{R}_1] \leq \frac{3Q_t^2}{2^{2\lambda}}, \qquad \Pr[\mathsf{L}_2 \wedge \mathsf{R}_1 \wedge \neg\mathsf{Aux}_1] \leq \frac{2Q_tQ_c}{2^{2\lambda}},$$
$$\Pr[\mathsf{L}_1 \wedge \mathsf{R}_2 \wedge \neg\mathsf{Aux}_2] \leq \frac{2Q_tQ_c}{2^{2\lambda}}, \qquad \Pr[\mathsf{L}_2 \wedge \mathsf{R}_2 \wedge \neg\mathsf{Aux}_3] \leq \frac{Q_tQ_c}{2^{2\lambda}},$$

and

$$\begin{aligned}\Pr[\mathsf{Bad}] &\leq \Pr[\mathsf{Aux}] + \sum_{i,j\in[2]} \Pr[\mathsf{L}_i \wedge \mathsf{R}_j \wedge \neg\mathsf{Aux}] \\ &\leq \frac{2Q_t}{2^\lambda} + \frac{3Q_t^2 + 8Q_tQ_c}{2^{2\lambda}} + \frac{2Q_tQ_c^2}{2^{3\lambda}} + \frac{Q_tQ_c^3}{2^{4\lambda}} \leq \frac{9Q}{2^\lambda}\end{aligned}$$

provided that $Q_c + Q_t \leq Q \leq 2^{\lambda-1}$, which concludes the proof. □

Now we can prove the extractable semi-binding property of $\mathsf{RO\text{-}VSC}$ using this lemma.

Lemma 4. *Let $H_{\mathsf{com}} : \{0,1\}^* \to \{0,1\}^\lambda$ and $H_{\mathsf{tree}} : \{0,1\}^* \to \{0,1\}^\lambda$ be a random oracle. Let $\mathcal{A}$ be an arbitrary adversary that makes Q queries to the random oracles. Then $\mathcal{A}$'s u-extractable semi-binding advantage $\mathbf{Adv}^{u\text{-}\mathsf{ESB}}_{\mathsf{RO\text{-}VSC}}(\mathcal{A})$ against $\mathsf{RO\text{-}VSC}$ is bounded by*

$$\mathbf{Adv}^{u\text{-}\mathsf{ESB}}_{\mathsf{RO\text{-}VSC}}(\mathcal{A}) \leq \frac{11Q}{2^\lambda},$$

for $u = 2N\left(\frac{\lambda}{\log\lambda}\right)^2$.

Proof. Intuitively, according to Lemma 3, the probability of finding a collision in commitments derived by a non-leaf node is negligible. Furthermore, for each leaf node, the number of multi-collisions is bounded by Chernoff bound. We now proceed to formally bound the adversary's advantage.

Let Q_t be the number of queries to H_{tree} and Q_c be the number of queries to H_{com}. Without loss of generality, assume that $\mathcal{A}$ queries to random oracles with fixed salt, and we omit the salt input for each random oracle query. Let $\mathcal{Q}_t$ and $\mathcal{Q}_c$ be the collection of queries to H_{tree} and H_{com}, respectively. At the end of the game, we define $H_{\mathsf{tree}}(x) = \perp$ (resp. $H_{\mathsf{com}}(x) = \perp$) for non-queried input x to H_{tree} (resp. H_{com}), and we consider that two $\perp$'s are not identical for simplicity.

We first define the extractor $\mathsf{Ext}(\mathcal{Q}_t, \mathcal{Q}_c, \mathsf{com} = (\mathsf{com}_1, \ldots, \mathsf{com}_N))$ as follows.

1. For each $i \in [N]$, find $S_{0,i}$ such that
$$S_{0,i} = \{s : H_{\mathsf{com}}(i, s) = \mathsf{com}_i\}$$
2. For each $e \in [d-1]$ and $i \in [N/2^e]$, find $S_{e,i}$ such that
$$S_{e,i} = \{n : H_{\mathsf{tree}}(s) \in S_{e-1,2i-1}, H_{\mathsf{tree}}(s) \oplus s \in S_{e-1,2i}\}$$
3. For $i \in [N]$, let
$$A_i = \left\{(p_1, \ldots, p_d) : p_e \in S_{e-1,i_{e-1}} \text{ for } e \in [d]\right\}$$
where $i_e = (\lfloor(\bar{i}-1)/2^e\rfloor \oplus 1) + 1$ for $e \in [d-1]$
4. Let S be the set of messages, where
$$S = \Big\{(s_1, \ldots, s_N) : s_i = \bot \text{ if } S_{0,i} = \emptyset \text{ and } s_i \in S_{0,i} \text{ otherwise,}$$
$$\mathsf{Recon}(p_1, \ldots, p_d, \mathsf{com}_i, I = [N] \setminus \{i\}) = (s_i)_{i \in I} \text{ for } (p_1, \ldots, p_d) \in A_i\Big\}$$
5. Finally, Ext outputs arbitrary u or less elements in S.

We define some bad events.

– $\mathsf{Bad}_1 \Leftrightarrow$ there exists $e \in [d-1]$ and $i \in [N/2^e]$ such that $|S_{e,i}| \geq 2$.
– $\mathsf{Bad}_2 \Leftrightarrow$ there exists $i \in [N]$ such that $|S_{0,i}| \geq 2\lambda/\log\lambda$.

By Lemma 3 and (1),
$$\Pr[\mathsf{Bad}_1 \vee \mathsf{Bad}_2] \leq \frac{11Q}{2^\lambda} \tag{3}$$

In the following, we analyze the extracting condition without bad events.

– As S contains all possible (pdecom_I, I), $\mathcal{A}$ wins the game only if $|S| \geq u$.
– By $\neg\mathsf{Bad}_1$, we have $|S| \leq \sum_{i \in [N/2]} |S_{0,2i}| \cdot |S_{0,2i-1}|$. Then, by $\neg\mathsf{Bad}_2$, we have
$$|S| \leq 2N\left(\frac{\lambda}{\log\lambda}\right)^2.$$

Therefore, $\mathcal{A}$ cannot win the game without bad events so we have
$$\mathbf{Adv}^{u\text{-}\mathsf{ESB}}_{\mathsf{RO\text{-}VSC}}(\mathcal{A}) \leq \Pr[\mathsf{Bad}_1 \vee \mathsf{Bad}_2] \leq \frac{11Q}{2^\lambda}$$
provided that $u = 2N\left(\frac{\lambda}{\log\lambda}\right)^2$. □

Lemma 5. *Let $H_{\mathsf{com}} : \{0,1\}^* \to \{0,1\}^\lambda$ and $H_{\mathsf{tree}} : \{0,1\}^* \to \{0,1\}^\lambda$ be a random oracle. Let $\mathcal{A}$ be an arbitrary adversary that makes Q queries to the random oracles. Then $\mathcal{A}$'s multi-instance hiding advantage $\mathbf{Adv}^{Q_I\text{-}\mathsf{MIH}}_{\mathsf{RO\text{-}VSC}}(\mathcal{A})$ against* $\mathsf{RO\text{-}VSC}$ *is bounded by*
$$\mathbf{Adv}^{Q_I\text{-}\mathsf{MIH}}_{\mathsf{RO\text{-}VSC}}(\mathcal{A}) \leq \frac{Q_I^2}{2^{2\lambda}} + \frac{Q}{2^\lambda}.$$

Proof. Let Q_t be the number of queries to H_{tree} and Q_c be the number of queries to H_{com}. Let Q_I be the number of instances.

We will bound the advantage using the H-coefficient technique. Denote $\mathcal{I}$ as the ideal world where the hidden nodes are always replaced with random strings, and denote $\mathcal{R}$ the real world where the hidden nodes are remain unchanged. Let γ be the transcript of $\mathcal{A}$ which contains queries to the random oracles and the instances given in the game. The parent node of the hidden seed $\overline{\mathsf{node}}$ is derived by $\overline{\mathsf{node}} = m_{\bar{i}} \oplus m_{((\bar{i}-1)\oplus 1)+1}$. Now we define some events of bad transcripts as follows.

- Bad_1: two salt's in the given instance collide. Since salt is sampled uniformly at random, $\Pr[\mathsf{Bad}_1] \le Q_I^2/2^{2\lambda}$.
- Bad_2: a query $(\mathsf{salt}, m_{\bar{i}})$ is queried to H_{com}. $\Pr[\mathsf{Bad}_2 \wedge \neg\mathsf{Bad}_1] \le Q_c/2^{\lambda}$.
- Bad_3: a query $(\mathsf{salt}, \overline{\mathsf{node}})$ is queried to H_{tree}. $\Pr[\mathsf{Bad}_3 \wedge \neg\mathsf{Bad}_1] \le Q_t/2^{\lambda}$.

We say $\mathcal{T}_{\mathsf{Bad}}$ be the set of bad transcripts, while $\mathcal{T}_{\mathsf{Good}}$ be the complement of $\mathcal{T}_{\mathsf{Bad}}$, and let T_{id} (resp. T_{re}) be the distribution of γ in $\mathcal{I}$ (resp. $\mathcal{R}$). As Q random oracle queries and Q_I instances of $m_{\bar{i}}$ are included in transcripts, for $\gamma \in \mathcal{T}_{\mathsf{Good}}$,

$$\Pr[T_{\mathsf{id}} = \gamma] = \Pr[T_{\mathsf{re}} = \gamma] = \left(\frac{1}{2^{\lambda}}\right)^{Q+Q_I}.$$

So, the advantage is bounded by

$$\mathbf{Adv}^{Q_I\text{-}\mathsf{MIH}}_{\mathsf{RO\text{-}VSC}}(\mathcal{A}) \le \frac{Q_I^2}{2^{2\lambda}} + \frac{Q}{2^{\lambda}}.$$

□

3.2 Instantiation from an Ideal Cipher

Now we replace the random oracles with a ideal cipher E. We instantiate VSC from the ideal cipher, named $\mathsf{IC\text{-}VSC}$, and also prove its properties. Let $E : \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ be a ideal cipher. Then $\mathsf{IC\text{-}VSC}$ is constructed as Figure 2.

The main difference between $\mathsf{IC\text{-}VSC}$ and $\mathsf{RO\text{-}VSC}$ is that all the random oracles are replaced with the ideal cipher so that a sophisticated domain separation is required. To separate the domains of com, tape, and node, the IV salt is used in two parts: key, and input masking. We note that the new variable b represents the current repetition in the signature scheme.

Proving the extractable semi-binding and multi-instance hiding properties of $\mathsf{IC\text{-}VSC}$ is similar to that of $\mathsf{RO\text{-}VSC}$. Due to page limitations, we only provide the statement here and include the full proofs in Appendix B.

Lemma 6. *Let $E : \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ be an ideal cipher. Let $\mathcal{A}$ be an arbitrary adversary that makes Q queries to E. Then, $\mathcal{A}$'s u-extractable semi-binding advantage $\mathbf{Adv}^{u\text{-}\mathsf{ESB}}_{\mathsf{IC\text{-}VSC}}(\mathcal{A})$ against $\mathsf{IC\text{-}VSC}$ is bounded by*

$$\mathbf{Adv}^{u\text{-}\mathsf{ESB}}_{\mathsf{IC\text{-}VSC}}(\mathcal{A}) \le \frac{12Q}{2^{\lambda}},$$

– **Parameters**: an ideal cipher E, integer C, a power-of-two integer N, and $d = \log N$.
– **Inputs**: $\mathsf{salt} = (\mathsf{salt}_1, \mathsf{salt}_2, \mathsf{b}) \in \{0,1\}^{\lambda+\lambda+8}$, and $\mathsf{root} \in \{0,1\}^{\lambda}$.

– $\mathsf{Commit}^E(\mathsf{salt}, \mathsf{root})$:
1. Set $\mathsf{node}_{1,1} \leftarrow \mathsf{root}$.
2. For each level $e \in [d-1]$ and $i \in [2^e]$, set

$$\mathsf{node}_{e+1,2i-1} \leftarrow 2 \cdot \mathsf{node}_{e,i} \oplus E_{\mathsf{salt}_2}(\mathsf{node}_{e,i} \oplus \mathsf{salt}_1)$$
$$\mathsf{node}_{e+1,2i} \leftarrow \mathsf{node}_{e+1,2i-1} \oplus \mathsf{node}_{e,i}.$$

3. For $i \in [N]$, set

$$\mathsf{seed}_i \leftarrow \mathsf{node}_{d,i}$$
$$\mathsf{ctr}[b,i,0] \leftarrow 0^{\lambda-24} \parallel \mathsf{b} \parallel \langle i \rangle_B \parallel \langle 0 \rangle_B$$
$$\mathsf{com}_i \leftarrow E_{\mathsf{seed}_i}(\mathsf{ctr}[b,i,0] \oplus \mathsf{salt}_1)$$

   (a) For $j \in [2C+1]$,

$$\mathsf{ctr}[b,i,0] \leftarrow 0^{\lambda-24} \parallel \mathsf{b} \parallel \langle i \rangle_B \parallel \langle j \rangle_B$$
$$\mathsf{tape}_{i,j} \leftarrow E_{\mathsf{seed}_i}(\mathsf{ctr}[b,i,j] \oplus \mathsf{salt}_1)$$

   (b) Set $\mathsf{tape}_i \leftarrow \mathsf{tape}_{i,1} \parallel \cdots \parallel \mathsf{tape}_{i,2C+1}$ and $m_i \leftarrow \mathsf{seed}_i \parallel \mathsf{tape}_i$.
4. Output a commitment $\mathsf{com} = (\mathsf{com}_1, \ldots, \mathsf{com}_N)$ with opening information $\mathsf{decom} = ((\mathsf{node}_{e,i})_{e \in [d-1], i \in [N]}, \mathsf{com})$, and messages $(m_1, \ldots, m_N)$.

– $\mathsf{Open}^E(\mathsf{salt}, \mathsf{decom}, I = [N] \setminus \{\bar{i}\})$:
1. Set $\mathsf{path}_I \leftarrow (\mathsf{node}_{d-e+1,i_e})_{e \in [d-1]}$ where $i_e = (\lfloor (\bar{i}-1)/2^{e-1} \rfloor \oplus 1) + 1$ for $e \in [d-1]$;
2. Output $\mathsf{pdecom} = (\mathsf{path}_I, \mathsf{com}_{\bar{i}})$

– $\mathsf{Recon}^E(\mathsf{salt}, \mathsf{pdecom}, I = [N] \setminus \{\bar{i}\})$:
1. Similarly as Step 2 in Commit, expand each node in path_I and get $(\mathsf{node}_{d,i})_{i \in I}$.
2. For $i \in I$, do Step 3 in Commit^E.
3. Output $((m_i)_{i \in I}, \mathsf{com})$.

– $\mathsf{Verify}^E(\mathsf{salt}, \mathsf{com} = (\mathsf{com}_i^*)_{i \in [N]}, \mathsf{pdecom}, I = [N] \setminus \{\bar{i}\})$:
1. Similarly as Step 2 in Commit, expand each node in path_I and get $(\mathsf{node}_{d,i})_{i \in I}$.
2. For $i \in I$, do Step 3 in Commit^E.
3. Output $(m_i)_{i \in I}$ if $\mathsf{com} = \mathsf{com}_i^*$ for all $i \in I$, or output $\perp$ otherwise.

Fig. 2: IC-VSC

for $u = 2N\left(\frac{\lambda}{\log\lambda}\right)^2$.

Proof. See Appendix B for the full proof.

Lemma 7. *Let* $E : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ *be an ideal cipher. Let* $\mathcal{A}$ *be an arbitrary adversary that makes* Q *queries to* E. *Then,* $\mathcal{A}$*'s multi-instance hiding advantage* $\mathbf{Adv}^{Q_I\text{-}\mathsf{MIH}}_{\mathsf{IC\text{-}VSC}}(\mathcal{A})$ *against* $\mathsf{IC\text{-}VSC}$ *is bounded by*

$$\mathbf{Adv}^{Q_I\text{-}\mathsf{MIH}}_{\mathsf{IC\text{-}VSC}}(\mathcal{A}) \le \frac{Q_I^2}{2^{2\lambda}} + \frac{6\lambda \cdot Q}{2^\lambda \cdot \log\lambda}.$$

Proof. See Appendix B for the full proof.

4 Application of VSC to BN++

4.1 Description of Reduced BN++

In this section, we apply $\mathsf{IC\text{-}VSC}$ to BN++, dubbed *reduced BN++*. The signing and verification algorithms of the reduced BN++ can be found in Algorithm 5 and Algorithm 6 in Appendix A. The major differences between the reduced BN++ and the original BN++ can be summarized into three points:

- The GGM trees are replaced by correlated half-trees. Roots of the trees are now fixed to the secret key $\mathsf{sk} \in \{0,1\}^\lambda$. If the secret key is longer than λ bits, the roots are fixed to the most significant λ bits of the secret key.
- The sizes of each commitments are reduced from 2λ bits to λ bits.
- Most of the random oracle calls are replaced by primitives based on a ideal cipher. Now, there are only 5 calls to random oracles for a signing query.

There are also some minor changes compared to the original BN++. The following modifications are made for either minor efficiency improvements or ease of proof.

- The message to be signed is hashed by $H_0 : \{0,1\}^* \to \{0,1\}^{2\lambda}$ along with the public key.
- The salt is generated by running $H_3 : \{0,1\}^* \to \{0,1\}^{2\lambda}$ with inputs the secret key sk, hashed message μ, and internal randomness ρ, rather than randomly sampled.
- The offsets $\Delta c_k, \Delta z_{k,j}$ are added to the last share, rather than to the first share.
- The function expanding h_1 and h_2 (the original function name in BN++ is Expand) is divided into two different functions: $\mathsf{ExpandH1} : \{0,1\}^{2\lambda} \to \mathbb{F}^{C\tau}$ and $\mathsf{ExpandH2} : \{0,1\}^{2\lambda} \to [N]^\tau$. These two functions are modeled as random oracles in the security proof.

4.2 Security Proof of Reduced BN++

In this section, we prove the security of reduced BN++. The main difference between the security proofs for BN++ and reduced BN++ is that the reduced commit size no longer allows VSC to provide binding. In the original proof, there can be only a single tuple of shares corresponding to a single H_1 query. However, in the reduced version, multiple tuples can exist, and new shares can be discovered even after the H_1 query. Consequently, the proof is no longer based on limiting the probability that a tuple of shares passes a random challenge. Instead, it focuses on the difficulty an attacker faces in finding a tuple of shares that passes a given challenge within a limited number of queries. This change complicates the proof and makes it challenging to use modular proofs that follow from the properties of VSC. Therefore, we abandon modular proofs and directly exploit the internal structure of VSC to prove the security of reduced BN++.

Theorem 1 (EUF-KO Security of reduced BN++). *Let $(N, \tau, \lambda, \mathbb{F})$ be parameters of the reduced BN++ (rBN++) signature scheme where $|\mathbb{F}| = 2^\lambda$ and $N = 2^d$. Assume that H_1, $H_2 : \{0,1\}^* \to \{0,1\}^{2\lambda}$, and* Expand *are modeled as random oracles. Let $\mathcal{A}$ be an arbitrary adversary against the EUF-KO security of reduced BN++ that makes a total of Q random oracle queries and P of ideal cipher queries. Then there exist PPT adversaries*

- *$\mathcal{B}$ against OWF-security of* KeyGen *with Q random oracle queries and P ideal cipher queries,*
- *$\mathcal{C}$ against extractable semi-binding security with P ideal cipher queries,*

such that

$$\mathbf{Adv}^{\mathsf{euf\text{-}ko}}_{\mathsf{rBN++}} \leq \frac{Q^2}{2^{2\lambda}} + \frac{(4\mu+2)Q}{2^\lambda} + \frac{14P}{2^\lambda} + \mathbf{Adv}^{\mathsf{owf}}(\mathcal{B}) + \mathbf{Adv}^{u\text{-}\mathsf{ESB}}_{\mathsf{IC\text{-}VSC}}(\mathcal{C}) \\ + Q_1 \cdot \sum_{i=\tau'}^{\tau} \binom{\tau}{i} \left(\frac{u}{|\mathbb{F}|}\right)^i \left(1 - \frac{u}{|\mathbb{F}|}\right)^{\tau-i} + \frac{Q_2}{N^{\tau-\tau'}}$$

where $\mu = 2\lambda/\log\lambda$, $u = \mu^2 N/2$.

Proof. Suppose that all the queries to H_1 (resp. H_2) are listed in $\mathcal{Q}_1$ (resp. $\mathcal{Q}_2$) and let $|\mathcal{Q}_1| = Q_1$ (resp. $|\mathcal{Q}_2| = Q_2$).

We program the random oracles for $\mathcal{A}$ as in Algorithm 2, 3, 1, and 4 respectively. MultCheck in Algorithm 1 is the multiplication checking protocol in Section 2.4. Also, $\mathsf{MultCheck}_1^{(i)}$ and $\mathsf{MultCheck}_2^{(i)}$ in Algorithm 4 are the first round and second round of the multiplication checking protocol in Section 2.4 for the i-th party. Observe that the above programming does not change the output distribution of random oracles. We have

$$\Pr[\mathcal{A} \text{ wins}] \leq \Pr[\mathsf{Bad}] + \Pr[\mathcal{A} \text{ wins} \mid \neg\mathsf{Bad}]$$

where $\mathsf{Bad} = \bigvee_{i=1}^{6} \mathsf{Bad}_i$. Now we analyze each bad event as follows.

Algorithm 1. $H_1\left(\mathsf{salt}, \sigma_1 = (\mathsf{com}_k, \Delta_k)_{k\in[\tau]}\right)$:

$h_1 \xleftarrow{\$} \{0,1\}^{2\lambda}$
if $h_1 \in \mathcal{H}_1$ **then**
  Raise Bad_1 and abort
$h_1 \to \mathcal{H}_1$.
$(\epsilon_1, \ldots, \epsilon_\tau) \leftarrow \mathsf{ExpandH1}(h_1)$
$\mathsf{Succ}_1[\mathsf{salt}, h_1] \leftarrow \emptyset$
for $k \in [\tau]$ **do**
  for $(m_1, \ldots, m_N) \in \mathsf{Ext}\left(\mathsf{salt}, \mathsf{com}_k\right)$ *where* $\perp \notin (m_1, \ldots, m_N)$ **do**
    $m_N \leftarrow m_N \oplus (0^\lambda \parallel \Delta_k)$
    $\mathsf{root} \leftarrow \mathsf{msb}_\lambda\left(\sum_{i\in[N]} m_i\right)$
    if $\mathsf{KeyGen}(\mathsf{root}) = \mathsf{ct}$ **then**
      Raise Bad_2 and abort. `// Secret key is found`
    $(\alpha^{(i)}, v^{(i)})_{i\in[N]} \leftarrow \mathsf{MultCheck}\left(\epsilon_k, (m_i)_{i\in[N]}\right)$
    if $\sum_{i\in[N]} v^{(i)} = 0$ **then**
      $k \to \mathsf{Succ}[\mathsf{salt}, h_1]$ `// Multiplication checking cheated`
if $|\mathsf{Succ}_1[\mathsf{salt}, h_1]| \geq \tau'$ **then**
  Raise Bad_3 and abort. `// Too many cheated iterations`
$(\mathsf{salt}, \sigma_1, h_1) \to \mathcal{Q}_1$
Return h_1.

Algorithm 2. $\mathsf{ExpandH1}(h_1)$:

$h_1 \to \mathcal{H}_1$.
$\left((\epsilon_{k,j})_{j\in[\ell]}\right)_{k\in[\tau]} \xleftarrow{\$} (\mathbb{F}^\ell)^\tau$.
Return $\left((\epsilon_{k,j})_{j\in[\ell]}\right)_{k\in[\tau]}$.

Algorithm 3. $\mathsf{ExpandH2}(h_2)$:

$h_2 \to \mathcal{H}_2$.
$(\bar{i}_1, \bar{i}_2, \ldots, \bar{i}_\tau) \xleftarrow{\$} ([N])^\tau$.
Return $(\bar{i}_1, \bar{i}_2, \ldots, \bar{i}_\tau)$.

Algorithm 4. $H_2\left(\mathsf{salt}, h_1, \sigma_2 = \left((\alpha_k^{(i)}, v_k^{(i)})_{i\in[N],k\in[\tau]}\right)\right)$:

$h_1 \to \mathcal{H}_1$
$h_2 \xleftarrow{\$} \{0,1\}^{2\lambda}$
if $h_2 \in \mathcal{H}_2$ **then**
  Raise Bad_1 and abort.
$h_2 \to \mathcal{H}_2$, $(\mathsf{salt}, h_1, \sigma_2, h_2) \to \mathcal{Q}_2$.
if $\exists k \in [\tau]$ *such that* $\sum_{i\in[N]} v_k^{(i)} \neq 0$ **then**
  Return h_2.
if $\exists \sigma_1$ *such that* $(h_1, \sigma_1) \in \mathcal{Q}_1$ **then**
  Parse σ_1 as $(\mathsf{com}_k, \Delta_k)_{k\in[\tau]}$
  $(\epsilon_1, \ldots, \epsilon_\tau) \leftarrow \mathsf{ExpandH1}(h_1)$
  $(\bar{i}_1, \ldots, \bar{i}_\tau) \leftarrow \mathsf{ExpandH2}(h_2)$
else
  Return h_2.
$\mathsf{Succ}_2[\mathsf{salt}, h_2] \leftarrow \emptyset$.
for $k \in [\tau] \setminus \mathsf{Succ}_1[\mathsf{salt}, h_1]$ **do**
  $\mathcal{J} \leftarrow \emptyset$
  for $(m_1, \ldots, m_N) \in \mathsf{Ext}(\mathsf{salt}, \mathsf{com}_k)$ **do**
    if $m_N \neq \perp$ **then**
      $m_N \leftarrow m_N \oplus (0^\lambda \parallel \Delta_k)$
    if $\perp \notin (m_1, \ldots, m_N)$ **then**
      $\mathsf{root} \leftarrow \mathsf{msb}_\lambda\left(\sum_{i\in[N]} m_i\right)$
      if $\mathsf{KeyGen}(\mathsf{root}) = \mathsf{ct}$ **then**
        Raise Bad_2 and abort. `// Secret key is found`
    for $i \in [N]$, $\beta^{(i)} \leftarrow \mathsf{MultCheck}_1^{(i)}(\epsilon_k, m_i)$
    for $i \in [N]$, $w^{(i)} \leftarrow \mathsf{MultCheck}_2^{(i)}(\epsilon_k, m_i, \alpha_k)$
    if $\forall i \in [N]$, $(\alpha_k^{(i)}, v_k^{(i)}) = (\beta^{(i)}, w^{(i)})$ **then**
      Raise Bad_4 and abort.
    else if $\exists j$, $\forall i \in [N] \setminus \{j\}$, $(\alpha_k^{(i)}, v_k^{(i)}) = (\beta^{(i)}, w^{(i)})$ **then**
      $j \to \mathcal{J}$.
  if $|\mathcal{J}| \geq 2$ **then**
    Raise Bad_5 and abort.
  else if $\mathcal{J} = \left\{\bar{i}_k\right\}$ **then**
    $k \to \mathsf{Succ}_2[\mathsf{salt}, h_2]$
if $|\mathsf{Succ}_2[\mathsf{salt}, h_2]| \geq \tau - \tau'$ **then**
  Raise Bad_6 and abort. `// Too many cheated iterations`
Return h_2.

– Upper bounding $\Pr[\mathsf{Bad}_1]$. As $|h_1| = |h_2| = 2\lambda$, we have

$$\Pr[\mathsf{Bad}_1] \leq \frac{Q^2}{2^{2\lambda}}.$$

Note that without Bad_1, we can avoid the event for preimage/collision-finding of H_1 and H_2.

– Upper bounding $\Pr[\mathsf{Bad}_2]$. Let $\mathcal{B}$ be an adversary that use $\mathcal{A}$ as a subroutine, and checks whether Bad_2 occurs by some root. Then, the winning probability of $\mathcal{B}$ is at least $\Pr[\mathsf{Bad}_2]$, so we have

$$\Pr[\mathsf{Bad}_2] \leq \mathbf{Adv}^{\mathsf{owf}}(\mathcal{B})$$

– Upper bounding $\Pr[\mathsf{Bad}_3]$. By the definition of Ext, the number of extracted messages is always less or equal than u. Therefore, for each $k \in [\tau]$, the success probability of cheating the multiplication checking is upper bounded by $\frac{u}{|\mathbb{F}|}$ and we have

$$\Pr[\mathsf{Bad}_3] \leq Q_1 \cdot \sum_{i=\tau'}^{\tau} \binom{\tau}{i} \left(\frac{u}{|\mathbb{F}|}\right)^i \left(1 - \frac{u}{|\mathbb{F}|}\right)^{\tau-i}$$

– Upper bounding $\Pr[\mathsf{Bad}_4]$. Let us fix H_1 query and assume that Bad_4 occurs at k-th iteration with $(m_1, \ldots, m_N)$. Since $(m_1, \ldots, m_N)$ is extracted messages, there exists ideal cipher queries such that

$$E_{m_{i,0}}(\mathsf{ctr}[k, i, 0] \oplus \mathsf{salt}_1) = \mathsf{com}_i, \text{ for } i \in [N], \tag{a}$$
$$E_{m_{i,0}}(\mathsf{ctr}[k, i, j] \oplus \mathsf{salt}_1) = m_{i,j}, \text{ for } i \in [N], j \in [2C+1] \tag{b}$$

Let $a \in [N]$ be the party index such that $E_{m_{a,0}}(\mathsf{ctr}[k, a, 0] \oplus \mathsf{salt}_1) = \mathsf{com}_a$ is the last query in (a). As $k \notin \mathsf{Succ}_1[\mathsf{salt}, h_1]$, the above query is done after the query to H_1. We divide two cases, up to the order of ideal cipher queries.

- Case 1: $E_{m_{a,0}}(\mathsf{ctr}[k, a, 0] \oplus \mathsf{salt}_1) = \mathsf{com}_a$ is the last query in (a) and (b). Then, there are at most 1 candidate of $m_{a,0}$ that passes multiplication check with fixed challenge ϵ_k. Also, by (8) in Lemma 9[4], there are at most $\frac{2\lambda}{\log \lambda}$ of $(N-1)$ tuple of seeds $(m_{i,0})_{i \in [N] \setminus \{a\}}$ such satisfies (a) except with probability $14P/2^\lambda$. Since E should satisfy the equation for $m_{a,0}$, we have

$$\Pr[\mathsf{Bad}_4 \text{ with Case 1.}] \leq \frac{\mu Q_1}{2^\lambda - P} + \frac{14P}{2^\lambda} \leq \frac{2\mu Q_1}{2^\lambda} + \frac{14P}{2^\lambda}$$

provided that $P \leq 2^{\lambda-1}$.

- Case 2: $E_{m_{b,0}}(\mathsf{ctr}[k, b, j] \oplus \mathsf{salt}_1) = m_{b,j}$ is the last query in (a) and (b). Similar to Case 1, the candidate of $m_{b,j}$ is now at most 1. The difference

[4] ideal cipher based version of (3) in Lemma 4.

is, there are at most P candidates of $m_{a,0}$ and E should satisfy two equations, so we have

$$\Pr[\mathsf{Bad}_4 \text{ with Case 2.}] \leq \frac{2\mu Q_1 P}{(2^\lambda - P)^2} + \frac{14P}{2^\lambda} \leq \frac{2\mu Q_1}{2^\lambda} + \frac{14P}{2^\lambda}$$

provided that $P \leq 2^{\lambda-1}$.

All in all, we get

$$\Pr[\mathsf{Bad}_4] \leq \frac{4\mu Q_1}{2^\lambda} + \frac{14P}{2^\lambda}$$

– Upper bounding $\Pr[\mathsf{Bad}_5]$. Let us fix the query to H_2 and assume that Bad_5 occurs by two extracted messages in k-th iteration: $(m_1, \ldots, m_N)$ adds j_1 to $\mathcal{J}$and $(m'_1, \ldots, m'_N)$ adds j_2 to $\mathcal{J}$. Then, by the definition of Ext, there exists $(m''_1, \ldots, m''_N)$ in extracted message sets, such that $m''_i \in \{m_i, m'_i\}$ for $i \in [N] \setminus \{j_1, j_2\}$, $m''_{j_1} = m'_{j_1}$ and $m''_{j_2} = m_{j_2}$. As $k \notin \mathsf{Succ}_1[\mathsf{salt}, h_1]$, $(m''_1, \ldots, m''_N)$ induces Bad_4, so

$$\Pr[\mathsf{Bad}_5 \wedge \neg\mathsf{Bad}_4] = 0.$$

Since the game immediately aborts if Bad_4 occurs, we can conclude that $\Pr[\mathsf{Bad}_5] = 0$.

– Upper bounding $\Pr[\mathsf{Bad}_6]$. Since $|\mathcal{J}| \leq 1$ for iterations in all queries to H_2, we have

$$\Pr[\mathsf{Bad}_6] \leq \frac{Q_2}{N^{\tau-\tau'}}$$

All in all, we have

$$\begin{aligned}\Pr[\mathsf{Bad}] \leq & \mathbf{Adv}^{\mathsf{owf}}(\mathcal{B}) + \frac{Q^2}{2^{2\lambda}} + \frac{4\mu Q_1}{2^\lambda} + \frac{14P}{2^\lambda} \\ & + Q_1 \cdot \sum_{i=\tau'}^{\tau} \binom{\tau}{i} \left(\frac{u}{|\mathbb{F}|}\right)^i \left(1 - \frac{u}{|\mathbb{F}|}\right)^{\tau-i} + \frac{Q_2}{N^{\tau-\tau'}}\end{aligned} \tag{4}$$

What is left is to upper bound $\Pr[\mathcal{A} \text{ wins} \mid \neg\mathsf{Bad}]$. Suppose that $\mathcal{A}$ outputs valid forgery

$$\left(\mathsf{salt}, h_1, h_2, (\mathsf{pdecom}_k, \Delta_k, \alpha_k^{(\bar{i}_k)})_{k\in[\tau]}\right)$$

but Bad does not occurs. Then, the forgery should satisfy followings.

– There exists $\sigma_1 = (\mathsf{com}_k, \Delta_k)_{k\in[\tau]}$ such that $(\mathsf{salt}, \sigma_1, h_1) \in \mathcal{Q}_1$.
– There exists $\sigma_2 = (\alpha_k^{(i)}, v_k^{(i)})_{i\in[N],k\in[\tau]}$ such that $(\mathsf{salt}, h_1, \sigma_2, h_2) \in \mathcal{Q}_2$.
– Let $\mathsf{Fail} = [\tau] \setminus (\mathsf{Succ}_1[\mathsf{salt}, h_1] \cup \mathsf{Succ}_2[\mathsf{salt}, h_1, h_2])$. Then, $\mathsf{Fail} \neq \emptyset$.
– Fix $a \in \mathsf{Fail}$, and let

$$\begin{aligned}\mathsf{ExpandH1}(h_1) &= (\epsilon_k)_{k\in[\tau]}, \\ \mathsf{ExpandH2}(h_2) &= (\bar{i}_k)_{k\in[\tau]}, \\ \mathsf{Recon}(\mathsf{salt}, \mathsf{pdecom}_a, I_a) &= ((m_i)_{i\in I_a}, \mathsf{com})\end{aligned}$$

where $m_{\bar{i}_a} = \perp$ and $I_a = [N] \setminus \{\bar{i}_a\}$. Then, we have

$$\mathsf{Verify}(\mathsf{salt}, \mathsf{com}, \mathsf{pdecom}_a, I_a) = (m_i)_{i \in I_a}.$$

If $(m_1, \ldots, m_N) \notin \mathsf{Ext}(\mathsf{com}, \Delta_a)$, one can directly construct adversary $\mathcal{C}$ against semi-binding security of VSC using $\mathcal{A}$ as subroutine, so

$$\Pr[\mathcal{A} \text{ wins } \wedge (m_1, \ldots, m_N) \notin \mathsf{Ext}(\mathsf{com}, \Delta_a) \mid \neg \mathsf{Bad}] \leq \mathbf{Adv}^{\text{u-ESB}}(\mathcal{C}). \quad (5)$$

Now assume that $(m_1, \ldots, m_N) \in \mathsf{Ext}(\mathsf{com}, \Delta_a)$ By the definition of Ext, there exists ideal cipher queries such that

$$E_{m_{i,0}}(\mathsf{ctr}[a, i, 0] \oplus \mathsf{salt}_1) = \mathsf{com}_i, \text{ for } i \in I_a, \quad \text{(c)}$$

Let $b \in [N]$ be the party index such that $E_{m_{b,0}}(\mathsf{ctr}[k, b, 0] \oplus \mathsf{salt}_1) = \mathsf{com}_b$ is the last query in (c). As $a \in \mathsf{Fail}$, the above query is done after the query to H_2. Also, there exists ideal cipher queries such that

$$E_{m_{b,0}}(\mathsf{ctr}[a, b, j] \oplus \mathsf{salt}_1) = m_{b,j}, \text{ for } j \in [2C+1]. \quad \text{(d)}$$

and let $c \in [\ell]$ be the circuit index such that

$$E_{m_{b,c}}(\mathsf{ctr}[a, b, c] \oplus \mathsf{salt}_1) = \begin{cases} \mathsf{com}_b & \text{if } c = 0 \\ m_{b,c} & \text{otherwise} \end{cases}$$

is the last query in (d). Since m_b should satisfy the conditions

$$\mathsf{MultCheck}_1^{(b)}(\epsilon_a, m_b) = \alpha_a^{(b)}$$
$$\mathsf{MultCheck}_2^{(b)}(\epsilon_a, m_b, \alpha_a) = v_a^{(b)}$$

there are at most 1 candidate of $m_{b,c}$. Therefore, we have

$$\Pr[\mathcal{A} \text{ wins} \wedge (m_1, \ldots, m_N) \in \mathsf{Ext}(\mathsf{com}, \Delta_a)] \leq \frac{Q_2}{2^\lambda - P} \leq \frac{2Q_2}{2^\lambda}. \quad (6)$$

We conclude the proof by combining 4, 5 and 6. □

Theorem 2 (EUF-CMA Security of Reduced BN++). *Assume that H_0, H_1, H_2, and* Expand *are modeled as random oracles and that the (N, τ, λ) parameters of reduced BN++ are appropriately chosen where $|\mathbb{F}| = 2^\lambda$. Let* $\mathsf{IC\text{-}VSC}$ *be a correlated vector semi-commitment based on the ideal cipher E which is multi-instance hiding. For a PPT adversary $\mathcal{A}$ against the EUF-CMA security of reduced BN++ with a total of Q_{sig} signing oracle queries, Q random oracle, and P ideal cipher queries, there exist PPT adversaries*

– *$\mathcal{B}$ against the EUF-KO security of reduced BN++*[5]*,*

[5] We assume $\mathcal{B}$ has the same amount of queries to random oracles.

– $\mathcal{C}$ *against the PRF security of* H_3[6],
– $\mathcal{D}$ *against the hiding security of* IC-VSC

such that

$$\mathbf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{rBN++}}(\mathcal{A}) \leq \frac{(Q_{\rm sig}+Q)^2}{2^{2\lambda}} + \frac{2Q_{\rm sig}(Q_{\rm sig}+P)}{2^{2\lambda}} + Q_{\rm sig}\cdot \mathbf{Adv}^{\mathsf{prf}}_{H_3}(\mathcal{C}) + \mathbf{Adv}^{Q_{\rm sig}\text{-}\mathsf{MIH}}_{\mathsf{IC\text{-}VSC}}(\mathcal{D}) + \mathbf{Adv}^{\mathsf{euf\text{-}ko}}_{\mathsf{rBN++}}(\mathcal{B})$$

Proof. Let $\mathcal{A}$ be an EUF-CMA adversary against reduced BN++ for given $(\mathsf{iv}, \mathsf{ct})$. Let G_0 be the original EUF-CMA game. Let $\mathcal{O}_{\rm sig}$ be the signing oracle, and let Q_i for $i = 0, 1, 2$ be the number of queries made to H_i by $\mathcal{A}$. We begin to prove the security of the deterministic version of reduced BN++ ($\rho \leftarrow 0^n$), and prove that of the probabilistic version later. Without loss of generality, we assume that all messages in signing queries are distinct.

G_1: This game acts the same as G_0 except that it aborts if there exist two different queries on H_0 with the same outputs. As the output length of H_0 is 2λ, we have
$$\Pr[\mathsf{G}_1 \text{ aborts}] \leq \frac{(Q_{\rm sig}+Q_0)^2}{2^{2\lambda}}.$$

G_2: $\mathcal{O}_{\rm sig}$ replaces $\mathsf{salt} \in \{0,1\}^{2\lambda}$ and root seeds $(\mathsf{seed}_k)_{k\in[\tau]} \in (\{0,1\}^{\lambda})^{\tau}$ with randomly sampled values, instead of computing $H_3(\mathsf{pt}, \mu, \rho)$. As μ is always distinct for each query, the difference between this game and the previous one reduces to the PRF security of H_3 with secret key pt. Therefore, there exists a PPT adversary $\mathcal{C}$ against the PRF security of H_3 such that
$$|\Pr[\mathcal{A} \text{ wins } \mathsf{G}_1] - \Pr[\mathcal{A} \text{ wins } \mathsf{G}_2]| \leq Q_{\rm sig}\cdot \mathbf{Adv}^{\mathsf{prf}}_{H_3}(\mathcal{C}).$$

G_3: $\mathcal{O}_{\rm sig}$ samples $h_1 \in \{0,1\}^{2\lambda}$ at random instead of computing
$$H_1(\mu, \mathsf{salt}, (\mathsf{com}_k, (\mathsf{pk}_k^{(i)})_{i\in[N]}, \Delta c_k, (\Delta z_{k,j})_{j\in[C]})_{k\in[\tau]})$$
and programs the random oracle H_1 to output h_1 for the respective query. The first challenge $(\epsilon_{k,j})_{k\in[\tau],j\in[\ell+1]}$ is derived by expanding the randomly sampled h_1. The simulation is aborted if the queries to H_1 have been made previously in a signing oracle query. As $\mathsf{salt} \in \{0,1\}^{2\lambda}$ is random, this game is indistinguishable from the previous game unless the simulation is aborted, and the probability of abort is
$$\Pr[\mathsf{G}_3 \text{ aborts}] \leq \frac{Q_{\rm sig}(Q_{\rm sig}+Q_1)}{2^{2\lambda}}.$$

G_4: $\mathcal{O}_{\rm sig}$ now samples $h_2 \in \{0,1\}^{2\lambda}$ at random instead of computing
$$H_2(h_1, \mathsf{salt}, ((\alpha_k^{(i)})_{i\in[N]}, (v_k^{(i)})_{i\in[N]})_{k\in[\tau]})$$

[6] H_3 itself is not a PRF, but it is used as a PRF with key prepending. We use this notation for convenience.

and also program the random oracle H_2 to output h_2 for the respective query. In this game, both h_1 and h_2 are sampled in advance, and all the derived values are computed from h_1 and h_2. After computing all such values, $\mathcal{O}_{\text{sig}}$ program the H_1 oracle and the H_2 oracle. The simulation is aborted if the queries to H_2 have been made previously in a signing oracle query. As $h_1 \in \{0,1\}^{2\lambda}$ is random, this game is indistinguishable with the previous game unless the simulation is aborted, and the probability of abort is

$$\Pr[\mathsf{G}_4 \text{ aborts}] \leq \frac{Q_{\text{sig}}(Q_{\text{sig}} + Q_2)}{2^{2\lambda}}.$$

G_5: $\mathcal{O}_{\text{sig}}$ replaces the seed of the unopened parties $\mathsf{seed}_k^{(\bar{i}_k)} \| \mathsf{tape}_k^{(\bar{i}_k)}$ in the $\mathsf{IC\text{-}VSC}$ with a random element for each $k \in [\tau]$. Since $\bar{i}_k$'s are all random, the difference between this game and the previous one reduces to the hiding game of VSC. Then, there exists a PPT adversary $\mathcal{D}$ against the hiding game of VSC such that

$$|\Pr[\mathcal{A} \text{ wins } \mathsf{G}_4] - \Pr[\mathcal{A} \text{ wins } \mathsf{G}_5]| \leq \mathbf{Adv}_{\mathsf{IC\text{-}VSC}}^{Q_{\text{sig}}\text{-}\mathsf{MIH}}(\mathcal{D}).$$

G_6: $\mathcal{O}_{\text{sig}}$ replaces $\mathsf{com}_k^{(\bar{i}_k)}$ with randomly sampled elements for each k. The difference between this game and the previous one reduces to indistinguishability from uniform random. We prove the indistinguishability using the H-coefficient technique. Let T be a transcript of signing oracle queries of the form $(\mathsf{salt}, \mathsf{ctr}, \mathsf{com}_k^{(\bar{i}_k)})$ and ideal cipher queries of the form (K, X, Y) such that $E_K(X) = Y$. After the distinguishing game, we will give $\mathsf{seed}_k^{(\bar{i}_k)}$ to compute the probability easily. We define the bad events as follows.

- Bad_1: $\mathsf{salt}_2 = K$ and $\mathsf{ctr} \oplus \mathsf{salt}_1 = X$. The probability in the G_6 is

$$\Pr[\mathsf{Bad}_1] \leq \frac{Q_{\text{sig}}(Q_{\text{sig}} + P)}{2^{2\lambda}}.$$

- Bad_2: $\mathsf{salt}_2 = K$ and $\mathsf{com}_k^{(\bar{i}_k)} = Y$. The probability in the G_6 is

$$\Pr[\mathsf{Bad}_2] \leq \frac{Q_{\text{sig}}(Q_{\text{sig}} + P)}{2^{2\lambda}}.$$

Let $\mathcal{T}_{\mathsf{Bad}}$ be the set of transcripts with bad events and $\mathcal{T}_{\mathsf{Good}}$ is complement of $\mathcal{T}_{\mathsf{Bad}}$. Let T_{G_5} and T_{G_6} be the distribution of transcripts in G_5 and G_6, respectively. Then,

$$\Pr\left[T_{\mathsf{G}_6} \in \mathcal{T}_{\mathsf{Bad}}\right] \leq \frac{2Q_{\text{sig}}(Q_{\text{sig}} + P)}{2^{2\lambda}}$$

and for $\gamma \in \mathcal{T}_{\mathsf{Good}}$,

$$\Pr[\gamma = T_{\mathsf{G}_5}] = \frac{1}{(2^{\lambda})^{Q_{\text{sig}}}} \cdot \prod_{\mathsf{seed}} \frac{1}{(2^{\lambda})_{P_{\mathsf{seed}}}} \geq \prod_{\mathsf{seed}} \frac{1}{(2^{\lambda})_{P'_{\mathsf{seed}} + P_{\mathsf{seed}}}} = \Pr[\gamma = T_{\mathsf{G}_5}]$$

where P_{seed} is the number of ideal cipher queries with same seed, and P'_{seed} is the number of ideal cipher queries involved in the signing oracle queries. We note that $\sum_{\mathsf{seed}} P_{\mathsf{seed}} = P$ and $\sum_{\mathsf{seed}} P'_{\mathsf{seed}} = Q_{\mathrm{sig}}$. By Lemma 2, the distinguishing advantage between two games is bounded by

$$|\Pr[\mathcal{A} \text{ wins } \mathsf{G}_5] - \Pr[\mathcal{A} \text{ wins } \mathsf{G}_6]| \leq \frac{2Q_{\mathrm{sig}}(Q_{\mathrm{sig}} + P)}{2^{2\lambda}}.$$

G_7: $\mathcal{O}_{\mathrm{sig}}$ replaces

$$(\Delta c_k, (\Delta t_{k,j})_{j\in[C]})_{k\in[\tau]}$$

with random elements instead of computing them using pt and S-box outputs. As $\left((t_{k,j}^{(\bar{i}_k)})_{j\in[\ell]}, c_k^{(\bar{i}_k)}\right)_{k\in[\tau]}$ is random, the distribution of these variables does not change.

Note that now for all $k \in [\tau]$, $(\alpha_k^{(\bar{i}_k)})_{k\in[\tau]}$ is random and independent of pt. If the multiplication triple is wrong, then $v_k^{(\bar{i}_k)} \leftarrow -\sum_{i\neq\bar{i}_k} v_k^{(i)}$ is different from an honest value derived from legitimate calculation. However, $(\bar{i}_k)$ is unopened and the multiplication check is still passed. Since the signature oracle in G_7 does not depend on the secret key pt, it implies that G_7 can be reduced to the EUF-KO security. Therefore, there exists a PPT adversary $\mathcal{B}$ on EUF-KO security against reduced BN++ such that

$$\Pr[\mathcal{A} \text{ wins } \mathsf{G}_7] \leq \mathbf{Adv}^{\mathsf{euf\text{-}ko}}_{\mathsf{rBN++}}(\mathcal{B}).$$

All in all, we have

$$\begin{aligned}
\mathbf{Adv}^{\mathsf{euf\text{-}cma}}_{\mathsf{rBN++}}(\mathcal{A}) \leq\ & \frac{(Q_{\mathrm{sig}} + Q_0)^2}{2^{2\lambda}} + Q_{\mathrm{sig}} \cdot \mathbf{Adv}^{\mathsf{prf}}_{H_3}(\mathcal{C}) \\
& + \frac{Q_{\mathrm{sig}}(Q_{\mathrm{sig}} + Q_1)}{2^{2\lambda}} + \frac{Q_{\mathrm{sig}}(Q_{\mathrm{sig}} + Q_2)}{2^{2\lambda}} + \mathbf{Adv}^{Q_{\mathrm{sig}}\text{-}\mathsf{MIH}}_{\mathsf{IC\text{-}VSC}}(\mathcal{D}) \\
& + \frac{2Q_{\mathrm{sig}}(Q_{\mathrm{sig}} + P)}{2^{2\lambda}} + \mathbf{Adv}^{\mathsf{euf\text{-}ko}}_{\mathsf{rBN++}}(\mathcal{B}) \\
\leq\ & \frac{(Q_{\mathrm{sig}} + Q)^2}{2^{2\lambda}} + \frac{2Q_{\mathrm{sig}}(Q_{\mathrm{sig}} + P)}{2^{2\lambda}} + Q_{\mathrm{sig}} \cdot \mathbf{Adv}^{\mathsf{prf}}_{H_3}(\mathcal{C}) \\
& + \mathbf{Adv}^{Q_{\mathrm{sig}}\text{-}\mathsf{MIH}}_{\mathsf{IC\text{-}VSC}}(\mathcal{D}) + \mathbf{Adv}^{\mathsf{euf\text{-}ko}}_{\mathsf{rBN++}}(\mathcal{B})
\end{aligned}$$

provided that $Q_0 + Q_1 + Q_2 \leq Q$.

For the non-deterministic version of $\mathcal{A}$, all games are defined in a manner almost identical to the deterministic version, with the exception of handling two queries to $\mathcal{O}_{\mathrm{sig}}$ that involve the same messages and ρ values. If (m, ρ) are identical in two queries, the outputs must also be identical; thus, we avoid random sampling and use already programmed outputs for the random oracles in such cases. Consequently, the differences between the adjacent games remain unchanged from the deterministic version, leading to the same bounds on the advantage of $\mathcal{A}$. $\square$

4.3 Parameters and Efficiency

The parameters N and τ can be chosen to prevent the soundness attack [20]. As in the EUF-KO security proof, since the vector semi-commitment allows u valid partial decommitment information, the complexity of the soundness attack is slightly different from the original BN++ paper. The total complexity of the attack $\mathcal{C}$ is computed as

$$\begin{aligned} P_1 &= \sum_{k=\tau'}^{\tau} \binom{\tau}{k} \left(\frac{u}{|\mathbb{F}|}\right)^k \cdot \left(1 - \frac{u}{|\mathbb{F}|}\right)^{\tau-k} \\ P_2 &= \frac{1}{N^{\tau-\tau'}} \\ \mathcal{C} &= \min_{0 \le \tau' \le \tau} (1/P_1 + 1/P_2). \end{aligned}$$

If the field size $|\mathbb{F}|$ is large enough, then τ' remains unchanged and it implies (N, τ) in reduced BN++ is same as the original BN++. For a small field, τ may be required to be much larger.

Efficiency improvements of reduced BN++ can be explained by the reduced number of random oracle calls and reduced signature size. In the literature, random oracles are implemented as a comparatively heavy hash functions such as SHAKE, whereas pseudorandom generators and ideal ciphers are implemented using the AES block cipher. In reduced BN++, the random oracle calls for commitments are translated into ideal cipher calls, and PRG calls for GGM trees are translated into a halved number of ideal cipher calls. Signature size is reduced by $2\tau\lambda$ bits due to $\Delta\mathsf{sk}$ and reduced commitment size. The results are summarized in Table 1. In this table, it is assumed that the secret key size is λ bits.

Scheme	Field Size	N	τ	RO call	PRG or IC call	Sig. size (B)
BN++	2^{128}	16	33	532	$1056C + 1518$	$1056C + 3792$
	2^{128}	256	17	4356	$8704C + 13022$	$544C + 3088$
	2^{64}	16	34	548	$544C + 1564$	$544C + 3904$
	2^{64}	256	18	4612	$4608C + 13788$	$288C + 3264$
Reduced BN++	2^{128}	16	33	5	$1056C + 1551$	$1056C + 2736$
	2^{128}	256	17	5	$8704C + 13039$	$544C + 2544$
	2^{64}	16	34	5	$544C + 1598$	$544C + 2816$
	2^{64}	256	18	5	$4608C + 13806$	$288C + 2688$

Table 1: Parameter sets for 128 bit security and the number of calls to the random oracles and the ideal cipher. Repeated multiplier is not applied for both schemes. C is the number of $\mathbb{F}$-multiplications in the targeted circuit.

5 Performance

As many MPCitH-based signature schemes (specifically the first round candidates in the NIST call for additional post-quantum signatures) have similar forms with BN++, our improvements can be applied (possibly with some tweaks). However, our improvements may not be always applied in the best way for efficiency; if the probability of passing the first challenge is tight enough before applying our improvements (e.g., SDitH-L1-hyp [26] has the probability $2^{-71.2}$), the application may require larger τ for security. We choose AIMer v2.0 [24] for the performance measurement since it is the best scheme for efficiency improvement. We will call it *reduced* AIMer. For application to similar non-interactive zero-knowledge proofs such as Threshold-Computation-in-the-Head (TCitH) [15] and VOLE-in-the-Head (VOLEitH) [4], we leave it as a future work.

Environment. We developed reduced AIMer in C, with AVX2 and AES-NI instructions. A large part of our implementation is taken from the AIMer v2.0 source code.[7] For other schemes, we used packages that have been officially submitted to NIST PQC standardization project, and all the packages were compiled using the default options in the compilation scripts. Our experiments were measured in a single thread of AMD Ryzen Threadripper PRO 5995WX 64-cores with 128GB memory. For a fair comparison, we measure the execution time for each signature scheme on the same CPU using the `taskset` command.

In Table 2, we compare the performance of reduced AIMer with various post-quantum signature schemes. We measured all the benchmarks of listed schemes in the same environment, and the table only contains publicly available implementations. It lists different schemes along with their respective public key sizes ($|pk|$), signature sizes ($|sig|$), signing times (Sign), and verification times (Verify). The sizes are provided in bytes (B), while the times for signing and verification are given in kilo-cycles (Kc). The table includes NIST-selected schemes Dilithium2, and SPHINCS$^+$, as well as the first round MPCitH-based candidates of the NIST call for additional signatures like SDitH, FAEST, and AIMer.

From the data, we observe significant improvements in reduced AIMer compared to AIMer v2.0. Reduced AIMer enjoys up to 109% faster signing and 112% faster verification, as well as up to 18% smaller signature sizes. Compare to other MPCitH-based signature schemes, reduced AIMer enjoys the fastest signing and verification times and smallest signature sizes. When compared with selected algorithms, reduced AIMer shows significant superiority in performance compared to SPHINCS$^+$, while it is quite inefficient compared to Dilithium.

Application of One-tree Technique. Recently, Baum et al. introduced *batched all-but-one vector commitment (BAVC)* [2] which consists of

1. a single large GGM tree containing all the τN seeds,
2. a proof-of-work mechanism.

[7] The source codes are retrieved from `https://aimer-signature.org`

Scheme	$\|pk\|$ (B)	$\|sig\|$ (B)	Sign (Kc)	Verify (Kc)
Dilithium2 [25]	1,312	2,420	162	57
SPHINCS$^+$-128f* [18]	32	17,088	38,216	2,158
SPHINCS$^+$-128s* [18]	32	7,856	748,053	799
SDitH-Hypercube-gf256 [26]	132	8,496	20,820	10,935
FAEST-128f [3]	32	6,336	2,387	2,344
FAEST-128s [3]	32	5,006	20,926	20,936
AIMer-v2.0-128f [24]	32	5,888	788	752
AIMer-v2.0-128s [24]	32	4,160	5,926	5,812
Reduced AIMer-128f	32	4,848	421	395
Reduced AIMer-128s	32	3,632	2,826	2,730

*: -SHAKE256-simple

Table 2: Performance Comparison of post-quantum signature schemes.

For the first one, the single large GGM tree generates all the τN seeds and reveals all-but-τ seeds. This technique has the effect of reducing average-case (but not worst-case) signature size. Unfortunately, this technique cannot be applied to the reduced BN++ scheme since the reduced BN++ scheme requires each root of the τ trees to be fixed.

For the second one, the last challenge hash (H_2 in the reduced BN++ scheme) checks whether the last w bits are all zero. If the bits are not all zero, the prover calls the hash once more with an increased counter, which is called *proof-of-work*. The prover includes the counter in the signature for the verifier to verify without the proof-of-work. Fortunately, this technique can be directly applied to the reduced BN++ scheme. In Table 3, we summarize the increased number of random oracle calls and reduced signature size for a reasonable amount of proof-of-work. Unlike FAEST, since the reduced BN++ scheme calls almost no random oracle per signature, the proof-of-work mechanism may incur some computation overhead.

N	τ	w	RO call	IC call	Sig. size
16	33	0	5	3663	4848
16	32	4	21	3552	4706
16	31	8	271	3441	4562
256	17	0	5	30447	3634
256	16	8	271	28656	3426

Table 3: The number of calls to the random oracles and the ideal cipher for reduced AIMer when the proof-of-work is applied. w is the number of bits for the proof-of-work. We assume that the counter is 2-byte long.

References

1. Aguilar-Melchor, C., Gama, N., Howe, J., Hülsing, A., Joseph, D., Yue, D.: The return of the SDitH. In: EUROCRYPT 2023. pp. 564–596. Springer (2023)
2. Baum, C., Beullens, W., Mukherjee, S., Orsini, E., Ramacher, S., Rechberger, C., Roy, L., Scholl, P.: One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures. Cryptology ePrint Archive, Paper 2024/490 (2024), https://eprint.iacr.org/2024/490
3. Baum, C., Braun, L., de Saint Guilhem, C.D., Klooß, M., Majenz, C., Mukherjee, S., Orsini, E., Ramacher, S., Rechberger, C., Roy, L., Scholl, P.: FAEST. Technical report, National Institute of Standards and Technology, 2023 (2023), available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures
4. Baum, C., Braun, L., de Saint Guilhem, C.D., Klooß, M., Orsini, E., Roy, L., Scholl, P.: Publicly Verifiable Zero-Knowledge and Post-Quantum Signatures from VOLE-in-the-Head. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023. pp. 581–615. Springer Nature Switzerland, Cham (2023)
5. Baum, C., Nof, A.: Concretely-Efficient Zero-Knowledge Arguments for Arithmetic Circuits and Their Application to Lattice-Based Cryptography. In: PKC 2020. pp. 495–526. Springer (2020)
6. Baum, C., Saint Guilhem, C.D.d., Kales, D., Orsini, E., Scholl, P., Zaverucha, G.: Banquet: Short and fast signatures from AES. In: PKC 2021. pp. 266–297. Springer (2021)
7. Beullens, W.: Sigma protocols for mq, pkp and sis, and fishy signature schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020. pp. 183–211. Springer International Publishing, Cham (2020)
8. Bui, D., Carozza, E., Couteau, G., Goudarzi, D., Joux, A.: Short Signatures from Regular Syndrome Decoding, Revisited. Cryptology ePrint Archive, Paper 2024/252 (2024), https://eprint.iacr.org/2024/252
9. Bui, D., Cong, K., de Saint Guilhem, C.D.: Improved all-but-one vector commitment with applications to post-quantum signatures. Cryptology ePrint Archive, Paper 2024/097 (2024), https://eprint.iacr.org/2024/097
10. Cui, H., Liu, H., Yan, D., Yang, K., Yu, Y., Zhang, K.: ResolveD: Shorter signatures from regular syndrome decoding and vole-in-the-head. In: PKC 2024. pp. 229–258. Springer (2024)
11. Dobraunig, C., Kales, D., Rechberger, C., Schofnegger, M., Zaverucha, G.: Shorter Signatures Based on Tailor-Made Minimalist Symmetric-Key Crypto. In: ACM CCS 2022. pp. 843–857. Association of Computing Machinery (November 2022), https://www.microsoft.com/en-us/research/publication/shorter-signatures-based-on-tailor-made-minimalist-symmetric-key-crypto/
12. Don, J., Fehr, S., Majenz, C.: The Measure-and-Reprogram Technique 2.0: Multi-Round Fiat-Shamir and More. In: CRYPTO 2020. p. 602–631. Springer (2020)
13. Feneuil, T.: Building MPCitH-based Signatures from MQ, MinRank, Rank SD and PKP. Cryptology ePrint Archive (2022)
14. Feneuil, T., Joux, A., Rivain, M.: Syndrome Decoding in the Head: Shorter Signatures from Zero-Knowledge Proofs. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022. pp. 541–572. Springer Nature Switzerland, Cham (2022)
15. Feneuil, T., Rivain, M.: Threshold Computation in the Head: Improved Framework for Post-Quantum Signatures and Zero-Knowledge Arguments. Cryptology ePrint Archive, Paper 2023/1573 (2023), https://eprint.iacr.org/2023/1573

16. Goldreich, O., Goldwasser, S., Micali, S.: How to Construct Random Functions. J. ACM **33**(4), 792–807 (aug 1986). https://doi.org/10.1145/6490.6503
17. Guo, X., Yang, K., Wang, X., Zhang, W., Xie, X., Zhang, J., Liu, Z.: Half-tree: Halving the cost of tree expansion in cot and dpf. In: EUROCRYPT 2023. pp. 330–362. Springer (2023)
18. Hulsing, A., Bernstein, D.J., Dobraunig, C., Eichlseder, M., Fluhrer, S., Gazdag, S.L., Kampanakis, P., Kolbl, S., Lange, T., Lauridsen, M.M., Mendel, F., Niederhagen, R., Rechberger, C., Rijneveld, J., Schwabe, P., Aumasson, J.P., Westerbaan, B., Beullens, W.: SPHINCS+. Technical report, National Institute of Standards and Technology, 2022 (2022), available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`
19. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from Secure Multiparty Computation. In: ACM STOC 2007. pp. 21–30 (2007)
20. Kales, D., Zaverucha, G.: An Attack on Some Signature Schemes Constructed from Five-Pass Identification Schemes. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) Cryptology and Network Security. pp. 3–22. Springer International Publishing, Cham (2020)
21. Kales, D., Zaverucha, G.: Efficient Lifting for Shorter Zero-Knowledge Proofs and Post-Quantum Signatures. Cryptology ePrint Archive, Paper 2022/588 (2022), `https://eprint.iacr.org/2022/588`
22. Katz, J., Kolesnikov, V., Wang, X.: Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures. In: ACM CCS 2018. pp. 525–537. ACM (2018)
23. Kim, S., Ha, J., Son, M., Lee, B., Moon, D., Lee, J., Lee, S., Kwon, J., Cho, J., Yoon, H., Lee, J.: Aim: Symmetric primitive for shorter signatures with stronger security. In: Proceedings of the 2023 ACM SIGSAC Conference on Computer and Communications Security. p. 401–415. CCS '23, Association for Computing Machinery, New York, NY, USA (2023). https://doi.org/10.1145/3576915.3616579
24. Lee, J., Cho, J., Ha, J., Kim, S., Kwon, J., Lee, B., Lee, J., Lee, S., Moon, D., Son, M., Yoon, H.: The AIMer Signature Scheme (2024), version 2.0, available at `https://kpqc.or.kr/competition_02.html`
25. Lyubashevsky, V., Ducas, L., Kiltz, E., Lepoint, T., Schwabe, P., Seiler, G., Stehlé, D., Bai, S.: CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022 (2022), available at `https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022`
26. Melchor, C.A., Feneuil, T., Gama, N., Gueron, S., Howe, J., Joseph, D., Joux, A., Persichetti, E., Randrianarisoa, T.H., Rivain, M., Yue, D.: The Syndrome Decoding in the Head (SD-in-the-Head) Signature Scheme (2023), available at `https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures`
27. de Saint Guilhem, C.D., Orsini, E., Tanguy, T.: Limbo: Efficient Zero-Knowledge MPCitH-Based Arguments. In: ACM CCS 2021. p. 3022–3036. Association for Computing Machinery (2021)
28. Zaverucha, G., Chase, M., Derler, D., Goldfeder, S., Orlandi, C., Ramacher, S., Rechberger, C., Slamanig, D., Katz, J., Wang, X., Kolesnikov, V., Kales, D.: Picnic. Technical report, National Institute of Standards and Technology, 2020 (2022), available at `https://csrc.nist.gov/projects/post-quantum-cryptography/round-3-submissions`

A Detailed Description of the reduced BN++ Signature Scheme

Algorithm 5. $\mathsf{Sign}(\mathsf{sk}, \mathsf{pk}, m)$ - reduced BN++, signing algorithm.

`// Phase 1: Committing to the views of the parties.`

Compute the hash of the message: $\mu \leftarrow H_0(\mathsf{pk}, m)$

Sample randomness: $\rho \leftarrow_{\$} \{0,1\}^{\lambda}$ ($\rho \leftarrow 0^{\lambda}$ for deterministic signature)

Compute salt: $\mathsf{salt} \leftarrow H_3(\mathsf{sk}, \mu, \rho)$.

for *each repetition* $k \in [\tau]$ **do**

- $(\mathsf{com}_k, \mathsf{decom}_k, (\mathsf{seed}_k^{(i)} \| \mathsf{tape}_k^{(i)})_{i \in [N]}) \leftarrow \mathsf{IC\text{-}VSC.Commit}((\mathsf{salt} \| \langle \tau \rangle_B, k), \mathsf{sk})$.
- **for** *each gate g with index j* **do**
  - **if** *g is an addition with inputs* $(x_{k,j}, y_{k,j})$ **then**
    - For each party i, set the output share of $z_{k,j}^{(i)} = x_{k,j}^{(i)} + y_{k,j}^{(i)}$.
  - **if** *g is a multiplication with inputs* $(x_{k,j}, y_{k,j})$ **then**
    - For each party i, sample $z_{k,j}^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)})$.
    - Compute output offset and adjust the last share: $\Delta z_{k,j} = z_{k,j} - \sum_i z_{k,j}^{(i)}$, $z_{k,j}^{(N)} \leftarrow z_{k,j}^{(N)} + \Delta z_{k,j}$.
    - For each party i, sample $a_{k,j}^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)})$.
    - Compute $a_{k,j} = \sum_i a_{k,j}^{(i)}$ and set $b_{k,j} = y_{k,j}$.
- Compute $c_k = \sum_j a_{k,j} \cdot b_{k,j}$.
- For each party i, sample $c_k^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)})$.
- Compute offset and adjust the last share : $\Delta c_k = c_k - \sum_i c_k^{(i)}$, $c_k^{(N)} \leftarrow c_k^{(N)} + \Delta c_k$.

Set $\sigma_1 \leftarrow (\mathsf{salt}, (\mathsf{com}_k, \Delta c_k, (\Delta z_{k,j})_{j \in [C]})_{k \in [\tau]})$.

`// Phase 2: Challenging the checking protocol.`

Compute challenge: $h_1 \leftarrow H_1(\mu, \sigma_1)$, $((\epsilon_{k,j})_{j \in [C]})_{k \in [\tau]} \leftarrow \mathsf{ExpandH1}(h_1)$.

`// Phase 3: Committing to the checking protocol.`

for *each repetition* $k \in [\tau]$ **do**

- Simulate the triple checking protocol as in Section 2.4 for all parties with challenge $\epsilon_{k,j}$. The inputs are $(x_{k,j}^{(i)}, y_{k,j}^{(i)}, z_{k,j}^{(i)}, a_{k,j}^{(i)}, b_{k,j}^{(i)}, c_k^{(i)})$, and let $\alpha_{k,j}^{(i)}$ and $v_k^{(i)}$ be the broadcast values.

Set $\sigma_2 \leftarrow (\mathsf{salt}, (((\alpha_{k,j}^{(i)})_{j \in [C]}, v_k^{(i)})_{i \in [N]})_{k \in [\tau]})$.

`// Phase 4: Challenging the views of the MPC protocol.`

Compute challenge hash: $h_2 \leftarrow H_2(h_1, \sigma_2)$, $(\bar{i}_k)_{k \in [\tau]} \leftarrow \mathsf{ExpandH2}(h_2)$.

`// Phase 5: Opening the views.`

for *each repetition* $k \in [\tau]$ **do**

- $\mathsf{pdecom}_k \leftarrow \mathsf{IC\text{-}VSC.Open}((\mathsf{salt}, k), \mathsf{decom}_k, [N] \setminus \{\bar{i}_k\})$.

Output $\sigma \leftarrow (\mathsf{salt}, h_1, h_2, (\mathsf{pdecom}_k, \Delta c_k, (\Delta z_{k,j}, \alpha_{k,j}^{(\bar{i}_k)})_{j \in [\ell]})_{k \in [\tau]})$.

Algorithm 6. $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$ - reduced BN++, verification algorithm.

Parse σ as $(\mathsf{salt}, h_1, h_2, (\mathsf{pdecom}_k, \Delta c_k, (\Delta z_{k,j}, \alpha_{k,j}^{(\bar{i}_k)})_{j\in[\ell]})_{k\in[\tau]})$.
Compute the hash value of the message: $\mu \leftarrow H_0(\mathsf{pk}, m)$
Expand hashes:
  $((\epsilon_{k,j})_{j\in[\ell+1]})_{k\in[\tau]} \leftarrow \mathsf{ExpandH1}(h_1)$ and $(\bar{i}_k)_{k\in[\tau]} \leftarrow \mathsf{ExpandH2}(h_2)$.
for *each repetition* $k \in [\tau]$ **do**
  $((\mathsf{seed}_k^{(i)} \| \mathsf{tape}_k^{(i)})_{i\in[N]\setminus\{\bar{i}_k\}}, \mathsf{com}_k) \leftarrow$
  $\mathsf{IC\text{-}VSC.Recon}(\mathsf{salt}\|\langle\tau\rangle_B, \mathsf{pdecom}_k, [N] \setminus \{\bar{i}_k\})$.
  for *each party* $i \in [N] \setminus \{\bar{i}_k\}$ **do**
    for *each gate* g *with index* j **do**
      if g *is an addition with inputs* $(x_{k,j}, y_{k,j})$ **then**
        Compute the output share of $z_{k,j}^{(i)} = x_{k,j}^{(i)} + y_{k,j}^{(i)}$.
      if g *is a multiplication with inputs* $(x_{k,j}^{(i)}, y_{k,j}^{(i)})$ **then**
        Sample an output share: $z_{k,j}^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)})$.
        if $i = N$ **then**
          Adjust the last share $z_{e,j}^{(i)} \leftarrow z_{e,j}^{(i)} + \Delta z_{e,j}$.
        Sample $a_{k,j}^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)})$, and set $b_{k,j}^{(i)} = y_{k,j}^{(i)}$.
    Sample $c_k^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)})$
    if $i = N$ **then**
      Adjust the last share $c_k^{(i)} \leftarrow c_k^{(i)} + \Delta c_k$.
    Let $\mathsf{pk}_k^{(i)}$ be the final output shares.
  Compute $\mathsf{pk}_k^{(\bar{i}_k)} = \mathsf{pk} - \sum_{i\neq\bar{i}_k} \mathsf{pk}_k^{(i)}$.
Set $\sigma_1 \leftarrow (\mathsf{salt}, (\mathsf{com}_k, (\mathsf{pk}_k^{(i)})_{i\in[N]}, \Delta c_k, (\Delta z_{k,j})_{j\in[C]})_{k\in[\tau]})$. Set $h_1' \leftarrow H_1(\mu, \sigma_1)$.
for *each parallel execution* $k \in [\tau]$ **do**
  for *each party* $i \in [N] \setminus \{\bar{i}_k\}$ **do**
    Simulate the triple checking protocol as in Section 2.4 for all parties with challenge $\epsilon_{k,j}$. The inputs are $(x_{k,j}^{(i)}, y_k^{(i)}, z_{k,j}^{(i)}, a_{k,j}^{(i)}, b_{k,j}^{(i)}, c_k^{(i)})$, and let $\alpha_{k,j}^{(i)}$ and $v_k^{(i)}$ be the broadcast values.
  Compute $v_k^{(\bar{i}_k)} = 0 - \sum_{i\neq\bar{i}_k} v_k^{(i)}$.
Set $\sigma_2 \leftarrow (\mathsf{salt}, (((\alpha_{k,j}^{(i)})_{j\in[C]}, v_k^{(i)})_{i\in[N]})_{k\in[\tau]})$, and $h_2' \leftarrow H_2(h_1', \sigma_2)$.
Output Accept if $h_1 = h_1'$ and $h_2 = h_2'$, or output Reject otherwise.

B Security Proofs for IC-VSC

In this section, we provide a security proof of the extractable semi-binding and multi-instance hiding property for IC-VSC. The outline of the proof is similar to the proof in Section 3.1.

B.1 Extratable Semi-Binding Property

Lemma 8. *Let $E : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ be an ideal cipher. Let $\mathcal{A}$ be an arbitrary adversary that makes Q queries to E. Then, the probability that $\mathcal{A}$ finds $\mathsf{salt} = (\mathsf{salt}_1, \mathsf{salt}_2, \mathsf{b}) \in \{0,1\}^{\lambda+\lambda+8}, i \in \{0, 2, \ldots, 254\}$, and distinct $n, n' \in \{0,1\}^\lambda$ such that*

$$\begin{cases} E_{2\cdot n \oplus E_{\mathsf{salt}_2}(n\oplus\mathsf{salt}_1)}((\mathsf{0b0}^{\lambda-24}\|\mathsf{b}\|\langle i\rangle_B\|\langle 0\rangle_B)\oplus\mathsf{salt}_1) \\ = E_{2\cdot n' \oplus E_{\mathsf{salt}_2}(n'\oplus\mathsf{salt}_1)}((\mathsf{0b0}^{\lambda-24}\|\mathsf{b}\|\langle i\rangle_B\|\langle 0\rangle_B)\oplus\mathsf{salt}_1), \\ E_{2\cdot n \oplus E_{\mathsf{salt}_2}(n\oplus\mathsf{salt}_1)\oplus n}((\mathsf{0b0}^{\lambda-24}\|\mathsf{b}\|\langle i+1\rangle_B\|\langle 0\rangle_B)\oplus\mathsf{salt}_1) \\ = E_{2\cdot n' \oplus E_{\mathsf{salt}_2}(n'\oplus\mathsf{salt}_1)\oplus n'}((\mathsf{0b0}^{\lambda-24}\|\mathsf{b}\|\langle i+1\rangle_B\|\langle 0\rangle_B)\oplus\mathsf{salt}_1), \end{cases} \tag{7}$$

is at most $10Q/2^\lambda$.

Proof. At the end of game, we define $E_k(x) = \perp$ for non-queried input (k, x) to E, and we write $\perp \neq \perp$ for simplicity. We also denote $\mathsf{ctr}_l = \mathsf{0b0}^{\lambda-24}\|\mathsf{b}\|\langle i\rangle_B\|\langle 0\rangle_B$ and $\mathsf{ctr}_r = \mathsf{0b0}^{\lambda-24}\|\mathsf{b}\|\langle i+1\rangle_B\|\langle 0\rangle_B$. We define

$$\begin{aligned} \mathcal{L}_1 &= \{(n, l, l') \in \{0,1\}^{3\lambda} : 2\cdot n \oplus E_{\mathsf{salt}_2}(n\oplus\mathsf{salt}_1) = l, \\ &\qquad\qquad E_l(\mathsf{ctr}_l \oplus \mathsf{salt}_1) = E'_l(\mathsf{ctr}_l\oplus\mathsf{salt}_1), l \neq l'\}, \\ \mathcal{L}_2 &= \{(n, r, r') \in \{0,1\}^{3\lambda} : 2\cdot n \oplus E_{\mathsf{salt}_2}(n\oplus\mathsf{salt}_1)\oplus n = r, \\ &\qquad\qquad E_r(\mathsf{ctr}_r \oplus \mathsf{salt}_1) = E'_r(\mathsf{ctr}_r\oplus\mathsf{salt}_1), r \neq r'\}, \\ \mathcal{L}_3 &= \big\{(n, l, l', r, r') \in \{0,1\}^{5\lambda} : (n, l, l') \in \mathcal{L}_1, (n, r, r') \in \mathcal{L}_2\big\} \end{aligned}$$

and auxiliary events Aux_j for $j \in [3]$, where

$$\mathsf{Aux}_j \Leftrightarrow |\mathcal{L}_i| > Q$$

and let $\mathsf{Aux} = \mathsf{Aux}_1 \vee \mathsf{Aux}_2 \vee \mathsf{Aux}_3$. Then, by Markov's inequality, we have

$$\begin{aligned} \Pr[\mathsf{Aux}_1] &\leq \frac{\mathsf{Ex}[|\mathcal{L}_1|]}{Q} \leq \frac{1}{Q}\left(\frac{Q^2}{2^\lambda} + \frac{Q^3}{2^{2\lambda}}\right), \\ \Pr[\mathsf{Aux}_2] &\leq \frac{\mathsf{Ex}[|\mathcal{L}_2|]}{Q} \leq \frac{1}{Q}\left(\frac{Q^2}{2^\lambda} + \frac{Q^3}{2^{2\lambda}}\right), \\ \Pr[\mathsf{Aux}_3] &\leq \frac{\mathsf{Ex}[|\mathcal{L}_3|]}{Q} \leq \frac{1}{Q}\left(\frac{Q^3}{2^{2\lambda}} + \frac{2Q^4}{2^{3\lambda}} + \frac{Q^5}{2^{4\lambda}}\right). \end{aligned}$$

For each query $E_{\mathsf{salt}_2}(n'')$, let $n = n''\oplus\mathsf{salt}_1$. We say Bad occurs if there exists $n' \neq n$ satisfies (7). Let $2\cdot n\oplus E_{\mathsf{salt}_2}(n\oplus\mathsf{salt}_1) = l$, $n \oplus l = r$, $2\cdot n'\oplus E_{\mathsf{salt}_2}(n'\oplus\mathsf{salt}_1) = l'$, and $n' \oplus l' = r'$. We classify the Bad into sub-events, up to the freshness of $E_l(\mathsf{ctr}_l \oplus \mathsf{salt}_1)$ and $E_r(\mathsf{ctr}_r \oplus \mathsf{salt}_1)$.

– $\mathsf{L}_1 \Leftrightarrow l = l'$ or $E_l(\mathsf{ctr}_l \oplus \mathsf{salt}_1)$ is fresh. Observe that n should satisfies

$$(2 \cdot n \oplus E_{\mathsf{salt}_2}(n \oplus \mathsf{salt}_1) = l') \vee$$
$$(2 \cdot n \oplus E_{\mathsf{salt}_2}(n \oplus \mathsf{salt}_1) \neq l' \wedge E_l(\mathsf{ctr}_l \oplus \mathsf{salt}_1) = E_{l'}(\mathsf{ctr}_l \oplus \mathsf{salt}_1)).$$

– $\mathsf{L}_2 \Leftrightarrow E_l(\mathsf{ctr}_l \oplus \mathsf{salt}_1)$ is not fresh and $l \neq l'$. Observe that n should satisfies

$$(n', l', 2 \cdot n \oplus E_{\mathsf{salt}_2}(n \oplus \mathsf{salt}_1)) \in \mathcal{L}_1.$$

– $\mathsf{R}_1 \Leftrightarrow r = r'$ or $E_r(\mathsf{ctr}_r \oplus \mathsf{salt}_1)$ is fresh. Observe that n should satisfies

$$(2 \cdot n \oplus E_{\mathsf{salt}_2}(n \oplus \mathsf{salt}_1) \oplus n = r') \vee$$
$$(2 \cdot n \oplus E_{\mathsf{salt}_2}(n \oplus \mathsf{salt}_1) \oplus n \neq r' \wedge E_r(\mathsf{ctr}_r \oplus \mathsf{salt}_1) = E_{r'}(\mathsf{ctr}_r \oplus \mathsf{salt}_1)).$$

– $\mathsf{R}_2 \Leftrightarrow E_r(\mathsf{ctr}_r \oplus \mathsf{salt}_1)$ is not fresh and $r \neq r'$. Observe that n should satisfies

$$(n', r', (2 \cdot n \oplus E_{\mathsf{salt}_2}(n \oplus \mathsf{salt}_1) \oplus n) \in \mathcal{L}_2.$$

Note that one cannot have $l = l'$ and $r = r'$ at the same time, since it implies $n = n'$. Then, we have

$$\Pr[\mathsf{L}_1 \wedge \mathsf{R}_1] \leq \frac{6Q^2}{2^{2\lambda}}, \qquad \Pr[\mathsf{L}_2 \wedge \mathsf{R}_1 \wedge \neg\mathsf{Aux}_1] \leq \frac{2Q^2}{2^{2\lambda}},$$
$$\Pr[\mathsf{L}_1 \wedge \mathsf{R}_2 \wedge \neg\mathsf{Aux}_2] \leq \frac{2Q^2}{2^{2\lambda}}, \qquad \Pr[\mathsf{L}_2 \wedge \mathsf{R}_2 \wedge \neg\mathsf{Aux}_3] \leq \frac{Q^2}{2^{2\lambda}},$$

and

$$\begin{aligned}\Pr[\mathsf{Bad}] &\leq \Pr[\mathsf{Aux}] + \sum_{i,j\in[2]} \Pr[\mathsf{L}_i \wedge \mathsf{R}_j \wedge \neg\mathsf{Aux}] \\ &\leq \frac{2Q}{2^\lambda} + \frac{14Q^2}{2^{2\lambda}} + \frac{2Q^3}{2^{3\lambda}} + \frac{Q^4}{2^{4\lambda}} \\ &\leq \frac{10Q}{2^\lambda}\end{aligned}$$

provided that $Q \leq 2^{\lambda-1}$, which concludes the proof. □

Lemma 9. *Let $E : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ be an ideal cipher. Let $\mathcal{A}$ be an arbitrary adversary that makes Q queries to E. Then, $\mathcal{A}$'s u-extractable semi-binding advantage $\mathbf{Adv}^{u\text{-}\mathsf{ESB}}_{\mathsf{IC\text{-}VSC}}(\mathcal{A})$ against* IC-VSC *is bounded by*

$$\mathbf{Adv}^{u\text{-}\mathsf{ESB}}_{\mathsf{IC\text{-}VSC}}(\mathcal{A}) \leq \frac{14Q}{2^\lambda},$$

for $u = N \cdot (2\lambda/\log\lambda)$.

Proof. Intuitively, according to Lemma 8, the probability of finding a collision in commitments derived by a non-leaf node is negligible. Furthermore, for each leaf node, the number of multi-collisions is bounded by Chernoff bound. We now proceed to formally bound the adversary's advantage.

Let Q be the number of queries to E and $\mathcal{Q}$ be the collection of all queries to E. Without loss of generality, we remove the salt input in Commit and consider as all salts are equaled to 0^λ.

We first define the extractor $\mathsf{Ext}(\mathcal{Q}, \mathsf{com} = (\mathsf{com}_1, \ldots, \mathsf{com}_N))$ as follows.

1. For each $i \in [N]$, find $S_{0,i}$ such that
$$S_{0,i} = \{s : E_s(\mathsf{ctr}[0, i, 0]) = \mathsf{com}_i\}$$
2. For each $e \in [d-1]$ and $i \in [N/2^e]$, find $S_{e,i}$ such that
$$S_{e,i} = \{n : E_0(s) \oplus 2s \in S_{e-1,2i-1}, E_0(s) \oplus 3s \in S_{e-1,2i}\}$$
3. For $i \in [N]$, let
$$A_i = \Big\{(p_1, \ldots, p_d) : p_e \in S_{e-1,i_{e-1}} \text{ for } e \in [d]\Big\}$$
where $i_e = (\lfloor(\bar{i} - 1)/2^e\rfloor \oplus 1) + 1$ for $e \in [d-1]$
4. Let S be the set of messages, where
$$S = \Big\{(s_1, \ldots, s_N) : s_i = \perp \text{ if } S_{0,i} = \emptyset \text{ and } s_i \in S_{0,i} \text{ otherwise,}$$
$$\mathsf{Recon}(p_1, \ldots, p_d, \mathsf{com}_i, I = [N] \setminus \{i\}) = (s_i)_{i \in I} \text{ for } (p_1, \ldots, p_d) \in A_i\Big\}$$
5. Finally, Ext outputs arbitrary u or less elements in S.

We define some bad events.

- $\mathsf{Bad}_1 \Leftrightarrow$ there exists $e \in [d-1]$ and $i \in [N/2^e]$ such that $|S_{e,i}| \geq 2$.
- $\mathsf{Bad}_2 \Leftrightarrow$ there exists $i \in [N]$ such that $|S_{0,i}| \geq 2\lambda/\log\lambda$.

Observe that $\Pr[|S_{0.i}| \geq c] \leq \Pr[X \geq c]$ where X follows $\mathcal{B}(Q, 2/2^\lambda)$. Similar to (1), one have
$$\Pr[\mathsf{Bad}_2] \leq \frac{4Q}{2^\lambda}$$
By Lemma 8 and (1),
$$\Pr[\mathsf{Bad}_1 \vee \mathsf{Bad}_2] \leq \frac{14Q}{2^\lambda} \tag{8}$$
In the following, we analyze the extracting condition without bad events.

- As S contains all possible (pdecom_I, I), $\mathcal{A}$ wins the game only if $|S| \geq u$.
- By $\neg\mathsf{Bad}_1$, we have $|S| \leq \sum_{i\in[N/2]} |S_{0,2i}| \cdot |S_{0,2i-1}|$. Then, by $\neg\mathsf{Bad}_2$, we have
$$|S| \leq 2N\left(\frac{\lambda}{\log\lambda}\right)^2.$$

Therefore, $\mathcal{A}$ cannot win the game without bad events so we have
$$\mathbf{Adv}^{u\text{-}\mathsf{ESB}}_{\mathsf{IC\text{-}VSC}}(\mathcal{A}) \leq \Pr[\mathsf{Bad}_1 \vee \mathsf{Bad}_2] \leq \frac{11Q}{2^\lambda}$$
provided that $u = 2N\left(\frac{\lambda}{\log\lambda}\right)^2$. $\square$

B.2 Multi-Instance Hiding Property

Lemma 10. *Let $E : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ be an ideal cipher. Let $\mathcal{A}$ be an arbitrary adversary that makes Q queries to E. Then, $\mathcal{A}$'s multi-instance hiding advantage $\mathbf{Adv}^{Q_I\text{-}\mathsf{MIH}}_{\mathsf{IC\text{-}VSC}}(\mathcal{A})$ against* $\mathsf{IC\text{-}VSC}$ *is bounded by*

$$\mathbf{Adv}^{\mathsf{MIH}}_{\mathsf{IC\text{-}VSC}}(\mathcal{A}) \le \frac{Q_I^2}{2^{2\lambda}} + \frac{6\lambda \cdot Q}{2^\lambda \cdot \log \lambda}.$$

Proof. Let Q_I be the number of instances. We will bound the advantage using the H-coefficient technique. Denote $\mathcal{I}$ the ideal world where the hidden nodes are always replaced to random strings, and denote $\mathcal{R}$ the real world where the hidden nodes are always unchanged. Let $\mathcal{T}$ be the transcript of $\mathcal{A}$ which contains queries to the random oracles and the instances given in the game. The parent node of the hidden seed $\overline{\mathsf{node}}$ is derived by $\overline{\mathsf{node}} = m_{\bar{i}} \oplus m_{((\bar{i}-1)\oplus 1)+1}$. Now we define some events of bad transcript as follows.

- Bad_1: The maximal multi-collision in salt_1 is greater than $2\lambda/\log\lambda$. From Equation 1, $\Pr[\mathsf{Bad}_1] \le 2Q_I/2^\lambda$.
- Bad_2: The maximal multi-collision in salt_2 is greater than $2\lambda/\log\lambda$. From Equation 1, $\Pr[\mathsf{Bad}_2] \le 2Q_I/2^\lambda$.
- Bad_3: A query $(m_{\bar{i}}, (\mathsf{0b0}^{\lambda-24}\|\mathsf{b}\|\langle i\rangle_B\|\langle 0\rangle_B)\oplus\mathsf{salt}_1)$ is queried to E. $\Pr[\mathsf{Bad}_3 \wedge \neg\mathsf{Bad}_1] \le (2\lambda\cdot Q)/(2^\lambda\cdot\log\lambda)$.
- Bad_4: A query $(\mathsf{salt}_2, m_{\bar{i}} \oplus \mathsf{salt}_1)$ is queried to E. $\Pr[\mathsf{Bad}_4 \wedge \neg\mathsf{Bad}_2] \le (2\lambda \cdot Q)/(2^\lambda\cdot\log\lambda)$.
- $\mathsf{Bad}_{5.1}$: A $m_{\bar{i}}$ is a left child and a query $(\mathsf{salt}_2, 2\cdot m_{\bar{i}} \oplus \mathsf{salt}_1)$ is queried to E^{-1}.
- $\mathsf{Bad}_{5.2}$: A $m_{\bar{i}}$ is a right child and a query $(\mathsf{salt}_2, 2\cdot m_{\bar{i}} \oplus m_{\bar{i}} \oplus \mathsf{salt}_1)$ is queried to E^{-1}. $\Pr[(\mathsf{Bad}_{5.1} \vee \mathsf{Bad}_{5.2}) \wedge \neg\mathsf{Bad}_2] \le (2\lambda\cdot Q)/(2^\lambda\cdot\log\lambda)$.

We say $\mathcal{T}_{\mathsf{Bad}}$ be the set of bad transcripts, while $\mathcal{T}_{\mathsf{Good}}$ be the complement of $\mathcal{T}_{\mathsf{Bad}}$, and let T_{id} (resp. T_{re}) be the distribution of γ in $\mathcal{I}$ (resp. $\mathcal{R}$). As Q ideal cipher queries and Q_I instances of $m_{\bar{i}}$ are included in transcripts, for $\gamma \in \mathcal{T}_{\mathsf{Good}}$,

$$\Pr[\gamma = T_{\mathsf{id}}] = \frac{1}{(2^\lambda)^{Q_I}} \cdot \prod_{s\in\{0,1\}^\lambda} \frac{1}{(2^\lambda)_{P_s}},$$

where P_s denotes the number of ideal cipher queries with key input s. Additionally, depending on oracle queries, some values may be excluded as candidates for $m_{\bar{i}}$,

$$\Pr[\gamma = T_{\mathsf{re}}] \ge \frac{1}{(2^\lambda)^{Q_I}} \cdot \prod_{s\in\{0,1\}^\lambda} \frac{1}{(2^\lambda)_{P_s}},$$

where P_s denotes the number of ideal cipher queries with key input s. Therefore, by Lemma 2, the advantage is bounded by

$$\mathbf{Adv}^{4Q_I\text{-}\mathsf{MIH}}_{\mathsf{IC\text{-}VSC}}(\mathcal{A}) \le \frac{Q_I^2}{2^{2\lambda}} + \frac{6\lambda \cdot Q}{2^\lambda \cdot \log \lambda}.$$

□