Relaxed Vector Commitment for Shorter Signatures

Seongkwang Kim¹, Byeonghak Lee¹, and Mincheol Son²

Abstract. MPC-in-the-Head (MPCitH) has recently gained traction as a foundation for post-quantum signature schemes, offering robust security without 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 relaxing vector commitments within MPCitH-based schemes. We introduce the concept of *vector semi-commitment*, which relaxes the binding property of traditional vector commitment. Vector semi-commitment schemes may allow an adversary to find more than one preimage of a commitment. We instantiate vector semi-commitment schemes in both the random oracle model and the ideal cipher model, leveraging recent optimizations on GGM tree such as correlated GGM tree.

We apply the ideal-cipher-based vector semi-commitment scheme to the BN++ signature scheme and prove it fully secure in the ideal cipher model. Implementing these improvements in the AlMer 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

1 Introduction

The MPC-in-the-Head (MPCitH) paradigm [20] has recently 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 depends solely on the one-way function used in key generation. This advantage allows primitives without trapdoors, such as block ciphers [29,11,24], 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 result 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 MPCitH-based signature schemes through protocol optimization [23,6,28,1] and improved cryptographic primitives [11,24].

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 correlated GGM technique [10,9,19].

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. We call the latter characteristic *extractable semi-binding* — an extractor (only defined in the security proof) cannot find multi-collisions of a commitment more than a specific amount — and it is a key factor for the security proof.

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 except a constant number of hash function calls, incorporate recent optimizations of the GGM tree construction [9,8,19]. Previous works on optimizing the GGM tree can be divided into three contributions: the use of fixed-key block cipher, reducing primitive calls,

and correlated GGM tree. While none of previous works include all the desirable characteristics, our vector semi-commitment scheme include all the above characteristics; ours calls (N-1) fixed-key block cipher for evaluating the GGM tree, and the correction to the secret key share (usually denoted by Δ sk) is not required since correlated GGM preserves the difference. By slightly modifying the Davies-Mayer construction provided in [8], our vector semi-commitment scheme enjoys better performance of both the fixed-key block cipher and the correlated GGM tree.

To see the practical implication of our vector semi-commitment, we apply all the improvements to BN++ [22], an MPCitH-based signature scheme. We prove its full security³ in the ideal cipher model. In other words, we prove that replacing a vector commitment with a vector semi-commitment does not compromise the security of the resulted signature scheme. Performance-related parameters, such as the number of repetitions, remain unchanged, thereby reducing the signature size while maintaining performance.

We also implement our improvements in AlMer v2.0 [25], achieving up to 18% shorter signatures and up to 112% faster signing and verification speeds compared to AlMer v2.0. Compared to other MPCitH-based signature schemes such as SDitH [27] 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 [23]. Since then, the GGM tree has been used as a core technique to reduce the signature size in the MPCitH-based signature schemes [11,24,14,27,7].

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 technique reduces the number of random permutation calls by half. The correlated GGM tree was recently applied VOLE-in-the-Head-based signature schemes. Cui et al. reduces the number of random permutation calls for generating seeds is halved [10], and Huth and Joux reduces the signature size by deleting the correction to the secret key share. However, both works did not accomplish the both contributions simultaneously. 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

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

efficient primitives such as fixed-key AES. However, the provable security of their proposal did not exceed the birthday bound. Independent with the correlated GGM tree, 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. Compared to these works, our vector semi-commitment enjoys all the good characteristics: fast, salted, half number of calls, reduced signature size, and full security.

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 $[n] = \{1, \dots, n\}$. Unless stated otherwise, all logarithms are to the base 2. For an integer $a \in \{0, 1, \dots, 255\}$, $\langle a \rangle_B$ 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 $(n)_k = n \cdot (n-1) \cdot \dots \cdot (n-k+1)$.

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 $\sum_{i} x^{(i)} = x$.

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 $S = \{x_1, x_2, \dots, x_Q\}$ with $x_i \leftarrow_{\$} \{0, 1\}^{\lambda}$, we can bound the probability that the maximum number of 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

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

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.

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

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

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

We 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 [22], 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 $\epsilon_1, \dots, \epsilon_C \in \mathbb{F}$.

- For $i \in [N]$, the *i*-th party locally sets $\alpha_1^{(i)}, \dots, \alpha_C^{(i)}$ where $\alpha_i^{(i)} = \epsilon_j \cdot x_i^{(i)} + a_i^{(i)}$.
- The parties open $\alpha_1, \dots, \alpha_C$ 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 ([22]) 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.

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 or a real world and tries to distinguish two worlds. Let T_{id} and T_{id} be the distribution of transcript in the ideal world and the real world,

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

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

Then, the distinguishing advantage of A in distinguishing I from 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 VC scheme typically consists of four 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.

- Commit^H(salt, root) \rightarrow (com, decom, (m_1, \ldots, m_N)): given an IV salt and a λ -bit string 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 salt, opening information decom and a subset $I \subseteq [N]$ of indices, output a partial opening information pdecom for I.
- Recon^H(salt, pdecom, I) \rightarrow ($(m_i)_{i \in I}$, com): given an IV salt, a partial opening information pdecom for a subset I, output partially reconstructed messages and the full commitment com.

- Verify^H(salt, com, pdecom, I) \rightarrow { $(m_i)_{i \in I}$ } \cup { \bot }: given an IV salt, a commitment com, a partial opening information pdecom, and a subset I, either output the messages $(m_i)_{i \in I}$ (accept) or \bot (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. This opening is referred to as an *all-but-one*, with I being the all-but-one subset. If the context is clear, we will omit the random oracle indication $({}^{H})$.

The first part of messages in 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 Δ sk) is always zero. The correlated GGM tree [17] enables this in vector semi-commitments, and the following property is inspired by this technique. This technique was currently used in [19].

Definition 2 (Correlated Vector Semi-commitment). A vector commitment scheme (Commit, Open, Verify) is called correlated vector commitment if

$$Commit(salt, root) = (com, decom, (m_1, ..., m_N))$$

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

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 $\operatorname{Ext}(\operatorname{salt}, \mathcal{Q}, \operatorname{com}) \to ((m_i^{(j)})_{i \in [N]})_{j \in J}$ 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 $N = \operatorname{poly}(\lambda)$ and Q queries to the random oracle and stateful \mathcal{A} , is defined as follows.

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

We define A's u-extractable semi-binding advantage by

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

Although the extractor Ext extracts the set of 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 $m_i = \bot$ 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 A is defined as follows.

```
1. \operatorname{root} \leftarrow_{\$} \{0,1\}^{\lambda}, \ b^{*} \leftarrow_{\$} \{0,1\}
2. For \ j \in [Q_{I}], \ do \ the \ following:
(a) \operatorname{salt}_{j} \leftarrow_{\$} \{0,1\}^{2\lambda}
(b) (\operatorname{com}_{j}, \operatorname{decom}_{j}, (m_{j,1}^{*}, \ldots, m_{j,N}^{*})) \leftarrow \operatorname{Commit}(\operatorname{salt}_{j}, \operatorname{root})
(c) \overline{i}_{j} \leftarrow_{\$} [N], \ I_{j} = [N] \setminus \{\overline{i}_{j}\}
(d) \operatorname{pdecom}_{I_{j}} \leftarrow \operatorname{Open}(\operatorname{salt}_{j}, \operatorname{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}((\operatorname{salt}_{j}, \operatorname{pdecom}_{I_{j}}, I_{j})_{j \in [Q_{I}]})
4. For \ j \in [Q_{I}], \ set \ m_{j,\overline{i}_{j}} \leftarrow \begin{cases} m_{j,\overline{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 A's multi-instance hiding advantage by

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

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 RO-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, RO-VSC is constructed as Figure 1.

The commitment process Commit involves an evaluation of the correlated GGM tree to generate seeds and tapes, culminating in the output of a commitment com and opening information decom. Compared to the traditional vector

- **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: salt $\in \{0,1\}^{2\lambda}$, and root $\in \{0,1\}^{\lambda}$.
- $Commit^{H}(salt, root)$:
 - 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

$$\begin{split} \mathsf{seed}_i &\leftarrow \mathsf{node}_{d,i} \\ \mathsf{com}_i &\leftarrow H_{\mathsf{com}}(\mathsf{salt}, i, \mathsf{seed}_i) \\ \mathsf{tape}_i &\leftarrow H_{\mathsf{exp}}(\mathsf{salt}, i, \mathsf{seed}_i) \\ m_i &= \mathsf{seed}_i \parallel \mathsf{tape}_i \end{split}$$

- 4. Output a commitment $\mathsf{com} = (\mathsf{com}_1, \dots, \mathsf{com}_N)$ with opening information $\mathsf{decom} = ((\mathsf{node}_{e,i})_{e \in [d-1], i \in [2^e]}, \mathsf{com})$, and messages (m_1, \dots, 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 $pdecom = (path_I, com_{\bar{i}})$.
- Recon^H(salt, 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 $((m_i)_{i \in I}, \text{com})$.
- $\ \mathsf{Verify}^H(\mathsf{salt},\mathsf{com} = (\mathsf{com}_i^*)_{i \in [N]},\mathsf{pdecom},I = [N] \setminus \{\bar{i}\}):$
 - 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 \bot otherwise.

Fig. 1: RO-VSC

commitment based on the GGM tree, our RO-VSC employs the correlated GGM tree, and the commitment size is reduced from 2λ to λ . The double-length PRG part (Step 2 in Commit^H) of RO-VSC is depicted in fig. 2a. Other algorithms are similar to vector commitment schemes. The opening algorithm Open extracts path information for a given index set, while the reconstruction algorithm Recon rebuilds messages and commitments from path data and verifies the integrity of the commitment. Finally, the verification algorithm Verify ensures that the 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.

As this work focuses on security proof in the ideal cipher model, we only state Lemma 3 for extractable semi-binding property and Lemma 4 for multi-instance hiding property of RO-VSC. The proofs of the lemmas are provided in Supplementary Material B.1.

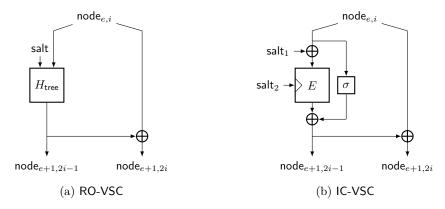


Fig. 2: Double-length PRG of RO-VSC and IC-VSC.

Now we can prove the extractable semi-binding property of RO-VSC using this lemma.

Lemma 3 Let $H_{\mathsf{com}} : \{0,1\}^* \to \{0,1\}^{\lambda}$ and $H_{\mathsf{tree}} : \{0,1\}^* \to \{0,1\}^{\lambda}$ be random oracles. Let \mathcal{A} be an arbitrary adversary that makes Q queries to the random oracles. Then \mathcal{A} 's u-extractable semi-binding advantage $\mathbf{Adv}^{u\text{-esb}}_{\mathsf{RO-VSC}}(\mathcal{A})$ against RO-VSC is bounded by

$$\mathbf{Adv}^{u\text{-esb}}_{\mathsf{RO-VSC}}(\mathcal{A}) \leq \frac{10Q}{2^{\lambda}},$$

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

Lemma 4 Let $H_{\mathsf{com}}: \{0,1\}^* \to \{0,1\}^{\lambda}$ and $H_{\mathsf{tree}}: \{0,1\}^* \to \{0,1\}^{\lambda}$ be random oracles. Let \mathcal{A} be an arbitrary adversary that makes Q queries to the random oracles. Then \mathcal{A} 's multi-instance hiding advantage $\mathbf{Adv}_{\mathsf{RO-VSC}}^{Q_I\text{-mih}}(\mathcal{A})$ against RO-VSC

is bounded by

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

3.2 Instantiation from Ideal Cipher

Now we replace the random oracles with an ideal cipher E. We instantiate VSC from the ideal cipher, named IC-VSC, and also prove its properties. Let $E: \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ be a ideal cipher, and $\sigma: \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ be an orthomorphism. Then IC-VSC is constructed as Figure 3.

The main difference between IC-VSC and RO-VSC is that all the random oracles are replaced with the ideal cipher, which requires sophisticated domain separation. The GGM tree evaluation (Step 2 in Commit^E), the share generation (Step 3 in Commit^E) are realized using only ideal cipher calls. We note that the whole τ GGM trees in a signature can be evaluated by a fixed-key block cipher. The double-length PRG part of the GGM tree evaluation is depicted in Figure 2b. One noteworthy point is that the double-length PRG of IC-VSC is inspired by Davies-Meyer-based GGM tree proposed by Bui et al. [8], and it is modified to an orthomorphism-applied version to work properly in correlated GGM tree. The new variable b represents the current repetition in the signature scheme.

In the following, we prove the extractable semi-binding property and the multi-instance hiding property of IC-VSC with a supporting lemma. Although the proofs are straightforward, beware that the definition of Ext in Lemma 6 will be used in the EUF-KO security proof. Lemma 5 is the supporting lemma which limits the probability of legitimate height-1 nodes (height-0 nodes mean leaf nodes, not commitments).

Lemma 5 Let $E: \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ be an ideal cipher and $\sigma: \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ be an orthomorphism. Let \mathcal{A} be an arbitrary adversary that makes Q queries to E. Then, the probability that \mathcal{A} finds 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

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

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

Proof. Because of the page limit, see Supplementary material B.2 for the proof.

Lemma 6 Let $E: \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ be an ideal cipher and $\sigma: \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ be an orthomorphism. Let A be an arbitrary adversary that

- **Parameters**: an ideal cipher E, an orthomorphism σ , integer C, a power-of-two integer N, and $d = \log N$.
- Inputs: salt = (salt₁, salt₂, b) $\in \{0, 1\}^{\lambda + \lambda + 8}$, and root $\in \{0, 1\}^{\lambda}$.
- $\mathsf{Commit}^E(\mathsf{salt},\mathsf{root})$:
 - 1. Set $node_{1,1} \leftarrow root$.
 - 2. For each level $e \in [d-1]$ and $i \in [2^e]$, set

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

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

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

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

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

- (b) Set tape_i \leftarrow tape_{i,1} $\| \cdots \|$ tape_{i,2C+1} and $m_i \leftarrow \mathsf{seed}_i \|$ tape_i.
- 4. Output a commitment $\mathsf{com} = (\mathsf{com}_1, \dots, \mathsf{com}_N)$ with opening information $\mathsf{decom} = ((\mathsf{node}_{e,i})_{e \in [d-1], i \in [N]}, \mathsf{com})$, and messages (m_1, \dots, 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 $pdecom = (path_I, com_{\bar{i}})$
- $\mathsf{Recon}^E(\mathsf{salt},\mathsf{pdecom},I=[N]\setminus\{\bar{i}\})$:
 - 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 $((m_i)_{i \in I}, \text{com})$.
- Verify^E(salt, com = $(com_i^*)_{i \in [N]}$, 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 \bot otherwise.

Fig. 3: IC-VSC

makes Q queries to E. Then, A's u-extractable semi-binding advantage $\mathbf{Adv}^{u\text{-esb}}_{\mathsf{IC-VSC}}(A)$ against $\mathsf{IC-VSC}$ is bounded by

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

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

Proof. Intuitively, according to Lemma 5, 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 the Chernoff bound. We now proceed to formally bound the adversary's advantage.

Let Q be the number of queries to E, and Q be the collection of all queries to E. Since the choice of salt and b does not affect extractable semi-binding, we can assume all salts and b are equaled to zero strings.

We first define the extractor $\mathsf{Ext}(0^\lambda, \mathcal{Q}, \mathsf{com} = (\mathsf{com}_1, \dots, \mathsf{com}_N))$ as follows.

1. For each $i \in [N]$, find leaf node sets $S_{0,i}$ such that

$$S_{0,i} = \{s : E_s(\mathsf{ctr}[\langle 0 \rangle_B, i, 0]) = \mathsf{com}_i\}$$

2. For each $e \in [d-1]$ and $i \in [N/2^e]$, find internal node sets $S_{e,i}$ such that

$$S_{e,i} = \{n : E_0(s) \oplus \sigma(s) \in S_{e-1,2i-1}, E_0(s) \oplus \sigma(s) \oplus s \in S_{e-1,2i}\}$$

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

$$A_i = \{(p_1, \dots, p_d) : p_e \in S_{e-1, i_{e-1}} \text{ for } e \in [d]\}$$

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

$$S = \left\{ (s_1, \dots, s_N) : s_i = \bot \text{ if } S_{0,i} = \emptyset \text{ and } s_i \in S_{0,i} \text{ otherwise,} \right.$$

$$\mathsf{Recon}(p_1, \dots, p_d, \mathsf{com}_i, I = [N] \setminus \{i\}) = (s_i)_{i \in I} \text{ for } (p_1, \dots, p_d) \in A_i \right\}$$

5. Finally, Ext outputs arbitrary u or less elements in S.

We define some bad events.

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

Since the probability of querying a fixed commitment is $1/(2^{\lambda} - Q)$, we have $\Pr[|S_{0,i}| \geq c] \leq \Pr[X \geq c]$ where X follows $B(Q, 2/2^{\lambda})$. Similar to (1), one have

$$\Pr\left[\mathsf{Bad}_2\right] \leq \frac{4Q}{2^{\lambda}}$$

By Lemma 5 and (1),

$$\Pr\left[\mathsf{Bad}_1 \vee \mathsf{Bad}_2\right] \le \frac{14Q}{2^{\lambda}} \tag{3}$$

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

- As S contains all possible (pdecom_I, I), A wins the game only if |S| > 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

$$|S| \le 2N \left(\frac{\lambda}{\log \lambda}\right)^2.$$

Therefore, A cannot win the game without the bad events so we have

$$\mathbf{Adv}^{u\text{-esb}}_{\mathsf{IC-VSC}}(\mathcal{A}) \leq \Pr\left[\mathsf{Bad}_1 \vee \mathsf{Bad}_2\right] \leq \frac{14Q}{2^{\lambda}}$$

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

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 $\mathbf{Adv}^{Q_1-\mathrm{mih}}_{\mathsf{IC-VSC}}(\mathcal{A})$ against IC-VSC is bounded by

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

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

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 $m_{\tilde{i}}$ 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 $m_{\tilde{i}}$,

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

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

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

4 Application of VSC to BN++

4.1 Description of rBN++

In this section, we apply IC-VSC to BN++, dubbed rBN++ (reduced BN++). rBN++ is a signature scheme containing three algorithms KeyGen, Sign and Verify, where KeyGen is assumed to be a one-way function. The signing and verification algorithms of the rBN++ can be found in Algorithm 5 and Algorithm 6 in Supplementary material A. The major differences between the rBN++ 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 an 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: ExpandH1: $\{0,1\}^{2\lambda} \to \mathbb{F}^{C \cdot \tau}$ and 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 rBN++

In this section, we prove the security of rBN++. The main difference between the security proofs for BN++ and rBN++ is that, semi-binding does not guarantee the uniqueness of H_1 -query input (μ, σ_1) . In the original BN++, individual commitment is of length 2λ bits so that the probability of H_1 -query input collision (not H_1 collision) is negligible. However, an EUF-KO adversary may find an H_1 -query input collision by querying to ideal cipher E, as the individual commitment size is less than 2λ bits. This case is covered in bad events Bad_4 , and Bad_5 in the security proof. Except this difference, the security can proved similarly.

Theorem 1 (EUF-KO Security of rBN++). Let $(N, \tau, \lambda, \mathbb{F})$ be parameters of rBN++ where $|\mathbb{F}|=2^{\lambda}$ and $N=2^{d}$. Assume that $H_{1}, H_{2}:\{0,1\}^{*} \rightarrow \{0,1\}^{2\lambda}$, ExpandH1, and ExpandH2 are modeled as random oracles. Let \mathcal{A} be an arbitrary adversary against the EUF-KO security of rBN++ that makes a total of Q random oracle queries and P ideal cipher queries. Then there exists PPT adversaries

- 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

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

where $\nu = 2\lambda/\log \lambda$, $u = \nu^2 N/2$, and $P, Q \leq 2^{\lambda-1}$.

Proof. Suppose that all the queries to H_1 , H_2 , E are listed in \mathcal{Q}_1 , \mathcal{Q}_2 , and \mathcal{P} , respectively. Let $|\mathcal{Q}_1| = Q_1$, $|\mathcal{Q}_2| = Q_2$, and $|\mathcal{P}| = P$. Suppose that KeyGen can be verified by C number of \mathbb{F} -multiplications.

Algorithm 1. ExpandH1(h_1):

- 1 $h_1 \rightarrow \mathcal{H}_1$.
- 2 $((\epsilon_{k,j})_{j\in[C]})_{k\in[\tau]} \leftarrow_{\$} (\mathbb{F}^C)^{\tau}$.
- **3** Return $((\epsilon_{k,j})_{j\in[C]})_{k\in[\tau]}$.

We program the random oracles for \mathcal{A} as in Algorithm 1, 2, 3, and 4, respectively. For convenience, we denote $\Delta_k = (\Delta c_k, (\Delta z_{k,j})_{j \in [2C+1]}), \alpha_k =$

Algorithm 2. ExpandH2(h_2):

- 1 $h_2 \rightarrow \mathcal{H}_2$.
- $\mathbf{z} \ (\bar{i}_1, \bar{i}_2, \dots, \bar{i}_{\tau}) \leftarrow_{\$} ([N])^{\tau}.$
- **з** Return $(\bar{i}_1, \bar{i}_2, \ldots, \bar{i}_{\tau})$.

 $(\alpha_{k,j})_{j\in[2C+1]}$, and $\epsilon_k = (\epsilon_{k,j})_{j\in[2C+1]}$. We manage sets \mathcal{H}_1 and \mathcal{H}_2 while the adversary is querying to identify any collision of H_1 and H_2 . We also manage sets $\operatorname{Succ}_1[\operatorname{salt}, h_1]$ (resp. $\operatorname{Succ}_2[\operatorname{salt}, h_2]$) for $\operatorname{salt}, h_1 \in \{0, 1\}^{2\lambda}$ (resp. $\operatorname{salt}, h_2 \in \{0, 1\}^{2\lambda}$). These sets contain $k \in [\tau]$ if and only if the k-th repetition passes the first (resp. second) multiplication check for a specific salt and first hash (resp. second hash). If one of such sets become equal to $[\tau]$, the adversary can forge a signature. Ext is the extractor in the u-ESB game.

 $\mathsf{MultCheck}_1^{(i)}$ and $\mathsf{MultCheck}_2^{(i)}$ in Algorithm 4 are the first round and second round of the multiplication checking protocol for the *i*-th party, which is explicitly defined by

$$\begin{split} & \mathsf{MultCheck}_1^{(i)}((\epsilon_j)_j, m_i) = (\epsilon_j \cdot x_j^{(i)} + a_j^{(i)})_j \\ & \mathsf{MultCheck}_2^{(i)}((\epsilon_j)_j, m_i, (\alpha_j^{(i)})_j) = \sum_{j=1}^C \epsilon_j \cdot z_j^{(i)} - \sum_{j=1}^C \alpha_j \cdot y_j^{(i)} + c^{(i)} \end{split}$$

where \mathbb{F} -elements $x_j^{(i)}, y_j^{(i)}, z_j^{(i)}, a_j^{(i)}$, and $c^{(i)}$ are components of m_i in specific positions, and \mathbb{F} -vectors $(*_j)_j$ are of length C. We note that the length of m_i is 2C+2 as some outputs are fed into some inputs. MultCheck in Algorithm 3 is defined by combining above two functions as follows.

$$\mathsf{MultCheck}((\epsilon_j)_j, m_i) = (\mathsf{MultCheck}_1((\epsilon_j)_j, m_i), \mathsf{MultCheck}_2((\epsilon_j)_j, m_i, (\alpha_i^{(i)})_j))$$

where $(\alpha_j^{(i)})_j = \mathsf{MultCheck}_1((\epsilon_j)_j, m_i)$. We note that the above programmings do not change the output distribution of random oracles.

We define bad events Bad consisting 6 small bad events Bad_i for i = 1, ..., 6 (i.e., Bad = $\bigvee_{i=1}^{6} \text{Bad}_i$). If the bad event happens, the game aborts and the adversary wins. Each small bad event is explained as follows.

- Bad_1 : A hash collision of H_1 or H_2 is found.
- Bad_2 : The PPT adversary $\mathcal B$ finds the preimage of KeyGen.
- Bad₃: There are too many repetitions which are illegitimately passed in the first challenge.
- Bad₄: In this case, the extracted message in fact passes the first challenge, but the message was not extracted at the time of querying to H_1 since the extractor lacks some query at that time. This event can be considered as preimage finding of the vector semi-commitment.

Algorithm 3. H_1 (salt, $\sigma_1 = (\text{com}_k, \Delta_k)_{k \in [\tau]}$):

```
1 h_1 \leftarrow_{\$} \{0,1\}^{2\lambda}
 2 if h_1 \in \mathcal{H}_1 then
  \mathbf{3} Raise \mathsf{Bad}_1 and abort
 4 h_1 \rightarrow \mathcal{H}_1.
 \mathbf{5} \ (\epsilon_1, \dots, \epsilon_{\tau}) \leftarrow \mathsf{ExpandH1}(h_1)
 6 Succ<sub>1</sub>[salt, h_1] \leftarrow \emptyset
 7 for k \in [\tau] do
            for (m_1, \ldots, m_N) \in \mathsf{Ext}(\mathsf{salt}, \mathcal{P}, \mathsf{com}_k) \ where \perp \notin (m_1, \ldots, m_N) \ \mathbf{do}
                  m_N \leftarrow m_N \oplus (0^{\lambda} \parallel \Delta_k)
  9
                 \operatorname{root} \leftarrow \operatorname{msb}_{\lambda} \left( \sum_{i \in [N]} m_i \right)
10
                  if KeyGen(sk) = pk then
11
                   Raise Bad2 and abort. // Secret key is found
12
                  (\alpha^{(i)}, v^{(i)})_{i \in [N]} \leftarrow \mathsf{MultCheck}\left(\epsilon_k, (m_i)_{i \in [N]}\right)
13
                 if \sum_{i \in [N]} v^{(i)} = 0 then
14
                   k \to \mathsf{Succ}_1[\mathsf{salt}, h_1] // Multiplication checking cheated
15
16 if |\mathsf{Succ}_1[\mathsf{salt}, h_1]| \geq \tau' then
       Raise Bad<sub>3</sub> and abort. // Too many cheated iterations
     (\mathsf{salt}, \sigma_1, h_1) \to \mathcal{Q}_1
19 Return h_1.
```

- Bad₅: At least two punctured messages are extracted, where two punctured points are distinct. Since rBN++ signature contains a path of GGM tree rather than leaves, those two punctured messages can be combined to a whole path for any undisclosed party. So, this event also means that the extractor finds a message which in fact passes the first challenge as Bad₄.
- Bad₆: there are too many repetitions which are illegitimately passed in the second challenge.

 Bad_4 and Bad_5 did not happen when the "individual" commitment size is 2λ bits (i.e., vector commitment), as it is hard to find multiple preimages of individual commitment itself. On the other hand, in our case (i.e., vector semicommitment), there may be a few preimages of a fixed commitment. Bad_4 describes an event that a directly found preimage from commitment queries passes the first challenge, and Bad_5 describes an event that multiple directly found punctured images from commitment queries passes the first challenge. We will show that the probability of such bad events are negligible in the following.

We have

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

where the probability of each bad event is analyzed as follows.

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

```
1 h_1 \rightarrow \mathcal{H}_1
  2 h_2 \leftarrow_{\$} \{0,1\}^{2\lambda}
 3 if h_2 \in \mathcal{H}_2 then
  4 Raise \mathsf{Bad}_1 and abort.
 5 h_2 \rightarrow \mathcal{H}_2, (salt, h_1, \sigma_2, h_2) \rightarrow \mathcal{Q}_2.
 6 if \exists k \in [\tau] such that \sum_{i \in [N]} v_k^{(i)} \neq 0 then
  7 | Return h_2.
 8 if \exists \sigma_1 \text{ such that } (h_1, \sigma_1) \in \mathcal{Q}_1 \text{ then}
           Parse \sigma_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)
11
12 else
13
       Return h_2.
14 Succ<sub>2</sub>[salt, h_2] \leftarrow \emptyset.
     for k \in [\tau] \setminus \mathsf{Succ}_1[\mathsf{salt}, h_1] do
            \mathcal{J} \leftarrow \emptyset
16
            for (m_1, \ldots, m_N) \in \mathsf{Ext}(\mathsf{salt}, \mathcal{P}, \mathsf{com}_k) do
17
                  if m_N \neq \bot then
18
                    | m_N \leftarrow m_N \oplus (0^{\lambda} \parallel \Delta_k)
19
                  if \bot \notin \{m_1, \ldots, m_N\} then
20
                        \mathsf{root} \leftarrow \mathsf{msb}_{\lambda} \left( \sum_{i \in [N]} m_i \right)
21
                        if KeyGen(root) = ct then
22
                          Raise Bad<sub>2</sub> and abort. // Secret key is found
\mathbf{23}
                  for i \in [N], \beta^{(i)} \leftarrow \mathsf{MultCheck}_1^{(i)} (\epsilon_k, m_i)
\mathbf{24}
                  for i \in [N], w^{(i)} \leftarrow \mathsf{MultCheck}_2^{(i)} (\epsilon_k, m_i, \alpha_k)
25
                  if \forall i \in [N], \ (\alpha_k^{(i)}, v_k^{(i)}) = (\beta^{(i)}, w^{(i)}) then
26
                   Raise \mathsf{Bad}_4 and abort.
27
                  else if \exists j, \forall i \in [N] \setminus \{j\}, (\alpha_k^{(i)}, v_k^{(i)}) = (\beta^{(i)}, w^{(i)}) then
28
                    j \to \mathcal{J}.
29
            if |\mathcal{J}| > 2 then
30
                 Raise \mathsf{Bad}_5 and abort.
31
            else if \mathcal{J} = \{\bar{i}_k\} then
32
             k \rightarrow \mathsf{Succ}_2[\mathsf{salt}, h_2]
33
34 if |\mathsf{Succ}_2[\mathsf{salt}, h_2]| \ge \tau - \tau' then
          Raise Bad<sub>6</sub> and abort. // Too many cheated iterations
36 Return h_2.
```

- Upper bounding Pr [Bad₁]. As $|h_1| = |h_2| = 2\lambda$, we have

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

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 OWF 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

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

– Upper bounding Pr [Bad₃]. 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 $\frac{u}{|\mathbb{R}|}$ and we have

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

- Upper bounding Pr [Bad₄]. Let us fix H_1 query, and assume that Bad₄ occurs at k-th iteration with (m_1, \ldots, m_N) . Denote $m_{i,j} \in \mathbb{F}$ be the (j+1)-th component of m_i for $j = 0, 1, \ldots, 2C + 1$, interpreting as \mathbb{F} -vector. We note that $m_{i,0} = \mathsf{seed}_i$ in Figure 3. Since (m_1, \ldots, m_N) is extracted messages, there exists ideal cipher queries such that

$$E_{m_{i,0}}(\mathsf{ctr}[\langle k \rangle_B, i, 0] \oplus \mathsf{salt}_1) = \mathsf{com}_i \quad \text{for } i \in [N], \tag{a}$$

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

Let A and B be the list of ideal cipher queries of the form (a) and (b), and $A \cup B$ be the union list of A and B while the orders are preserved. Let $i^* \in [N]$ be the party index of the last query in A. As $k \notin Succ_1[salt, h_1]$, the above query is done after the query to H_1 . We divide two cases as follows.

• Case 1: The query $E_{m_{i^*,0}}(\text{ctr}[\langle k \rangle_B, i^*, 0] \oplus \text{salt}_1) = \text{com}_{i^*}$ is the last query in $A \cup B$. Considering a fixed first challenge ϵ_k , the multiplication checking protocol is a set of linear relations of the shares. As all the other ideal cipher queries are fixed, there are at most 1 candidate of $m_{i^*,0}$ which passes the multiplication checking protocol per H_1 query (i.e., all the other variables in the linear relations are fixed, but only $m_{i^*,0}$ is not). Also, by (3) in Lemma 6, there are at most $\nu = \frac{2\lambda}{\log \lambda}$ of (N-1)-tuples $(m_{i,0})_{i \in [N] \setminus \{i^*\}}$ which satisfies (a) except with probability $14P/2^{\lambda}$. Since E should satisfy the equation for $m_{i^*,0}$, we have

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

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

• Case 2: The query $E_{m_{i^*,0}}(\text{ctr}[\langle k \rangle_B, i^*, j] \oplus \text{salt}_1) = m_{i^*,j}$ is the last query in $A \cup B$. Similar to Case 1, the candidate of $m_{i^*,j}$ is now at most 1. The difference is that there are at most P candidates of $m_{i^*,0}$, and E should satisfy both (a) and (b). So, there are at most νP candidates per H_1 query except $14P/2^{\lambda}$ probability by by (3) in Lemma 6. We have

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

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

All in all, we get

$$\Pr\left[\mathsf{Bad}_4\right] \le \frac{4\nu Q_1}{2^{\lambda}} + \frac{14P}{2^{\lambda}}$$

- Upper bounding Pr [Bad₅]. Let us fix the query to H_2 , and assume that Bad₅ occurs by two extracted messages in the 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 $i \in [N] \setminus \{j_1, j_2\}, {}^4 m''_{j_1} = m'_{j_1}$ 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

$$\Pr\left[\mathsf{Bad}_5 \land \neg \mathsf{Bad}_4\right] = 0.$$

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

- Upper bounding $\Pr[\mathsf{Bad}_6]$. If Bad_4 and Bad_5 do not happen, then $|\mathcal{J}| \leq 1$ for iterations in all queries to H_2 . So, we have

$$\Pr\left[\mathsf{Bad}_6\right] \le \frac{Q_2}{N^{\tau - \tau'}}.$$

All in all, we have

$$\Pr\left[\mathsf{Bad}\right] \leq \mathbf{Adv}_{\mathsf{KeyGen}}^{\mathsf{owf}}(\mathcal{B}) + \frac{Q^{2}}{2^{2\lambda}} + \frac{4\nu Q_{1}}{2^{\lambda}} + \frac{14P}{2^{\lambda}} + Q_{1} \cdot \sum_{i=\tau'}^{\tau} \binom{\tau}{i} \left(\frac{u}{|\mathbb{F}|}\right)^{i} \left(1 - \frac{u}{|\mathbb{F}|}\right)^{\tau-i} + \frac{Q_{2}}{N^{\tau-\tau'}}$$
(4)

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

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

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 $(\mathsf{salt}, \sigma_1, h_1) \in \mathcal{Q}_1$.

⁴ We note that all the subtrees which do not contain both j_1 and j_2 should choose from the same kind of message. This is because of the structure of the GGM tree.

- 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 $k^* \in \mathsf{Fail}$, and let

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

where
$$m_{\bar{i}_{k^*}} = \bot$$
 and $I_{k^*} = [N] \setminus \{\bar{i}_{k^*}\}$. Then, we have

$$\mathsf{Verify}(\mathsf{salt},\mathsf{com},\mathsf{pdecom}_{k^*},I_{k^*}) = (m_i)_{i \in I_{k^*}}.$$

If $(m_1, \ldots, m_N) \notin \mathsf{Ext}(\mathsf{com}, \Delta_{k^*})$, it means that there are more than u messages to be extracted (which is inhibited by the definition of the extractor). One can directly construct adversary \mathcal{C} against semi-binding security of VSC using \mathcal{A} as subroutine, so

$$\Pr\left[\mathcal{A} \text{ wins } \wedge (m_1, \dots, m_N) \notin \mathsf{Ext}(\mathsf{com}, \Delta_{k^*}) \mid \neg \mathsf{Bad}\right] \leq \mathbf{Adv}_{\mathsf{IC-VSC}}^{u\text{-esb}}(\mathcal{C}). \tag{5}$$

Now assume that $(m_1, \ldots, m_N) \in \operatorname{Ext}(\operatorname{com}, \Delta_{k^*})$. Since the bad events are only checked in H_1 and H_2 , the only chance to forgery is that additional ideal cipher queries after queries to H_1 and H_2 makes extra extracted messages to pass the challenges. It is equivalent to the event that the additional ideal cipher queries incur $\operatorname{\mathsf{Bad}}_4$ or $\operatorname{\mathsf{Bad}}_5$. From the probability computation above, we have

$$\Pr\left[\mathcal{A} \text{ wins } \wedge (m_1, \dots, m_N) \in \mathsf{Ext}(\mathsf{com}, \Delta_{k^*}) \mid \neg \mathsf{Bad}\right] \le \frac{4\nu Q_1}{2^{\lambda}} + \frac{14P}{2^{\lambda}}. \tag{6}$$

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

The EUF-CMA security is proved by game hopping technique as in the original BN++ scheme. Between the adjacent games, the security proof of BN++ mainly replaces a set of strings to random ones. Similarly, the security proof of rBN++ replaces strings in a straightforward manner, except one: commitment. Since the commitment size is λ bits, commitments cannot be replaced by random strings. This step is described in G_6 . Because of the page limit, we provide the sketch here and refer to Supplementary Material C for the full proof.

Theorem 2 (EUF-CMA Security of rBN++). Assume that H_0 , H_1 , H_2 , ExpandH1, and ExpandH2 are modeled as random oracles and that the (N, τ, λ) parameters of rBN++ are appropriately chosen where $|\mathbb{F}| = 2^{\lambda}$. For a PPT adversary \mathcal{A} against the EUF-CMA security of rBN++ with a total of Q_{sig} signing oracle queries, Q random oracle queries, and P ideal cipher queries, there exist PPT adversaries

- B against the EUF-KO security of rBN++,⁵

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

- C against the PRF security of H_3 ,⁶
- \mathcal{D} against the multi-instance hiding security of IC-VSC

such that

$$\begin{split} \mathbf{Adv}^{\text{euf-cma}}_{\text{rBN++}}(\mathcal{A}) \leq & \frac{(Q_{\text{sig}} + Q)^2}{2^{2\lambda}} + \frac{2Q_{\text{sig}}(Q_{\text{sig}} + P)}{2^{2\lambda}} + Q_{\text{sig}} \cdot \mathbf{Adv}^{\text{prf}}_{H_3}(\mathcal{C}) \\ & + \mathbf{Adv}^{Q_{\text{sig}}\text{-mih}}_{\text{IC-VSC}}(\mathcal{D}) + \mathbf{Adv}^{\text{euf-ko}}_{\text{rBN++}}(\mathcal{B}). \end{split}$$

Proof (Sketch). Let \mathcal{A} be an EUF-CMA adversary against rBN++ for given pk. Let G_0 be the original EUF-CMA game. Let \mathcal{O}_{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 only prove the security of the deterministic version of rBN++ ($\rho \leftarrow 0^n$) here. 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 , the abort probability is negligible.
- G₂: \mathcal{O}_{sig} replaces 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 $H_3(\mathsf{sk},\mu,\rho)$. 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 sk .
- G_3 : \mathcal{O}_{sig} samples $h_1 \in \{0,1\}^{2\lambda}$ at random instead of computing $H_1(\cdot)$, and programs the random oracle H_1 to output h_1 for the respective query. The simulation is aborted if the queries to H_1 have been made previously in a signing oracle query. As 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 negligible.
- G₄: \mathcal{O}_{sig} now samples $h_2 \in \{0,1\}^{2\lambda}$ at random instead of computing $H_2(\cdot)$, and also program the random oracle H_2 to output h_2 for the respective query. 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 negligible.
- G_5 : $\mathcal{O}_{\mathrm{sig}}$ replaces the seed of the unopened parties $\mathsf{seed}_{k_-}^{(\bar{\imath}_k)} \| \mathsf{tape}_k^{(\bar{\imath}_k)}$ in the IC-VSC with a random element for each $k \in [\tau]$. Since i_k 's are all random, the difference between this game and the previous one reduces to the hiding game of VSC.
- G_6 : $\mathcal{O}_{\mathrm{sig}}$ replaces $\mathsf{com}_k^{(\bar{\imath}_k)}$ with randomly sampled elements for each k. The difference between this game and the previous one reduces to indistinguishability of $\mathsf{com}_k^{(\bar{\imath}_k)}$ from uniform random. We prove the indistinguishability using the H-coefficient technique. See Supplementary Material \mathbb{C} for the details.

 $^{^6}$ H_3 itself is not a PRF, but it is used as a PRF with key prepending. We use this notation for convenience.

 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 sk and S-box outputs. As $\left((t_{k,j}^{(\bar{\imath}_k)})_{j \in [\ell]}, c_k^{(\bar{\imath}_k)}\right)_{k \in [\tau]}$ is random, the distribution of these variables does not change. If the multiplication triple is wrong, then $v_k^{(\bar{\imath}_k)} \leftarrow -\sum_{i \neq \bar{\imath}_k} v_k^{(i)}$ is different from an honest value derived from legitimate calculation. However, $(\bar{\imath}_k)$ is unopened and the multiplication check is still passed. Since the signature oracle in G_7 does not depend on the secret key sk , it implies that G_7 can be reduced to the EUF-KO security.

This proves the theorem.

4.3 Parameters and Efficiency

The parameters N and τ can be chosen to prevent the soundness attack [21]. 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

$$P_{1} = \sum_{k=\tau'}^{\tau} {\tau \choose k} \left(\frac{u}{|\mathbb{F}|}\right)^{k} \cdot \left(1 - \frac{u}{|\mathbb{F}|}\right)^{\tau-k}$$

$$P_{2} = \frac{1}{N^{\tau-\tau'}}$$

$$C = \min_{0 \le \tau' \le \tau} (1/P_{1} + 1/P_{2}).$$

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

Efficiency improvements of rBN++ 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 rBN++, 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 Δ 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.

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

Scheme	Field	N	τ	RO	PRG or IC	Sig. size
	Size			call	call	(B)
	2^{128}	16	33	532	1056C + 1518	1056C + 3792
BN++	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
	2^{128}	16	33	5	1056C + 1551	1056C + 2736
rBN++	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.

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 [27] has the probability $2^{-71.2}$), the application involves increasing τ for security. We choose AlMer v2.0 [25] for the performance measurement, as it is the best scheme for efficiency improvement. We will call it reduced AIMer (rAlMer). 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 future work.

Environment. We developed rAlMer in C, with AVX2 instructions. A significant portion of our implementation is based on the AIMer v2.0 source code. For other schemes, we used packages that have been officially submitted to NIST PQC standardization project, and all were compiled using the default options in compilation script. Our experiments were measured in 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 rAlMer 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 AlMer.

⁷ https://aimer-signature.org/

Scheme	pk (B)	sig (B)	Sign (Kc)	Verify (Kc)
Dilithium2 [26]	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 [27]	$1\bar{3}2$	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
AlMer- $v2.0-128f$ [25]	32	5,888	788	752
AlMer-v2.0-128s $[25]$	32	4,160	5,926	5,812
rAlMer-128f	$-\bar{3}2^{-}$	$-\bar{4},\bar{8}4\bar{8}$	421	-395
rAlMer-128s	32	3,632	2,826	2,730

*: -SHAKE256-simple

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

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

Additionally, rAlMer can be combined with the proof-of-work technique proposed in [2]. With this technique, the signature size of 128-bit secure rAlMer becomes 4.5 KB (fast) / 3.4 KB (small) without a significant performance degradation. See Supplementary Material D for the details.

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
- 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)

- 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)
- 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)
- 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
- 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)
- 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)
- 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
- 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, 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
- Huth, J., Joux, A.: Mpc in the head using the subfield bilinear collision problem.
 In: Reyzin, L., Stebila, D. (eds.) Advances in Cryptology CRYPTO 2024. pp. 39–70. Springer Nature Switzerland, Cham (2024)
- Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from Secure Multiparty Computation. In: ACM STOC 2007. pp. 21–30 (2007)

- 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)
- 22. 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
- 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)
- 24. 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, https://doi.org/10.1145/3576915.3616579
- 25. 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
- 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
- 27. 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
- 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)
- 29. 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

Supplementary Material

A Description of the rBN++ Signature Scheme

```
Algorithm 5. Sign(\mathsf{sk}, \mathsf{pk}, m) - reduced BN++, signing algorithm.
     // Phase 1: Committing to the views of the parties.
  1 Compute the hash of the message: \mu \leftarrow H_0(\mathsf{pk}, m)
  2 Sample randomness: \rho \leftarrow_{\$} \{0,1\}^{\lambda} \ (\rho \leftarrow 0^{\lambda} \text{ for deterministic signature})
     Compute salt: salt \leftarrow H_3(\mathsf{sk}, \mu, \rho).
  4 for each repetition k \in [\tau] do
           (\mathsf{com}_k, \mathsf{decom}_k, (\mathsf{seed}_k^{(i)} \parallel \mathsf{tape}_k^{(i)})_{i \in [N]}) \leftarrow
  5
             IC-VSC.Commit(salt\|\langle k \rangle_B, sk).
           for each gate q with index j do
  6
                if g is an addition with inputs (x_{k,j}, y_{k,j}) then
  7
                 8
                 if g is a multiplication with inputs (x_{k,j}, y_{k,j}) then
  9
                      For each party i, sample an output share:
10
                     z_{k,j}^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)}). Compute output offset and adjust first share:
11
                   \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 an helping values a: 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}.
12
13
           Compute c_k = \sum_j a_{k,j} \cdot b_{k,j}.
14
           For each party i, sample c_k^{(i)} \leftarrow \mathsf{Sample}(\mathsf{tape}_k^{(i)}).
           Compute offset and adjust first share : \Delta c_k = c_k - \sum_i c_k^{(i)},
            c_k^{(N)} \leftarrow c_k^{(N)} + \Delta c_k
17 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.
18 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.
19 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,i}^{(i)} and v_k^{(i)} be the broadcast values.
21 Set \sigma_2 \leftarrow (\text{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.
22 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.
23 for each repetition k \in [\tau] do
      \mathsf{pdecom}_k \leftarrow \mathsf{IC\text{-}VSC}.\mathsf{Open}((\mathsf{salt},k),\mathsf{decom}_k,[N]\setminus \{\bar{i}_k\}).
\textbf{25} \ \text{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 [C]})_{k \in [\tau]}).
```

Algorithm 6. Verify(pk, m, σ) - reduced BN++ signature scheme, verification algorithm.

```
1 Parse \sigma as (salt, h_1, h_2, (pdecom_k, \Delta c_k, (\Delta z_{k,j}, \alpha_{k,j}^{(\bar{i}_k)})_{j \in [C]})_{k \in [\tau]}).
2 Compute the hash value of the message: \mu \leftarrow H_0(\mathsf{pk}, m)
  3 Expand hashes:
           ((\epsilon_{k,j})_{j\in[2C+1]})_{k\in[\tau]}\leftarrow \mathsf{ExpandH1}(h_1) \text{ and } (\bar{i}_k)_{k\in[\tau]}\leftarrow \mathsf{ExpandH2}(h_2).
  4 for each repetition k \in [\tau] do
                ((\mathsf{seed}_k^{(i)} \parallel \mathsf{tape}_k^{(i)})_{i \in [N] \setminus \{\bar{i}_k\}}, \mathsf{com}_k) \leftarrow
   5
                  IC-VSC.Recon(salt\|\langle \tau \rangle_B, pdecom_k, [N] \setminus \{\bar{i}_k\}).
                for each party i \in [N] \setminus \{\bar{i}_k\} do
   6
                        for each gate g with index j do
   7
                               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)}.
                               \begin{array}{c} \textbf{if } g \textit{ is a multiplication with inputs } (x_{k,j}^{(i)}, y_{k,j}^{(i)}) \textbf{ then} \\ & \text{Sample an output share: } z_{k,j}^{(i)} \leftarrow \text{Sample}(\mathsf{tape}_k^{(i)}). \end{array}
 10
 11
                            \begin{array}{l} \textbf{if } i = N \textbf{ then} \\ & \triangle \textbf{ Adjust first share } z_{e,j}^{(i)} \leftarrow z_{e,j}^{(i)} + \Delta z_{e,j}. \\ & \triangle \textbf{ Sample } a_{k,j}^{(i)} \leftarrow \textbf{ Sample(tape}_k^{(i)}), \text{ and set } b_{k,j}^{(i)} = y_{k,j}^{(i)}. \end{array}
 12
 13
 14
                       \begin{aligned} & \text{Sample } c_k^{(i)} \leftarrow \text{Sample}(\mathsf{tape}_k^{(i)}) \\ & \text{if } i = N \text{ then} \end{aligned}
 15
 16
                        Adjust first share c_k^{(i)} \leftarrow c_k^{(i)} + \Delta c_k.
 17
                   Let 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)}.
20 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]}).
21 Set h'_1 \leftarrow H_1(\mu, \sigma_1).
22 for each parallel execution k \in [\tau] do
                for each party i \in [N] \setminus \{\bar{i}_k\} do
23
                        Simulate the triple checking protocol as in Section 2.4 for all
24
                 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)}.
26 Set \sigma_2 \leftarrow (\mathsf{salt}, (((\alpha_{k,j}^{(i)})_{j \in [C]}, v_k^{(i)})_{i \in [N]})_{k \in [\tau]}).
27 Set h'_2 = H_2(h'_1, \sigma_2).
28 Output Accept if h_1 = h'_1 and h_2 = h'_2.
29 Otherwise, output Reject.
```

B Proofs of Lemmas

B.1 Proofs for RO-VSC

Lemma 8 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

$$\begin{cases} H_{\mathsf{com}}(\mathsf{salt}, i, H_{\mathsf{tree}}(\mathsf{salt}, n)) = H_{\mathsf{com}}(\mathsf{salt}, i, H_{\mathsf{tree}}(\mathsf{salt}, n')), \\ H_{\mathsf{com}}(\mathsf{salt}, i, (H_{\mathsf{tree}}(\mathsf{salt}, n) \oplus n)) = H_{\mathsf{com}}(\mathsf{salt}, i, (H_{\mathsf{tree}}(\mathsf{salt}, n') \oplus n')) \end{cases}$$
(7)

is at most $8Q/2^{\lambda}$.

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 will consider that two \bot 's are not identical for simplicity. Let $H_{\mathsf{tree}}(n) = l$, $n \oplus l = r$, $H_{\mathsf{tree}}(n') = l'$, and $n' \oplus l' = r'$. We define

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

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

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

and let $Aux = Aux_1 \vee Aux_2 \vee Aux_3$. Then, by Markov's inequality, we have

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

For each query $H_{\mathsf{tree}}(n)$, we say Bad occurs if there exists $n' \neq n$ satisfies (7). Observing that Bad is the event of simultaneous two collisions, we divide Bad into left collisions L_i and right collisions R_i , up to the freshness of l and r.

 $- L_1 \Leftrightarrow l = l'$ or l is freshly queried, and $H_{\mathsf{com}}(l) = H_{\mathsf{com}}(l')$. Then, n should satisfies

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

- $L_2 \Leftrightarrow l$ is not freshly queried, $l \neq l'$, and $H_{\mathsf{com}}(l) = H_{\mathsf{com}}(l')$. Then, n should satisfies

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

- $R_1 \Leftrightarrow r = r'$ or r is freshly queried, and $H_{\mathsf{com}}(r) = H_{\mathsf{com}}(r')$. Then, 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 freshly queried, $r \neq r',$ and $H_{\mathsf{com}}(r) = H_{\mathsf{com}}(r').$ Then, n should satisfies

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

We have

$$\begin{split} \Pr\left[\mathsf{Bad}\right] & \leq \Pr\left[\mathsf{Aux}\right] + \sum_{i,j \in [2]} \Pr\left[\mathsf{L}_i \land \mathsf{R}_j \land \neg \mathsf{Aux}\right] \\ & \leq \Pr\left[\mathsf{Aux}\right] + \Pr\left[\mathsf{L}_1 \land \mathsf{R}_1\right] + \Pr\left[\mathsf{L}_2 \land \mathsf{R}_1 \land \neg \mathsf{Aux}_1\right] \\ & + \Pr\left[\mathsf{L}_1 \land \mathsf{R}_2 \land \neg \mathsf{Aux}_2\right] + \Pr\left[\mathsf{L}_2 \land \mathsf{R}_2 \land \neg \mathsf{Aux}_3\right] \end{split}$$

Each term can be bounded as follows.

- Pr [L₁ \wedge R₁]: This can be divided into 4 sub-cases; L₁ has 2 cases, and R₁ has 2 cases. The first case l = l' and r = r' at the same time cannot achieved since it implies n = n'. Since the probability of each sub-cases is upper bounded by $Q_t^2/2^{2\lambda}$, this case is upper bounded by $3Q_t^2/2^{2\lambda}$.
- $\Pr[\mathsf{L}_2 \land \mathsf{R}_1 \land \neg \mathsf{Aux}_1]$: Since L_2 and r = r' cannot happen at the same time, this probability is upper bounded by $Q_t Q_c / 2^{2\lambda}$.
- Pr [L₁ \wedge R₂ \wedge ¬Aux₂]: Similarly as the previous case, this is upper bounded by $Q_tQ_c/2^{2\lambda}$.
- Pr [L₂ \wedge R₂ \wedge ¬Aux₃]: L₂ \wedge R₂ means that $(n, l, l', r, r') \in \mathcal{L}_3$. So, this probability is upper bounded by $Q_c/2^{\lambda}$.

Then, we have

$$\Pr\left[\mathsf{Bad}\right] \leq \frac{2Q_t + Q_c}{2^\lambda} + \frac{3Q_t^2 + 5Q_tQ_c}{2^{2\lambda}} + \frac{2Q_tQ_c^2}{2^{3\lambda}} + \frac{Q_tQ_c^3}{2^{4\lambda}} \leq \frac{8Q_tQ_c^2}{2^{2\lambda}}$$

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

Lemma 3 Let $H_{\mathsf{com}}: \{0,1\}^* \to \{0,1\}^{\lambda}$ and $H_{\mathsf{tree}}: \{0,1\}^* \to \{0,1\}^{\lambda}$ be random oracles. Let \mathcal{A} be an arbitrary adversary that makes Q queries to the random oracles. Then \mathcal{A} 's u-extractable semi-binding advantage $\mathbf{Adv}^{u\text{-esb}}_{\mathsf{RO-VSC}}(\mathcal{A})$ against RO-VSC is bounded by

$$\mathbf{Adv}_{\mathsf{RO-VSC}}^{u\text{-esb}}(\mathcal{A}) \leq \frac{10Q}{2^{\lambda}},$$

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

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_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 $H_{\mathsf{tree}}(x) = \bot$ (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 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

$$S_{0,i} = \{s : H_{com}(i, s) = com_i\}$$

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

$$A_i = \{(p_1, \dots, p_d) : p_e \in S_{e-1, i_{e-1}} \text{ for } e \in [d]\}$$

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

$$S = \Big\{ (s_1, \dots, s_N) : s_i = \bot \text{ if } S_{0,i} = \emptyset \text{ and } s_i \in S_{0,i} \text{ otherwise,}$$

$$\mathsf{Recon}(p_1, \dots, p_d, \mathsf{com}_i, I = [N] \setminus \{i\}) = (s_i)_{i \in I} \text{ for } (p_1, \dots, p_d) \in A_i \Big\}$$

5. Finally, Ext outputs arbitrary u or less elements in S.

We define some bad events.

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

By Lemma 8 and (1),

$$\Pr\left[\mathsf{Bad}_1 \vee \mathsf{Bad}_2\right] \le \frac{10Q}{2^{\lambda}} \tag{8}$$

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

- As S contains all possible (pdecom_I, I), 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

$$|S| \le 2N \left(\frac{\lambda}{\log \lambda}\right)^2.$$

Therefore, A cannot win the game without bad events so we have

$$\mathbf{Adv}^{u\text{-esb}}_{\mathsf{RO-VSC}}(\mathcal{A}) \leq \Pr\left[\mathsf{Bad}_1 \vee \mathsf{Bad}_2\right] \leq \frac{10Q}{2^{\lambda}}$$

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

Lemma 4 Let $H_{\mathsf{com}} : \{0,1\}^* \to \{0,1\}^{\lambda}$ and $H_{\mathsf{tree}} : \{0,1\}^* \to \{0,1\}^{\lambda}$ be random oracles. Let \mathcal{A} be an arbitrary adversary that makes Q queries to the random oracles. Then \mathcal{A} 's multi-instance hiding advantage $\mathbf{Adv}^{Q_I-\mathsf{mih}}_{\mathsf{RO-VSC}}(\mathcal{A})$ against RO-VSC is bounded by

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

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 node is derived by $\overline{\mathsf{node}} = m_{\tilde i} \oplus m_{((\tilde i-1) \oplus 1)+1}$. 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, $\Pr[\mathsf{Bad}_1] \leq Q_T^2/2^{2\lambda}$.
- Bad_2 : a query $(\mathsf{salt}, m_{\overline{i}})$ is queried to H_{com} . $\Pr[\mathsf{Bad}_2 \land \neg \mathsf{Bad}_1] \leq Q_c/2^{\lambda}$.
- Bad_3 : a query ($\mathsf{salt}, \overline{\mathsf{node}}$) is queried to H_{tree} . $\Pr[\mathsf{Bad}_3 \land \neg \mathsf{Bad}_1] \leq Q_t/2^{\lambda}$.

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 $m_{\tilde{i}}$ 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

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

B.2 Proofs for IC-VSC

Lemma 5 Let $E: \{0,1\}^{\lambda} \times \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ be an ideal cipher and $\sigma: \{0,1\}^{\lambda} \to \{0,1\}^{\lambda}$ be an orthomorphism. Let \mathcal{A} be an arbitrary adversary that

makes Q queries to E. Then, the probability that A finds $\mathsf{salt} = (\mathsf{salt}_1, \mathsf{salt}_2, \mathsf{b}) \in \{0, 1\}^{\lambda + \lambda + 8}, i \in \{0, 2, \dots, 254\}, \text{ and distinct } n, n' \in \{0, 1\}^{\lambda} \text{ such that }$

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

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

Proof. At the end of game, we define $E_k(x) = \bot$ for non-queried input (k,x) to E, and we will consider that two \bot 's are not identical for simplicity. We also denote $\mathsf{ctr}_l = \mathsf{0b0}^{\lambda-24} \|\mathsf{b}\| \langle i \rangle_B \| \langle 0 \rangle_B$ and $\mathsf{ctr}_r = \mathsf{0b0}^{\lambda-24} \|\mathsf{b}\| \langle i+1 \rangle_B \| \langle 0 \rangle_B$ for readability. Let $\sigma(n) \oplus E_{\mathsf{salt}_2}(n \oplus \mathsf{salt}_1) = l, n \oplus l = r, \sigma(n') \oplus E_{\mathsf{salt}_2}(n' \oplus \mathsf{salt}_1) = l',$ and $n' \oplus l' = r'$. We define

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

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

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

and let $Aux = Aux_1 \vee Aux_2 \vee Aux_3$. Then, by Markov's inequality, we have

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

For each query $E_{\mathsf{salt}_2}(n \oplus \mathsf{salt}_1)$, we say Bad occurs if there exists $n' \neq n$ satisfies (2). Observing that Bad is the event of simultaneous two collisions, we divide Bad into left collisions L_i and right collisions R_i , up to the freshness of l and r.

- $\mathsf{L}_1 \Leftrightarrow l = l'$ or l is freshly queried, and $E_l(l \oplus \mathsf{salt}_1) = E_{l'}(l' \oplus \mathsf{salt}_1)$. Then, n should satisfies

$$\begin{split} (\sigma(n) \oplus E_{\mathsf{salt}_2}(n \oplus \mathsf{salt}_1) = l') \vee \\ (\sigma(n) \oplus E_{\mathsf{salt}_2}(n \oplus \mathsf{salt}_1) \neq l' \wedge E_l(\mathsf{ctr}_l \oplus \mathsf{salt}_1) = E_{l'}(\mathsf{ctr}_l \oplus \mathsf{salt}_1)). \end{split}$$

- $L_2 \Leftrightarrow l$ is not freshly queried, $l \neq l'$, and $E_l(l \oplus \mathsf{salt}_1) = E_{l'}(l' \oplus \mathsf{salt}_1)$. Then, n should satisfies

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

- $R_1 \Leftrightarrow r = r'$ or r is freshly queried, and $E_r(r \oplus \mathsf{salt}_1) = E_{r'}(r' \oplus \mathsf{salt}_1)$. Then, n should satisfies

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

- $R_2 \Leftrightarrow r$ is not freshly queried, $r \neq r'$, and $E_r(r \oplus \mathsf{salt}_1) = E_{r'}(r' \oplus \mathsf{salt}_1)$. Then, n should satisfies

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

We have

$$\begin{split} \Pr\left[\mathsf{Bad}\right] & \leq \Pr\left[\mathsf{Aux}\right] + \sum_{i,j \in [2]} \Pr\left[\mathsf{L}_i \land \mathsf{R}_j \land \neg \mathsf{Aux}\right] \\ & \leq \Pr\left[\mathsf{Aux}\right] + \Pr\left[\mathsf{L}_1 \land \mathsf{R}_1\right] + \Pr\left[\mathsf{L}_2 \land \mathsf{R}_1 \land \neg \mathsf{Aux}_1\right] \\ & + \Pr\left[\mathsf{L}_1 \land \mathsf{R}_2 \land \neg \mathsf{Aux}_2\right] + \Pr\left[\mathsf{L}_2 \land \mathsf{R}_2 \land \neg \mathsf{Aux}_3\right] \end{split}$$

Each term can be bounded as follows.

- Pr [L₁ \wedge R₁]: This can be divided into 4 sub-cases; L₁ has 2 cases, and R₁ has 2 cases. The first case -l = l' and r = r' at the same time cannot achieved since it implies n = n'. Since the probability of each sub-cases is is upper bounded by $\binom{Q}{2}/(2^{\lambda} Q)^2$, this case is upper bounded by $6Q^2/2^{2\lambda}$.
- Pr [L₂ \wedge R₁ \wedge ¬Aux₁]: Since L₂ and r = r' cannot happen at the same time, this probability is upper bounded by $Q^2/2^{2\lambda}$.
- Pr [L₁ \wedge R₂ \wedge ¬Aux₂]: Similarly as the previous case, this is upper bounded by $Q^2/2^{2\lambda}$.
- Pr [L₂ \wedge R₂ \wedge ¬Aux₃]: L₂ \wedge R₂ means that $(n, l, l', r, r') \in \mathcal{L}_3$. So, this probability is upper bounded by $Q/2^{\lambda}$.

Then, we have

$$\Pr\left[\mathsf{Bad}\right] \leq \frac{3Q}{2^\lambda} + \frac{11Q^2}{2^{2\lambda}} + \frac{2Q^3}{2^{3\lambda}} + \frac{Q^4}{2^{4\lambda}} \leq \frac{10Q}{2^\lambda}$$

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

C Full Proof of the EUF-CMA Security

Theorem 2 (EUF-CMA Security of rBN++). Assume that H_0 , H_1 , H_2 , ExpandH1, and ExpandH2 are modeled as random oracles and that the (N, τ, λ) parameters of rBN++ are appropriately chosen where $|\mathbb{F}| = 2^{\lambda}$. For a PPT adversary A against the EUF-CMA security of rBN++ with a total of Q_{sig} signing oracle queries, Q random oracle queries, and P ideal cipher queries, there exist PPT adversaries

- $-\mathcal{B}$ against the EUF-KO security of rBN++,8
- C against the PRF security of H_3 ,
- \mathcal{D} against the multi-instance hiding security of IC-VSC

such that

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

Proof. Let \mathcal{A} be an EUF-CMA adversary against rBN++ for given pk. Let G_0 be the original EUF-CMA game. Let \mathcal{O}_{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 rBN++ ($\rho \leftarrow 0^n$), 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

$$\Pr[\mathsf{G}_1 \text{ aborts}] \le \frac{(Q_{\text{sig}} + Q_0)^2}{2^{2\lambda}}.$$

G₂: \mathcal{O}_{sig} replaces salt $\in \{0,1\}^{2\lambda}$ and root seeds $(\text{seed}_k)_{k\in[\tau]} \in (\{0,1\}^{\lambda})^{\tau}$ with randomly sampled values, instead of computing $H_3(\mathsf{sk},\mu,\rho)$. 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 sk . Therefore, there exists a PPT adversary \mathcal{C} against the PRF security of H_3 such that

$$|\Pr[\mathcal{A} \text{ wins } \mathsf{G}_1] - \Pr[\mathcal{A} \text{ wins } \mathsf{G}_2]| \leq Q_{\mathrm{sig}} \cdot \mathbf{Adv}_{H_3}^{\mathsf{prf}}(\mathcal{C}).$$

 G_3 : \mathcal{O}_{sig} samples $h_1 \in \{0,1\}^{2\lambda}$ at random instead of computing

$$H_1(\mu,\mathsf{salt},(\mathsf{com}_k,(\mathsf{pk}_k^{(i)})_{i\in[N]},\Delta c_k,(\Delta z_{k,j})_{j\in[C]})_{k\in[\tau]})$$

 $^{^{8}}$ We assume ${\cal B}$ has the same amount of queries to random oracles.

 $^{^9}$ H_3 itself is not a PRF, but it is used as a PRF with key prepending. We use this notation for convenience.

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

$$\Pr[\mathsf{G}_3 \text{ aborts}] \le \frac{Q_{\text{sig}}(Q_{\text{sig}} + Q_1)}{2^{2\lambda}}.$$

 G_4 : $\mathcal{O}_{\mathrm{sig}}$ now samples $h_2 \in \{0,1\}^{2\lambda}$ at random instead of computing

$$H_2(h_1, \mathsf{salt}, ((\alpha_k^{(i)})_{i \in [N]}, (v_k^{(i)})_{i \in [N]})_{k \in [\tau]})$$

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}_{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

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

 G_5 : $\mathcal{O}_{\mathrm{sig}}$ replaces the seed of the unopened parties $\mathsf{seed}_{k_-}^{(\bar{i}_k)} \| \mathsf{tape}_k^{(\bar{i}_k)} \|$ in the IC-VSC with a random element for each $k \in [\tau]$. Since 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

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

 G_6 : $\mathcal{O}_{\mathrm{sig}}$ 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

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

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

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

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,

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

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

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

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 $\sum_{\text{seed}} P_{\text{seed}} = P$ and $\sum_{\text{seed}} P'_{\text{seed}} = Q_{\text{sig}}$. 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 sk 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 sk. 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 sk, 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 rBN++ such that

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

All in all, we have

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

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}_{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} .

D 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.

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 rBN++ scheme since the rBN++ scheme requires each root of the τ trees to be fixed.

For the second one, the last challenge hash (H_2 in the rBN++ 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 rBN++ 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 rBN++ scheme calls almost no random oracle per signature, there may be some computation overhead.

N	au	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	$\bar{3}0447$	3632
256	16	8	271	28656	3426

Table 3: The number of calls to the random oracles and the ideal cipher for rAlMer 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.