BACON: An Improved Vector Commitment Construction with Applications to Signatures

Yalan Wang[1,⋆], Bryan Kumara[2,⋆], Harsh Kasyap[2,3], Liqun Chen[1], Sumanta Sarkar[4], Christopher J.P. Newton[1], Carsten Maple[2,3], Ugur Ilker Atmaca[2,3]

Abstract

All-but-one Vector Commitments (AVCs) allow a committed vector to be verified by randomly opening all but one of the committed values. Typically, AVCs are instantiated using Goldwasser-Goldreich-Micali (GGM) trees. Generating these trees comprises a significant computational cost for AVCs due to a large number of hash function calls. Recently, correlated GGM (cGGM) trees were proposed to halve the number of hash calls and Batched AVCs (BAVCs) using one large GGM tree were integrated to FAEST to form the FAEST version 2 signature scheme, which improves efficiency and reduces the signature size. However, further optimizations on BAVC schemes remain possible. Inspired by the large-GGM based BAVC and the cGGM tree, this paper proposes BACON, a BAVC with aborts scheme by leveraging a large cGGM tree. BACON executes multiple instances of AVC in a single batch and enables an abort mechanism to probabilistically reduce the commitment size. We prove that BACON is secure under the ideal cipher model and the random oracle model. We also discuss the possible application of the proposed BACON, i.e., FAEST version 2. Furthermore, because the number of hash calls in a large cGGM tree is halved compared with that used in a large GGM tree, theoretically, our BACON is more efficient than the state-of-the-art BAVC scheme.

1 Introduction

Zero-knowledge proofs (ZKPs) allow a prover to convince a verifier that a statement is true, and the verifier will learn nothing beyond the validity of the statement. The Multi-party Computation-in-the-Head (MPCitH) paradigm [35] is a popular method to create post-quantum digital signatures using a non-interactive zero-knowledge (NIZK) proof, which proves that a signer knows a secret witness to the output of a function. MPCitH-style signatures can be

[1]University of Surrey, {yw0010,liqun.chen,c.newton}@surrey.ac.uk,

[2]The Alan Turing Institute, kumara.bryan@gmail.com, {hkasyap,cmaple,uatmaca}@turing.ac.uk,

[3]University of Warwick,

[4]University of Essex, sumanta.sarkar@essex.ac.uk, ⋆ Joint first authors

very efficient for small to medium-sized circuits and have been proposed as candidates for the NIST competition on post-quantum signatures [12, 21, 22, 36]. However, one drawback of the MPCitH-style signature is the proof size is linear with the size of the underlying circuit which renders this approach inefficient for large-sized circuits.

In 2018, Boyle *et al.* [15] proposed a new ZKP framework that makes use of Vector Oblivious Linear Evaluation (VOLE) correlations to create a commit-and-prove ZK protocol. Recently, several interactive ZK protocols [11, 13, 18, 24–26, 30, 39–42, 44–46] have been proposed, which use subfield VOLE (sVOLE) correlations. We will refer to these protocols as VOLE-ZK, which provide fast end-to-end runtimes and are scalable for large circuits compared to MPCitH-based ZK proofs. However, the aforementioned VOLE-based ZK protocols are restricted to a designated-verifier setting[1]. To overcome this restriction, in 2023, Baum *et al.* [9] proposed the VOLE-in-the-Head (VOLEitH) enabled VOLE-based ZK proofs to be publicly verifiable. The VOLEitH-style signature can be instantiated using AES as the one-way function. This creates the FAEST signature scheme, which is a second-round candidate of the NIST additional post-quantum signatures competition.

A key building block in both MPCitH and VOLEitH is the *All-but-one* Vector Commitment (AVC). Informally, an AVC scheme consists of four PPT algorithms, a Setup algorithm which establishes the parameters, a Commit algorithm which binds a hidden vector of values to commitment values, an Open algorithm which randomly opens (i.e., outputs) all but one of the previously committed values and a Verify algorithm which ensures that the committed values and the hidden values are consistent.

1.1 Related work

In current instantiations of AVC schemes, an initial seed acting as the root of the Goldreich-Goldwasser-Micali (GGM) tree [31] is extended to generate a set of pseudorandom numbers by using a length-doubling pseudorandom generator (PRG) recursively. Revealing all but one leaf allows verification of the commitments but retains the secrecy of the prover's witness. However, this construction requires a logarithmic number of tree nodes to be sent to the verifier. Guo *et al.* [33] replaced the length-doubling PRG with the Davies-Meyer circular correlation robust (CCR) hash function [43] and proposed a new tree structure, called a *correlated* GGM (cGGM) tree. In a cGGM tree, the left child node is the hash output of the parent node while the right child node is computed by XORing the parent node and the left child node. Then the sum of all nodes in one layer is a constant value. The cGGM tree can reduce the number of hash calls by half compared with the GGM tree, therefore this tree is also referred

[1]In a designated-verifier setting, a specified party, i.e., the "designated verifier", is introduced to prevent the verifier from turning around and convincing others that they have the secret. The proof is constructed in such a way that only the designated verifier can verify the proof, and no one else can. This provides stronger privacy guarantees, as the prover can be assured that only the intended verifier can learn anything about the secret.

to as a “half tree”. To be compatible with the sVOLE approach [16, 40], whose security relies on the pseudorandomness of leaf nodes, the authors use the PRG when generating the leaf layer (to break down the correlation of the nodes in the leaf layer). This new GGM tree is called the pseudorandom cGGM tree (pcGGM).

Bui *et al.* [19] applied a single cGGM tree to construct an AVC scheme and broke the correlation using the PRG (the same method to generate the leaf nodes in a GGM/pcGGM tree), when generating the messages and commitments from the leaf nodes. Furthermore, they also constructed an All-but-τ Vector Commitment (AVC) scheme by running τ cGGM trees in parallel. This all-but-τ vector commitment scheme meets the notion of Batched All-but-one Vector Commitments (BAVC) proposed in [8]. In [8], Baum *et al.* firstly proposed the notion of BAVC formally. Instead of generating a forest of τ trees in parallel, they made use of a big GGM tree that can be considered as τ small GGM trees connected together to construct a BAVC with aborts mechanism scheme. Opening all but τ leaves of the big GGM tree is more efficient than opening all but one leaf node in each of the τ small GGM trees in parallel. Because with high probability, some authentication paths in the big GGM tree will merge. This can reduce the number of internal nodes to be opened. Moreover, they mapped entries of the individual vector commitments to the leaves of the tree in an interleaved way. To prevent certain variability in the signature size due to choices of opening nodes, they set a threshold for the number of internal nodes to be opened; and abort the opening stage if the number of opened nodes exceeds the threshold. They implemented the FAEST scheme instantiated with their BAVC with aborts scheme. The results show that their improved FAEST is more efficient than the original FAEST. Further, they integrated the large GGM tree-based BAVC scheme [8] into FAEST to form the FAEST version 2 [10], released by NIST. Very recently, the FAEST team’s new work eplaced the hash function calls and revisited the evaluation of the AES block cipher to achieve improvements on both security and practical efficiency [7], which was just accepted by CRYPTO 2025 and the full paper is not accessible at present. However, the CCR hash construction [32] can only meet the 128-bit security when it is constructed by AES. When the security level is 192 or 256 bits, they applied Rijndael algorithm to construct CCR hash functions. After that, Cui *et al.* [20] proposed a new tweaked CCR hash algorithm constructed by using a standard AES algorithm, which can meet all three security levels, i.e., 128, 192, and 256 bits. Moreover, they also presented a single cGGM tree-based AVC scheme based on a non-tweaked CCR hash function, which is a simplified version of the proposed tweaked CCR hash [20].

Based on current literature, on the one hand, cGGM tree is the most efficient tree compared to the GGM tree and pcGGM tree. On the other hand, using one big tree is better than using multiple small trees in parallel to construct BAVC schemes. Therefore, even using the state-of-the-art, i.e., a big GGM tree, to construct BAVC schemes is not enough. *There is still room for improving the performance of BAVC schemes by examining efficient tree structures in theory.*

Except the efficiency problem in GGM/cGGM tree-based vector commitment schemes, there has been a line of research on the use of salts in GGM/cGGM to makes these vector commitment schemes against collision attack. This mainly originated after Dinur and Nadler [23] introduced the so-called multi-target attack on Picnic: they showed that after querying $2^{\lambda/2}$ (λ is the security level) signatures, an adversary is likely to find two signatures having the same master seed, which eventually leads to recovering the signing key. A general approach to mitigating this attack is to apply a 2λ-*bit salt* during signature generation, as in Picnic [47] and other MPC-in-the-Head signatures, including BBQ [21], Banquet [12], SDitH [3], MIRA [5], MiRitH [2], MQOM [28], PERK [1], Ryde [6], and Biscuit [14]. However, the application of salt has taken diverse forms. For instance, in [27], the signature description is based on a tree-based fashion using the master seed and the salt. When they implemented the scheme, they leveraged the AES in counter (CTR) mode. [12] and [4] suggested applying the salt to the leaves of the trees. Later, however, [17] showed that this approach fails to prevent the multi-target attack, and proposed using a salt at every node as input to the PRG, where the salt is derived from a *global* salt. In [37], Kim *et al.* also made use of a global salt in the generation of every right child node in the cGGM tree and introduced the concept of vector semi-commitment, which relaxes the binding property but does not address the use of a large cGGM tree to improve efficiency.

Therefore, *the question remains if we can propose a more efficient tree structure to construct BAVC schemes while being against the multi-target attack simultaneously or not.*

1.2 Our contributions

In this paper, to improve the efficiency, we propose one big cGGM tree to construct a new BAVC with aborts scheme called BACON, which is more efficient than the state-of-the-art BAVC with aborts scheme [8]. In short, a prover generates one big cGGM tree with sufficient leaves to be used for multiple AVCs. A verifier will provide a challenge index to reveal all but one value in each AVC. Then a prover can send the minimum number of cGGM tree nodes to construct an authentication path for verification. The abort mechanism allows the prover to request a new challenge index if the number of tree nodes exceeds a threshold at a small computational cost. After receiving a set of opened cGGM nodes, the verifier recomputes the partial cGGM tree recursively. Finally, the verifier checks whether the computed commitment matches the given commitment. Choosing the cGGM tree significantly reduces the cost of tree generation–specifically, given a parent node, generating two child nodes requires only a single invocation of the underlying hash function. Furthermore, to resist collision attacks, we apply a iv value as a master salt to internal nodes of the cGGM tree. Moreover, we use AES in CTR mode, which allows the salt to be varied easily using a counter. Our contributions are as follows:

1. We propose a new big cGGM tree to construct a new BAVC scheme, called BACON. Moreover, for notation convenience, we make use of state-of-the-

art non-tweaked CCR hash function from [20] constructed using only AES to meet all security levels.

2. We provide formal security proofs for our scheme, BACON, under the ideal cipher model and random oracle model.
3. We discuss the application of our BACON to FAEST/FAEST version 2/ [7].
4. We compare our BACON with the large GGM tree-based BAVC with aborts scheme [8] in theory and show our BACON is more efficient.

1.3 Paper organization.

We introduce preliminaries in Section 2. In Section 3, we define the syntax of a BAVC with aborts scheme and introduce our proposed scheme BACON. In Section 4.3, we prove our scheme BACON meets extractable binding and hiding. Then we discuss the application of our proposed BACON in Section 5. Furthermore, we compare BACON against the state-of-the-art BAVC scheme to show theoretical improvements. We conclude our work in Section 7.

2 Preliminaries

This section introduces the GGM tree and its variants, alongside the concept of BAVC with aborts.

2.1 GGM Tree and its variants

Cryptographic protocols such as MPCitH-style signatures require large amounts of random inputs. Generating them directly can be computationally costly. A more efficient approach is to start with an initial random value and expand it to a large number of values that are computationally indistinguishable from pseudorandom values. Generating pseudorandom values can be achieved by creating a binary tree. A GGM tree is a type of binary tree, which expands a random seed into a tree of pseudorandom values by recursively applying a length-doubling Pseudorandom Generator $PRG : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ to create a pseudorandom left and right child node. However, the repeated calls to length-doubling PRG for generating child nodes is computationally costly as one PRG call consists of two hash calls.

A cGGM tree was proposed [33], which halves the computation cost by XORing the left node and the parent node instead of a hash call to generate the right child node. This is done using a CCR hash function H where the left node is defined as $H(parent)$ and the right node as $left \oplus parent = H(parent) \oplus parent$ [33], $parent$ denotes a parent node. For non-leaf and non-root nodes of the tree, cGGM reduces the number of calls to hash functions by half compared to GGM trees as only one hash function is needed in the CCR function. An alternative to the cGGM tree is the pcGGM tree [33], which is similar to a cGGM tree except for the last layer - the tree leaves are generated using the same way in the GGM tree. In specific, the left leaf is defined as $H(parent \oplus 0)$ and the right leaf as $H(parent \oplus 1)$.

2.2 Concept of BAVC with aborts

Here, we follow the formal notion of BAVC with aborts in [8]. Let λ be the security parameter and $\tau \in \mathbb{N}$ be the repetition parameter which determines the number of AVCs required. Each AVC has a length $N_\alpha \in \mathbb{N}$, $\alpha \in [\tau]$. A GGM-type tree has $d \in \mathbb{N}$ layers. The total number of leaves across all AVCs is calculated as $L = \Sigma_{\alpha \in [\tau]} N_\alpha$. Define the challenge index set as $I = (i^{(1)}, ..., i^{(\tau)}) \subset [N_1] \times ... \times [N_\tau]$. A BAVC with aborts scheme is defined as follows:

Definition 1 *(BAVC with aborts) A BAVC with aborts scheme is denoted as BAVC=(Setup, Commit, Open, Verify) (with message space $\mathcal{M}$). Let E, E^{-1} be the ideal cipher oracles.*

- *Setup$(1^\lambda) \rightarrow$ pp: This setup algorithm is run by the sender, taking the security parameter λ as input and outputting public parameters pp, which is input to all other algorithms.*

- *Commit(pp) $\rightarrow$ (com, decom, $\boldsymbol{m} = (m_1^{(\alpha)}, ..., m_{N_\alpha}^{(\alpha)})_{\alpha \in [\tau]}$): This algorithm is run by the sender. Given the public parameters pp as input and outputs a commitment com with opening information decom for the message vector $\boldsymbol{m} = (m_1^{(\alpha)}, ..., m_{N_\alpha}^{(\alpha)})_{\alpha \in [\tau]} \in \mathcal{M}^{N_1 + \cdots + N_\tau}$.*

- *Open(decom, I) $\rightarrow$ decom$_I \vee \perp$: This algorithm is run by the sender. On input an opening decom and the index vector $I \subset [N_1] \times \cdots \times [N_\tau]$ chosen randomly by the sender, output $\perp$ or an opening decom$_I$ for I.*

- *Verify(com, decom$_I$, I) $\rightarrow \left(\left(m_j^{(\alpha)} \right)_{j \in [N_\alpha] \setminus \{i_\alpha\}} \right)_{\alpha \in [\tau]} \vee \perp$: This algorithm is run by the receiver. Given a commitment com, an opening decom$_I$, either output all messages $\left(m_j^{(\alpha)} \right)_{j \in [N_\alpha] \setminus \{i_\alpha\}}$ (accept the opening) or $\perp$ (reject the opening).*

3 BACON: a big cGGM-based BAVC with aborts scheme

In this section, we introduce our big cGGM-based BAVC with aborts scheme in detail. We name this scheme as BACON.

In our BACON, we adopt the cGGM tree structure [33] to propose a big cGGM tree using the **non-tweaked AES-based CCR hash** version in [20]

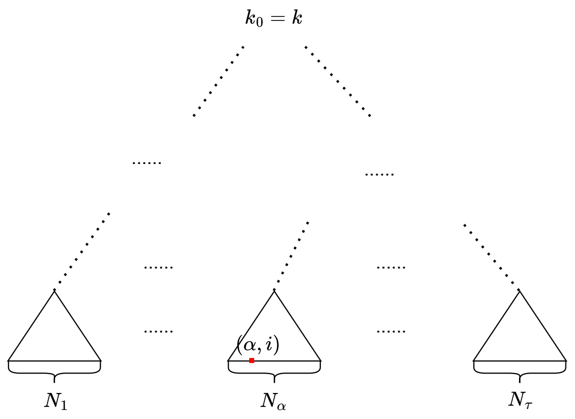

Figure 1: A mapping π in a the big cGGM tree. In this big cGGM tree, the number of leaf nodes $L = \sum_{i=1}^{\tau} N_i$. Every triangle can be considered as a small cGGM tree in the head after the mapping π.

to achieve all security levels. Moreover, to be against the collision attack, we add an **iv** value as a salt during the generation of the large cGGM generation. Following the notations defined in section 2.2, let L be the number of leaf nodes, d is the layer of the cGGM tree, so $L = 2^d$. Let $\pi : [0, 2^d - 1] \to \{(\alpha, i)\}_{1 \leq i \leq N_\alpha}$ as a bijective function mapping each leaf of the big cGGM tree to each AVC (shown in Fig. 1), T_{open} be the threshold of the number of opened internal nodes, $\mathsf{H}_{\mathsf{CCR}} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a CCR hash function defined in section 4.1, and $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{2\lambda}$ be a collision-resistant hash function. BACON is shown in Fig. 2 which consists of four algorithms, i.e., BACON.Setup, BACON.Commit, BACON.Open and BACON.Verify. The BACON.Setup algorithm simply establishes the public parameters. In BACON.Commit, a single big cGGM tree with L leaves is generated. All leaves are then interleaved into τ AVCs using the mapping π. A message and its corresponding commitment can be computed given a mapped leaf $\mathsf{sd}_i^{(\alpha)}$ using CCR hash functions. The prover can then compute the final commitment value h using all $\mathsf{com}_i^{(\alpha)}$, $\alpha \in [\tau]$. In BACON.Open, computes the decommitment with respect to the challenge set I, the decommitment consists of the commitments of change set I and authentication path of hidden nodes in I. The size of the opened nodes set $|S|$, is then compared against a threshold T_{open}. If $|S| \leq \mathsf{T}_{open}$, the new decommitment is produced, else a new challenge set is requested and the prover incurs a proof-of-work to do so. BACON.Verify recomputes the cGGM leaves given the final commitment value, decommitment and challenge set. Finally BACON.Verify compares the computed commitment value with the given commitment value and decides whether the commitment is valid or not.

- $\mathsf{BACON.Setup}(1^\lambda, L) \to \mathsf{pp}$: Generate public parameters pp.
- $\mathsf{BACON.Commit}(\mathsf{sd}, \mathsf{iv}) \to (\mathsf{com}, \mathsf{decom}, \mathbf{m} = (m_1^{(\alpha)}, ..., m_{N_\alpha}^{(\alpha)})_{\alpha\in[\tau]})$:
  1. Sample $k \leftarrow \{0,1\}^\lambda$. $k_0^1||k_1^1 = \mathsf{PRG}(\mathsf{sd}, \mathsf{iv}; 2\lambda)$, where $\Delta = k_0^1 \oplus k_1^1$
  2. For $i \in [2, d]$, $j \in \left[0, 2^{i-1}\right)$, compute $k_{2j}^i = \mathsf{H}_{\mathsf{CCR}}(k_j^{i-1})$, $k_{2j+1}^i = k_{2j}^i \oplus k_j^{i-1}$;
  3. Deinterleave the leaves:
  $$\left\{\mathsf{sd}_1^{(\alpha)}, \ldots, \mathsf{sd}_{N_\alpha}^{(\alpha)}\right\}_{\alpha\in[\tau]} \overset{\pi}{\leftarrow} \left\{k_0^d, \cdots, k_{2^d-1}^d\right\}.$$
  4. Compute $m_i^{(\alpha)} \leftarrow \mathsf{H}_{\mathsf{CCR}}(\mathsf{sd}_i^{(\alpha)})$ and $\mathsf{com}_i^{(\alpha)} \leftarrow \mathsf{H}_{\mathsf{CCR}}(\mathsf{sd}_i^{(\alpha)} \oplus 1)||\mathsf{H}_{\mathsf{CCR}}(\mathsf{sd}_i^{(\alpha)} \oplus 2)$, for $\alpha \in [\tau]$ and $i \in [N_\alpha]$.
  5. Compute $h^{(\alpha)} \leftarrow \mathsf{H}\left(\mathsf{iv}, \mathsf{com}_1^{(\alpha)}, \ldots, \mathsf{com}_{N_\alpha}^{(\alpha)}\right)$ for $\alpha \in [\tau]$ and compute $h \leftarrow \mathsf{H}\left(\mathsf{iv}, h^{(1)}, \ldots, h^{(\tau)}\right)$.
  6. Output $(\mathsf{com} = h, \mathsf{decom} = k, \mathbf{m} = (m_1^{(\alpha)}, ..., m_{N_\alpha}^{(\alpha)})_{\alpha\in[\tau]})$.
- $\mathsf{BACON.Open}(\mathsf{decom} = k, I = (i^{(1)}, ..., i^{(\tau)})) \to \mathsf{decom}_I \vee \perp$:
  1. Recompute k_j^i for $i \in [2, d], j \in [0, 2^{i-1})$ from k as in Commit.
  2. Let $S = \left\{k_0^d, \ldots, k_{2^d-1}^d\right\}$. For each $\alpha \in [\tau]$, remove the associated path of nodes $k_{\pi^{-1}\left(\alpha, i^{(\alpha)}\right)}$ from S.
  3. For i from $d-1$ to 0, $j \in [0, 2^i - 1]$, if $k_{2j}^i \in S$ and $k_{2j+1}^i \in S$, replace both with k_j^{i-1}.
  4. If $|S| \le \mathsf{T}_{open}$, output $\mathsf{decom}_I = ((\mathsf{com}_{i^{(\alpha)}}^{(\alpha)})_{\alpha\in[\tau]}, S)$, otherwise output $\perp$.
- $\mathsf{BACON.Verify}(\mathsf{com}, \mathsf{decom}_I, I)) \to \left(\left(m_j^{(\alpha)}\right)_{j\in[N_\alpha]\setminus\left\{i^{(\alpha)}\right\}}\right)_{\alpha\in[\tau]} \vee \perp$:
  1. Recompute $\mathsf{sd}_i^{(\alpha)}$ from decom_I, for each $\alpha \in [\tau]$ and $i \neq i^{(\alpha)}$ using the available keys in S, and compute $m_i^{(\alpha)} \leftarrow \mathsf{H}_{\mathsf{CCR}}(\mathsf{sd}_i^{(\alpha)})$ and $\mathsf{com}_i^{(\alpha)} \leftarrow \mathsf{H}_{\mathsf{CCR}}(\mathsf{sd}_i^{(\alpha)} \oplus 1)||\mathsf{H}_{\mathsf{CCR}}(\mathsf{sd}_i^{(\alpha)} \oplus 2)$.
  2. Compute $h^{(\alpha)} = \mathsf{H}\left(\mathsf{iv}, \mathsf{com}_1^{(\alpha)}, \ldots, \mathsf{com}_{N_\alpha}^{(\alpha)}\right)$ for each $\alpha \in [\tau]$.
  3. If $h \neq \mathsf{H}\left(\mathsf{iv}, h^{(1)}, \ldots, h^{(\tau)}\right)$ output $\perp$. Otherwise, output $\left(\left(m_i^{(\alpha)}\right)_{i\in[N_\alpha]\setminus\{i^{(\alpha)}\}}\right)_{\alpha\in[\tau]}$.

Figure 2: The proposed BAVC with aborts scheme, BACON.

4 Security analysis

In this section, We prove that our scheme satisfies extractable binding. Intuitively, we leverage the extractable-binding security property of the $\mathsf{H}_{\mathsf{CCR}}$ proved in [20] to prove that an adversary is probabilistically bounded in their ability to win in the extractable-binding game in our scheme. For the hiding, the proof relies on the CCR property of the $\mathsf{H}_{\mathsf{CCR}}$ proved in [20]. Further, we model the $\mathsf{H}_{\mathsf{CCR}}$ as an ideal cipher oracle. Before presenting our security proof, let's recall the definitions of the CCR and AES-based CCR hash function are given as follows:

4.1 Circular Correlation Robustness

Circular Correlation Robustness (CCR) is a security notion first introduced for circuit garbling with a free XOR optimization [38]. For this paper, we are only concerned with a hash function that meets the CCR security requirement. Informally, a function $f_R(x, b) = H(x \oplus R) \oplus b \cdot R$ is CCR if it is pseudorandom provided that x is never repeated. CCR hash functions can be constructed from a fixed-key block cipher such as AES and a *linear orthomorphism* σ [20] [2]. Formally:

Definition 2 *(Circular Correlation Robustness [32]). Let $\mathsf{H}^E\colon \{0,1\}^\lambda \to \{0,1\}^\lambda$, be a function defined upon an ideal cipher E. For $\Gamma \in \{0,1\}^\lambda$, define $\mathcal{O}_\Gamma^{\mathsf{CCR}}(x, b) = \mathsf{H}^E(x \oplus \Gamma) \oplus b \cdot \Gamma$. We prohibit the distinguisher from querying the same x with both $b = 0$ and $b = 1$ to avoid the trivial attack. For a distinguisher $\mathcal{D}$, we define the following advantage* $\mathrm{Adv}_{\mathsf{H}}^{\mathsf{CCR}}$

$$\left| \Pr_{\Gamma \leftarrow \{0,1\}^\lambda} \left[\mathcal{D}^{\mathcal{O}_\Gamma^{\mathsf{CCR}}(\cdot), E(\cdot,\cdot), E^{-1}(\cdot,\cdot)} \left(1^\lambda\right) = 1 \right] - \Pr_{f \leftarrow \mathcal{F}_{\lambda+1,\lambda}} \left[\mathcal{D}^{f(\cdot), E(\cdot,\cdot), E^{-1}(\cdot,\cdot)} \left(1^\lambda\right) = 1 \right] \right|$$

Where $\mathcal{F}_{\lambda+1,\lambda}$ denotes the set of all functions mapping $(\lambda+1)$-bit inputs to λ-bit outputs. H is (q_E, q_C, ϵ)-circular correlation robust if for all $\mathcal{D}$ making at most q_E queries to E and E^{-1} and at most q_C queries to the oracle we have $\mathrm{Adv}_{\mathsf{H}}^{\mathsf{CCR}} \leq \epsilon$.

In this paper, we will use a CCR hash function constructed using AES explicated in Fig. 3 to construct a binary tree whose leaves become message and commitment outputs. Furthermore, we will make use of the extractable binding property of the CCR hash, proved in the ideal cipher model [20], to argue our proposed scheme BACON meets the security requirements.

[2] A mapping $\sigma\colon \mathbb{G} \to \mathbb{G}$ for an additive Abelian group $\mathbb{G}$ is a linear orthomorphism if (i): σ is a permutation, (ii) $\sigma(x+y) = \sigma(x) + \sigma(y)$ for any $x, y \in \mathbb{G}$, and (iii) $\sigma'(x) = \sigma(x) - x$ is also a permutation. [32] presents two efficient instantiations of σ (with well-defined efficient σ^{-1}, σ', and σ'^{-1}): (i) if $\mathbb{G}$ is a field, $\sigma(x) = c \cdot x$ for some $c \neq 0, 1 \in \mathbb{G}$, and (ii) if $\mathbb{G} = \{0,1\}^\lambda$, $\sigma(x) = \sigma(x_L || x_R) = (x_L \oplus x_R) || x_L$ where x_L and x_R are the left and right halves of x.

```
**AES-based CCR Hash Function** $\mathsf{H}_{\mathsf{CCR}}(1^\lambda, 1^n, r)$ :

• If $\lambda = 128$,
    1. Return $\mathsf{AES\text{-}128}([C_0]_{128}, \sigma(r)) \oplus \sigma(r)$.
• If $\lambda = 192$,
    1. Let $r_L \leftarrow \mathsf{left}_{128}(r)$ and $r_R \leftarrow \mathsf{right}_{64}(r)$.
    2. Return $\mathsf{AES}_{192}(r_R||[C_0]_{128}, \sigma(r_L)) \oplus \sigma(r_L)||$
       $\mathsf{left}_{64}(\mathsf{AES}_{192}(r_R||[C_1]_{128}, \sigma(r_L)) \oplus \sigma(r_L))$.
• If $\lambda = 256$,
    1. Let $r_L \leftarrow \mathsf{left}_{128}(r)$ and $r_R \leftarrow \mathsf{right}_{128}(r)$.
    2. Return $\mathsf{AES}_{256}(r_R||[C_0]_{128}, \sigma(r_L)) \oplus \sigma(r_L)||$
       $(\mathsf{AES}_{256}(r_R||[C_1]_{128}, \sigma(r_L)) \oplus \sigma(r_L))$.
```

Figure 3: AES-based CCR [20]. C_0 and C_1 are two distinct $128-$bit public constants, r is a $\lambda-$bit value, $[.]_{128}$ denotes the $128-$bit binary representation, σ is a linear orthomorphism on $128-$bit values, left_n and right_n respectively denotes taking the left and right n bits of the input.

4.2 Security model

A BAVC with aborts scheme is captured by *correctness*, *extractable-binding* and *hiding (real-or-random)*. These requirements are defined as follows:

Definition 3 *(Correctness with aborts). A BAVC with aborts scheme is correct if for all $I \subset [N_1] \times \cdots \times [N_\tau]$, the following outputs True*

$$\begin{aligned} \mathit{pp} &\leftarrow \mathit{Setup}(\lambda) \\ (\mathit{com}, \mathit{decom}, \boldsymbol{m}) &\leftarrow \mathit{Commit}(\lambda, \mathit{pp}) \\ \forall \mathit{decom}_I &\leftarrow \mathit{Open}(\mathit{decom}, I) \end{aligned}$$

output $\mathit{decom}_I = \perp \vee \mathit{Verify}(\mathit{com}, \mathit{decom}_I, I) = \boldsymbol{m}$ with all but a negligible probability, where $\boldsymbol{m} = \left(m_1^{(\alpha)}, \ldots, m_{N_\alpha}^{(\alpha)}\right)_{\alpha \in [\tau]}$.

Informally, a commitment scheme is extractable-binding if there exists an extractor Ext such that the extracted message equals to the opened message associated with a commitment.

Definition 4 *(Extractable-Binding). Let the vector commitment scheme constructed by two hash functions, $\mathit{H_{CCR}}$ and H, Ext be a PPT algorithm such that*

- *$\mathit{Ext}(Q_{\mathit{H}}, Q_{\mathit{H_{CCR}}}, \mathit{com}) \rightarrow ((m_j^{(\alpha)})_{j \in [N_\alpha]})_{\alpha \in [\tau]}$, i.e., given two sets of oracles Q_{H} and $Q_{\mathit{H_{CCR}}}$ of query-response pairs, and a commitment com, Ext outputs $m_j^{(\alpha)}$ or $m_j^{(\alpha)} = \perp$ if the committed value is invalid.*

For any τ, $N_\alpha = poly(\lambda)$, the extractable-binding game $\mathsf{EB}_{\mathcal{A}}^{\mathsf{BAVC}}$ for BAVC with $\mathcal{A}$ is defined as follows:

1. $\mathsf{pp} \rightarrow \mathsf{Setup}(1^\lambda)$.
2. $\mathsf{com} \leftarrow \mathcal{A}^{\mathsf{H}, \mathsf{H_{CCR}}}(\mathsf{pp})$.
3. $\bar{\mathbf{m}} \leftarrow \mathsf{Ext}(Q_{\mathsf{H}}, Q_{\mathsf{H_{CCR}}}, \mathsf{com})$, where $\bar{\mathbf{m}} = ((\bar{m_1}^{(\alpha)}, ..., \bar{m_{N_\alpha}}^{(\alpha)})_{\alpha \in [\tau]})$ is a message vector output by Ext.
4. $(\mathbf{m}, \mathsf{decom}_I, I) \leftarrow \mathcal{A}^{\mathsf{H}, \mathsf{H_{CCR}}}(\mathsf{open}, \mathsf{com})$, where $\mathbf{m} = ((m_j^{(\alpha)})_{j \in [N_\alpha] \setminus \{i_\alpha\}})_{\alpha \in [\tau]}$.
5. Output 1 (success) if: $\mathsf{Verify}(\mathsf{com}, \mathsf{decom}_I, I) = \mathbf{m}$, but $m_j^{(\alpha)} \neq \bar{m}_j^{(\alpha)}$ for some $\alpha \in [\tau]$, $j \in [N_\alpha] \setminus \{i_\alpha\}$, where i_α is the index of unopened nodes. Else output 0 (failure).

The advantage of $\mathcal{A}$ to win is denoted by $\mathsf{AdvEB}_{\mathcal{A}}^{\mathsf{BAVC}} = Pr[\mathsf{EB}_{\mathcal{A}}^{\mathsf{BAVC}} = 1]$. A BAVC with aborts scheme is *extractable* with respect to Ext if $\mathsf{AdvEB}_{\mathcal{A}}^{\mathsf{BAVC}}$ is negligible, i.e., $\mathsf{AdvEB}_{\mathcal{A}}^{\mathsf{BAVC}}(\lambda) \leq negl(\lambda)$.

Informally, a BAVC with aborts is hiding if values at unopened indices remain hidden even after opening a subset of the vector. Formally:

Definition 5 *(Hiding (real-or-random)). Let BAVC be an iv-based vector commitment scheme. The adaptive hiding game for BAVC with τ, $N_\alpha = poly(\lambda)$, parameter n and stateful $\mathcal{A}$ is defined as follows:*

1. $\mathsf{pp} \to \mathsf{Setup}(1^\lambda)$.
2. $\bar{b} \leftarrow \{0,1\}$, $\mathsf{sd} \leftarrow \{0,1\}^\lambda$, $\mathsf{iv} \leftarrow \{0,1\}^\lambda$.
3. $(\mathsf{com}, \mathsf{decom}, (\bar{m_1}^{(\alpha)}, ..., \bar{m_N}^{(\alpha)})_{\alpha\in[\tau]}) \leftarrow$ $\mathsf{Commit}^{\mathsf{H}}(\mathsf{iv})$
4. $I \leftarrow \mathcal{A}^{\mathsf{H}}(1^\lambda, \mathsf{pp}, \mathsf{com})$, *where* $I \in [N_1] \times ... \times [N_\tau]$.
5. $\mathsf{decom}_I \leftarrow \mathsf{Open}(\mathsf{decom}, I)$.
6. $m_j^{(\alpha)} \leftarrow \bar{m}_j^{(\alpha)}$ *for* $j \in [N_\alpha] \setminus \{i_\alpha\}, \alpha \in [\tau]$.
7. *Set* $m_{i_\alpha}^{(\alpha)} \to \left\{ \begin{array}{ll} \text{random from } \mathcal{M} & \text{if } \bar{b} = 0 \wedge \alpha \leq n \\ \bar{m}_{i_\alpha}^{(\alpha)} & \text{otherwise} \end{array} \right\}$.
8. $b \leftarrow \mathcal{A}\left(\left(m_j^{(\alpha)}\right)_{j\in[N_\alpha]}, \mathsf{decom}_I\right)$
9. *Output 1 (win) if* $\bar{b} = b$ *else 0 (lose).*

In the selective hiding experiment, $\mathcal{A}$ must choose I prior to receiving com.

The advantage $\mathsf{AdvAdpHide}_{\mathcal{A},i}^{\mathsf{BAVC}}$ (resp. $\mathsf{AdvSelHide}_{\mathcal{A},i}^{\mathsf{BAVC}}$) of an adversary $\mathcal{A}$ is defined by $Pr[\mathcal{A}$ wins and $n = i] - 1/2$ in the adaptive (resp. selective) hiding game. We say BAVC is adaptively (resp. selectively) hiding if every PPT adversary $\mathcal{A}$ has a negligible advantage of winning $\mathsf{AdvAdpHide}_{\mathcal{A},i}^{\mathsf{BAVC}}$ ($\mathsf{AdvSelHide}_{\mathcal{A},i}^{\mathsf{BAVC}}$).

4.3 Security proof of BACON

Following the security model in subsection 4.2, in this section, we give the security proof of BACON.

Theorem 1 (Correctness) *Our BACON is correct.*

Proof 1 *After the algorithms Setup, Commit and Open, com, decom_I and I can be obtained, where $\mathsf{com} = h$, $I = (i^{(1)}, ..., i^{(\tau)}))$ and $\boldsymbol{m} = (m_1^{(\alpha)}, ..., m_{N_\alpha}^{(\alpha)})_{\alpha\in[\tau]}$. In the proposed BACON scheme, there is an abort mechanism. So the probability of outputting $\mathsf{decom}_I = \perp$ is negligible. When $\mathsf{decom}_I = ((\mathsf{com}_{i^{(\alpha)}}^{(\alpha)})_{\alpha\in[\tau]}, S)$, then the verification goes as follows:*

- *Given decom_I, for each $\alpha \in [\tau]$ and $i \neq i^{(\alpha)}$, compute $\mathsf{sd}_i^{(\alpha)}$;*
- *Compute $m_i^{(\alpha)} \leftarrow \mathsf{H}_{\mathsf{CCR}}(\mathsf{sd}_i^{(\alpha)})$ and $\mathsf{com}_i^{(\alpha)} \leftarrow \mathsf{H}_{\mathsf{CCR}}(\mathsf{sd}_i^{(\alpha)} \oplus 1)||\mathsf{H}_{\mathsf{CCR}}(\mathsf{sd}_i^{(\alpha)} \oplus 2)$;*

- *Given* $\mathsf{com}_{i^{(\alpha)}}^{(\alpha)})_{\alpha\in[\tau]}$, *Compute* $h' = H\left(\mathsf{com}_1^{(\alpha)}, \ldots, \mathsf{com}_{N_\alpha}^{(\alpha)}\right)$. *If* $h' = h$, *output* $\left(\left(m_i^{(\alpha)}\right)_{i\in[N_\alpha]\setminus\{i^{(\alpha)}\}}\right)_{\alpha\in[\tau]} \neq \boldsymbol{m}$*; otherwise, output* $\perp$.

The probability of outputting $\mathbf{m}$ *is negligible. Therefore, the BACON scheme offers correctness.* □

Theorem 2 (Extractable binding) *Let* $\mathsf{H}_{\mathsf{CCR}}$ *be a* $(q_E, \varepsilon, \delta)$*-extractable binding CCR function in the ideal model, where* ε *and* δ *are the upper bound of probability that the adversary fails and succeeds in the extractable-binding experiment of* $\mathsf{H}_{\mathsf{CCR}}$*, respectively.* H_{RO} *is a random oracle for the hash function* H*. Therefore, as defined in Definition 4, our proposed cGGM-based BAVC with aborts scheme, BACON, satisfies* $\mathsf{AdvEB}_{\mathcal{A}}^{\mathsf{BAVC}} \leq \frac{1+q_H(q_H-1)/2}{2^{2\lambda}} + L\cdot(\delta+\varepsilon)$ *for any adversary* $\mathcal{A}$ *making* q_E *queries to the ideal cipher oracle and* q_H *queries to the random oracle.*

Proof 2 *Firstly, define the extractor as* $\mathrm{Ext}(c, \mathcal{Q}_E)$ *where* c *is a commitment and* $\mathcal{Q}_E$ *is the transcript of ideal cipher queries and responses. The extractor is supposed to output the pre-image of the commitment* c*. The extractor goes through the random oracle's transcript and looks for the pre-image of* com*. If none can be found or the pre-image is not unique, then it outputs* $\perp$*.*

Assuming that the pre-image $\mathsf{com}_1^{(\alpha)}, \ldots, \mathsf{com}_{N_\alpha}^{(\alpha)}$ *is unique then the extractor applies* $\mathrm{Ext}(\mathsf{com}_{N_i}, \mathcal{Q}_E)$*,* $N_i \in \{1, ..., N_\alpha\}$ *to get* m_i*. Finally, the extractor outputs* $\{m_i\}_{i\in[1,N_\alpha]}$*. We bound the probability* $\mathrm{AdvEB}_{\mathcal{A}}^{\mathrm{BAVC}}$*. Since* H_{RO} *is a random oracle, the probability of the extractor outputting* $\perp$ *due to finding a collision is bounded by* $\frac{1}{2^{2\lambda}} + \frac{2}{2^{2\lambda}} + \cdots + \frac{q_H-1}{2^{2\lambda}} = \frac{q_H(q_H-1)}{2\cdot 2^{2\lambda}}$*. The probability of not being able to find a pre-image is bounded by* $\frac{1}{2^{2\lambda}}$*. Therefore, the extraction of* com *succeeds except with probability* $\frac{1+q_H(q_H-1)/2}{2^{2\lambda}}$*. Furthermore, the event of extraction failure for* $i \notin I$ *would imply extraction failure for* $\mathrm{Ext}(\mathsf{com}_{N_i}, \mathcal{Q}_E)$*, which is bounded by* $\varepsilon + \delta$*. Therefore, by taking a union-bound on all the leaf nodes, we conclude that*

$$\mathrm{AdvEB}_{\mathcal{A}}^{\mathrm{BAVC}} \leq \frac{1 + q_H(q_H-1)/2}{2^{2\lambda}} + L\cdot(\delta+\varepsilon)$$

□

Intuitively, to prove real-or-random hiding we again leverage the properties of the $\mathsf{H}_{\mathsf{CCR}}$ to show that an adversary will only have a negligible advantage when making queries to an ideal cipher oracle to guess the leaves of the whole tree.

Theorem 3 (Hiding real-or-random) *The BACON offers adaptive hiding. Let* $\mathsf{H}_{\mathsf{CCR}}$ *be* (t, q, p, ϵ)*-circular correlation robust and let* H *be a random oracle. Then for any adversary* $\mathcal{A}$ *in the adaptive hiding game making* $|Q|$ *queries to* H*, we have*

$$\mathsf{AdvAdpHide}_{\mathcal{A},i}^{\mathsf{BAVC}} \leq \frac{|Q|}{2^{2\lambda}} + \mathsf{Adv}_{\mathcal{B}}^{PRG}(\lambda) + \epsilon$$

Proof 3 *Let $\mathcal{B}$ be a distinguisher which can access the CCR hash oracle $\mathcal{O}_{\mathsf{CCR}}$, where $\mathcal{O}_{\mathsf{CCR}}(x, b) = \mathsf{H}_{\mathsf{CCR}}(x \oplus \Delta) \oplus b \cdot \Delta$. Here $\mathcal{B}$ simulates the random oracle H. $\mathcal{B}$'s goal is to output a bit that distinguishes between the real world and ideal world. Then $\mathcal{B}$, who is regarded as an adversary against the CCR game, works as follows:*

1. *Sample $\bar{b}$, $\mathsf{sd}_\alpha \leftarrow \{0,1\}^\lambda$, $\mathsf{iv} \leftarrow \{0,1\}^{2\lambda}$.*
2. *Generate com, decom_I and $\boldsymbol{m}$ as follows:*
   - *Sample decom, $\{\mathsf{com}_i^{(\alpha)}\}$ and $\{m_i^{(\alpha)}\}$, $\alpha \in [\tau]$, $i \in I$, uniformly at random.*
   - *Compute $\{m_i\}_{i \notin I}$ and $\{\mathsf{com}_i\}_{i \notin I}$ using the Verify algorithm.*
   - *Compute $\mathsf{com} = \mathsf{H}(\mathsf{com}_1, ..., \mathsf{com}_{N_\alpha})$.*
3. *Receive the challenge $I \leftarrow \mathcal{A}^{\mathsf{H}}(1^\lambda, \mathsf{pp}, \mathsf{com})$, where I is the set of opened nodes and $I \in [N_1] \times ... \times [N_\tau]$.*
4. *$i^{(\alpha)}$ is the index of the unopened node in the subtree α, $i^{(\alpha)} = \alpha_1...\alpha_d$. If $\alpha_i = 0$, we need to compute the right child, which is an element of the authentication path.*
5. *We assume that $\alpha_1 = 1$, which means the authentication path starts from the right subtree. Then the simulation process of Commit is as follows:*
   (a) *Samples $k^1_{\bar{\alpha}_1} \leftarrow \{0,1\}^\lambda$ and sets $s_1^{i^{(\alpha)}} = k^1_{\bar{\alpha}_1}$ and add $s_1^{i^{(\alpha)}}$ to S if $s_1^{i^{(\alpha)}}$ is not in S.*
   (b) *For $i \in [2, d]$, let $j^* = \alpha_1 || ... || \alpha_{i-1}$ computes $k^i_{j^* + \bar{\alpha}_i} = \mathcal{O}^{\mathsf{CCR}}(s_{i-1}^{i^{(\alpha)}}, \bar{\alpha}_i) \oplus \bar{\alpha}_i s_{i-1}^{i^{(\alpha)}}$. For all $j \in \left[0, 2^{i-1}\right)$, $j \neq j^*$, computes $k^i_{2j} = \mathsf{H}_{\mathsf{CCR}}(k^i_j)$, $k^i_{2j+1} = \mathsf{H}_{\mathsf{CCR}}(k^i_j) \oplus k^i_j$. Finally, updates $s_i^{i^{(\alpha)}} = s_{i-1}^{i^{(\alpha)}} \oplus k^i_{j^* + \bar{\alpha}_i}$ and add $s_i^{i^{(\alpha)}}$ to S if $s_i^{i^{(\alpha)}}$ is not in S.*
   (c) *For $i^{(\alpha)} \in I$, computes $m_i^{(\alpha)} \leftarrow \mathcal{O}_{\mathsf{CCR}}(s_d^{i^{(\alpha)}}, 0)$ and $\mathsf{com}_i^{(\alpha)} \leftarrow \mathcal{O}_{\mathsf{CCR}}(s_d^{i^{(\alpha)}} \oplus 1, 0) || \mathcal{O}_{\mathsf{CCR}}(s_d^{i^{(\alpha)}} \oplus 2, 0)$. Otherwise, for $i \in [0, N_\alpha) \subseteq [0, 2^d)$ but $i \notin I$, computes $m_i \leftarrow \mathsf{H}_{\mathsf{CCR}}(k^i_d)$ and $\mathsf{com}_i \leftarrow \mathsf{H}_{\mathsf{CCR}}(k^i_d \oplus 1) || \mathsf{H}_{\mathsf{CCR}}(k^i_d \oplus 2)$.*
   (d) *Compute $h^{(\alpha)}$ by simulating the oracle H by inputting $\left(\mathsf{iv}, \mathsf{com}_1^{(\alpha)}, \ldots, \mathsf{com}_{N_\alpha}^{(\alpha)}\right)$ for $\alpha \in [\tau]$.*
6. *Repeat steps 1, 2, 3, 4 and 5 for τ times to recompute the big cGGM trees and get all messages and commitments for all hidden nodes.*
7. *Compute h by simulating the oracle H via inputting $\left(\mathsf{iv}, h^{(1)}, \ldots, h^{(\tau)}\right)$.*
8. *Output $(\mathsf{com} = h$, $\mathsf{decom}_I = S$, $\boldsymbol{m} = (m_1^{(\alpha)}, ..., m_{N_\alpha}^{(\alpha)})_{\alpha \in [\tau]})$.*
9. *Assign decom_I and the adversary $\mathcal{A}$ outputs b.*
10. *Output 1 if $\bar{b} = b$, else 0.*

$\mathcal{B}$'s advantage. *b is the CCR oracle $\mathcal{O}_{\mathsf{CCR}}$'s hidden bit. If $b = 1$, then $\mathcal{B}$ is in the real world. If $b = 0$, $\mathcal{B}$ is in the real world is in the ideal world. The advantage that $\mathcal{B}$ distinguishes the CCR hash game is*

$$\begin{aligned}\mathsf{Adv}_{\mathcal{B}}^{\mathsf{CCR}}(\lambda) = \big|&\Pr[1 \leftarrow \mathcal{B}^{\mathcal{O}_{\mathsf{CCR}}}(1^{\lambda}) \mid b = 1] \\ &- \Pr[1 \leftarrow \mathcal{B}^{\mathcal{O}_{\mathsf{CCR}}}(1^{\lambda}) \mid b = 0]\big|\end{aligned}$$

Except the CCR oracle, there are two differences between the real adaptive game and the ideal world simulated by $\mathcal{B}$ for $\mathcal{A}$:

1. *$\mathcal{B}$ aborts if the simulation of $\mathcal{O}_{\mathsf{H}}$ fails;*
2. *k_0^1 and k_1^1 are chosen uniformly random, instead of computed by PRG.*

The first difference during the simulation of $\mathcal{O}_{\mathsf{H}}$ can be noticed if $\mathcal{A}$ make queries to $\mathcal{O}_{\mathsf{H}}$ using the same iv value. Then we can get the probability that $\mathcal{A}$ that notices this difference as follows:

$$\Pr[\mathcal{A} \text{ notices } \mathcal{O}_{\mathsf{H}}] \leq \frac{|Q|}{2^{2\lambda}} \tag{1}$$

The second difference can be reduced to distinguishing the output of PRG from a uniformly random output. The probability that $\mathcal{A}$ behaves differently equals to the case that $\mathcal{B}$ plays the PRG game using $\mathcal{A}$ as a subroutine. Then we can get

$$\Pr[\mathcal{A} \text{ notices } PRG] \leq \mathsf{Adv}_{\mathcal{B}}^{PRG}(\lambda) \tag{2}$$

If $b = 1$, then after changes 1 and 2, $\mathcal{A}$'s view is distributed identically to the real hiding game G_0 if $\bar{b} = 1$ and $\mathcal{A}$'s view is distributed identically to the ideal game G_1 if $\bar{b} = 0$. Therefore, we can get

$$\Pr[1 \leftarrow \mathcal{B}^{\mathcal{O}_{CCR}}(1^{\lambda}) \mid b = 1] = \Pr[\mathcal{A} \text{ wins changed game}]$$

If $b = 0$, the view of $\mathcal{A}$ is sampled uniformly at random. The we have

$$\Pr[1 \leftarrow \mathcal{B}^{\mathcal{O}_{CCR}}(1^{\lambda}) \mid b = 1] = 1/2$$

Finally, putting all cases together, we can get

$$AdvAdpHide_{\mathcal{A},i}^{\mathsf{BAVC}}(\lambda) \leq \frac{|Q|}{2^{2\lambda}} + \mathsf{Adv}_{\mathcal{B}}^{PRG}(\lambda) + Adv_{\mathcal{B}}^{\mathsf{H}_{\mathsf{CCR}}}(\lambda) \tag{3}$$

Because $\mathsf{H}_{\mathsf{CCR}}$ is a (t, q, p, ϵ)-circular correlation robust hash function, we can get the

$$\mathsf{AdvAdpHide}_{\mathcal{A},i}^{\mathsf{BAVC}} \leq \frac{|Q|}{2^{2\lambda}} + \mathsf{Adv}_{\mathcal{B}}^{PRG}(\lambda) + \epsilon$$

5 Applications

It is well known that AVCs are important components in FAEST and other MPCitH style NIZK proof systems. In this section, we discuss how to apply our BACON to FAEST version 2 to improve the efficiency.

In a MPCitH style signature, to create a signature, given a pair of public values x, y, the prover needs to prove knowing a witness w such that $f(x, w) = y$, where f is a one-way function. To achieve this, the prover simulates an MPC protocol between N simulated parties that realizes $f(x, w)$, where w is secret-shared as input shares to N parties. The prover then commits to the views and internal states of each party. After receiving the commitments, the verifier generates a challenge index and sends it to the prover. The prover opens a subset of these commitments to the verifier. Finally, the verifier checks them and decides whether to accept or reject. This process can be made non-interactive using the Fiat-Shamir transformation [29]. MPCitH is highly efficient for small to medium-sized circuits.

To handle large circuits, recent literature in interactive ZK protocols [11,13, 18, 24–26, 30, 39–42, 44–46] makes use of sVOLE, which are called VOLE-ZK. A VOLE is a two-party correlation of length n between a prover and a verifier defined by a random global key $\Delta \in \mathbb{F}_{2^k}$, a set of random bits $u_i \in \mathbb{F}_2$, a random VOLE tag $v_i \in \mathbb{F}_{2^k}$ and VOLE keys $w_i \in \mathbb{F}_{2^k}$ such that $w_i = u_i \cdot \Delta - v_i$, $i = 0, ..., n-1$. The prover obtains u_i, v_i while the verifier obtains Δ, w_i. The correlations allow the prover to commit all u_i as linear homomorphic commitments, which can lead to efficient proof systems. Compared with MPCitH style ZK systems, VOLE-ZK is featured with a shorter proof size and faster running time. However, this is in the designated verifier setting. In 2023, Baum *et al.* [9] transform VOLE to VOLEitH to achieve a postponed VOLE functionality and make use of the Fiat-Shamir transformation to propose a new NIZK framework. FAEST version 2 [10] integrated the large GGM based BAVC with aborts scheme [8] into FAEST [9] to improve the efficiency and achieve shorter signature size, which is a second-round candidate for additional NIST post-quantum digital signatures.

Optimizing FAEST version 2. As described, the recent VOLE-based interactive ZK protocols and non-interactive zero-knowledge (NIZK) proofs based on MPCitH or VOLEitH extensively make use of commitment schemes, which adopt a GGM-style tree to generate and open commitments. Here, we take the FAEST version 2 [10] signature scheme as an example of applications of our proposed large cGGM-based BAVC with aborts scheme, BACON.

In FAEST [9], to obtain the desired soundness, it is required to run τ instances of VOLEitH. Using a BAVC with aborts in FAEST version 2 [10], one can remove the need for $\tau > 1$ and take the leaves generated from one large GGM tree as inputs for the VOLEitH protocol. The verifier will then provide the random challenge value Δ, ask to reveal all but one tree leaf in each VOLE instance. FAEST version 2 [10] integrated the large GGM-based BAVC with aborts scheme into FAEST to get better efficiency and shorter signature size.

To enable BAVC with aborts for FAEST, the FAEST version 2 [10] signing algorithm is changed as described in [8]. FAEST version 2 retains the same VOLEitH approach but adds rejection sampling. To highlight the slight difference, consider FAEST signing algorithm when processing with the challenge vector detailed in Algorithm 1.

Algorithm 1 FAEST Signing

```
...
$chall_3 \leftarrow H_2^3(chall_2||\tilde{a}||\tilde{b};\lambda)$
$I \leftarrow DecodeChallenge(chall_3)$
$decom_I \leftarrow \mathbf{BACON.Open}(I)$
$\sigma \leftarrow \sigma||decom_I$
return $\sigma$
```

In Algorithm 1, the prover opens the challenge vector to create the decommitment to send to the verifier. This process is modified in FAEST version 2 as outlined in Algorithm 2.

Algorithm 2 FAEST version 2 Signing

```
...
Let $ctr \leftarrow 0$
Retry:
$chall_3 \leftarrow H_2^3(chall_2||\tilde{a}_0||\tilde{a}_1||\tilde{a}_2||ctr;\lambda)$
$I \leftarrow DecodeChallenge(chall_3[0:\lambda-w-1])$
$decom_I \leftarrow \mathbf{BACON.Open}(I)$
if $decom_I = \perp$ or $chal_3[\lambda-w:\lambda] \neq 0^w$ then $ctr \leftarrow ctr+1$ and go to Retry
$\sigma \leftarrow \sigma||decom_I||ctr$
return $\sigma$
```

The procedure above simply adds a counter *ctr* to enable rejection sampling. Every time the challenge index leads to a decommitment that is larger than the threshold, a new challenge index is generated, and the counter is incremented after the prover performs a proof of work.

6 Comparisons

There are some works on the GGM/cGGM based punturable pseudorandom function (PPRF) [17, 34] or AVC schemes [8, 37]. Among these schemes, we compare [8] with our BACON, because we both adopt a large tree structure and add the abort mechanism into the BAVC schemes.

6.1 Theoretical comparisons

As described throughout the paper, BACON utilizes one big cGGM tree to instantiate a BAVC with aborts.

Recall that we have discussed three basic GGM trees: a GGM tree [31], a cGGM tree [33], and a pcGGM tree [33]. Comparisons among these three trees are shown in Table 1. Given a tree with d layers, a GGM construction requires $(2^{d-1}-1)$ PRG calls and therefore $(2^{d}-2)$ hash calls, a cGGM tree requires $(2^{d-1}-1)$ $\mathsf{H}_{\mathsf{CCR}}$ hash calls, and a pcGGM tree requires $(2^{d-1}+2^{d-2}-1)$ $\mathsf{H}_{\mathsf{CCR}}$ hash calls. Based on the property of geometric progression, we can have $2^{d}-2=2^{0}+...+2^{d-2}+2^{d-1}-1>2^{d-2}+2^{d-1}-1$. Then one can get the efficiency result of these three tree constructions in descending order of cost as GGM tree [31]>pcGGM tree [33]>cGGM tree [33].

As claimed in [33], with the same number of layers, a cGGM tree can reduce 2x computation and a pcGGM tree can reduce 1.33x computation when they are used to construct correlated oblivious transfer (COT) and subfield VOLE (sVOLE) compared to a GGM tree. The difference between a cGGM tree and a pcGGM tree is that a pcGGM tree can be used to construct sVOLE [16], the security of which relies on the pseudorandomness of tree nodes. There is no BAVC scheme based on a large pcGGM, the possible reason is the efficiency is worse than a large cGGM. In our paper, we propose a big cGGM tree, which consists of multiple small cGGM subtrees.

We now analyze different tree structures for BAVC schemes. Existing BAVC schemes cover two instantiations, one big GGM tree and multiple cGGM trees. As mentioned in the Introduction of this paper, using one big tree instead of multiple small trees can make opening all but τ leaves of the former more efficient than opening all but one leaf in each of the small trees in parallel because some authentication paths in the big tree will merge, which can reduce the commitment size. We focus on the comparison between our proposed one big cGGM tree based BAVC scheme and state-of-the-art big tree structure, i.e., one big GGM tree [8] based BAVC scheme. We assume that there are τ small trees and each small tree i has N_i, $i \in [\tau]$ leaf nodes. The results are shown in Table 1. Our big cGGM tree needs fewer hash calls than one big GGM tree [8].

Moreover, adding the number of hash calls used in generating the commitments from the leaf layer, we list the results in Table 2. *Note that we make use of CCR hash functions for generating internal nodes while generating commitments from leaf layers, we utilize the H same as in [8], which can be modeled as random oracle. During implementation, the running times of CCR hash and H are different. Here, we just add them together to show that our BACON has improvement compared with [8].* We did not include the complexity comparison of $\mathsf{BAVC.Open}$ and $\mathsf{BAVC.Verify}$ because the number of opened nodes and the threshold T_{open} both depend on concrete implementations.

Discussion on future implementations. In the beginning, we implemented our BACON based on FAEST version 1, however, NIST released the FAEST version 2. Before implementing our BACON into FAEST version 2, the FAEST team claimed that they replaced the hash function calls and revisited the evaluation of the AES block cipher to achieve improvements on both security and practical efficiency [7], which was just accepted by CRYPTO 2025. [7] will be publicly accessible soon. We will integrate our BACON into [7] in the full version later.

Table 1: Comparisons among tree constructions

Type	**Tree construction**	**Number of CCR hash calls**
Small tree	GGM tree [31]	$(2^d - 2)$
	pcGGM tree [33]	$(2^{d-1} + 2^{d-2} - 1)$
	cGGM tree [33]	$(2^{d-1} - 1)$
Big tree	GGM tree [8]	$2^d - 2$
	Our cGGM tree	$2^{d-1} - 1$

For big tree constructions, we assume there are τ small trees and one small tree i, $i \in [\tau]$ has N_i leaf nodes, then the total number of layers in a big cGGM tree $d = \log_2 (\sum_{i=1}^{\tau} N_i)$.

Table 2: Comparisons between two large tree-based vector commitment schemes

Scheme	**Number of function calls**
[8]	$(2^d - 2)\mathsf{H}_{\mathsf{CCR}} + 3 * 2^d\mathsf{H}$
Our BACON	$(2^{d-1} - 1)\mathsf{H}_{\mathsf{CCR}} + 3 * 2^d\mathsf{H}$

7 Conclusions

BAVC allows the execution of multiple AVCs simultaneously. Further, adding an abort mechanism to BAVC allows further parameterization to reduce commitment sizes. This paper presents BACON based on a big cGGM tree, which is more efficient than the state-of-the-art [8] based on one large GGM tree.

We prove the security of BACON in the random oracle model and ideal cipher model. We have implemented our scheme based on FAEST version 1. Before updating this implementation to a new version based on FAEST version 2, we noticed that an improved FAEST scheme [7], called FAESTer, will soon be publicly available. Therefore, we will release our implementation based on the state-of-the-art FAEST scheme in the full version later.

References

[1] Najwa Aaraj, Slim Bettaieb, Alessandro Bidoux, Victor Dyseryn, Andre Esser, Philippe Gaborit, Mukul Kulkarni, Victor Mateu, Marco Palumbi, Lucas Perin, and Jean-Pierre Thilich. PERK, 2023. `https://csrc.nist.gov/Projects/pqc-dig-sig/round\-1-additional-signatures`.

[2] Gora Adj, Luis Rivera-Zamarripa, Javier verbel, Emanuele Bellini, Stefano Barbero, Andre Esser, Carlo Sanna, and Floyd Zweydinger.

MiRitH-MinRank in the Head, 2023. `https://csrc.nist.gov/Projects/pqc-dig-sig/\round-1-additional-signatures`.

[3] Carlos Aguilar-Melchor, Thibauld Feneuil, Nicolas Gama, Shay Gueron, James Howe, David Joseph, Antoine Joux, Edoardo Persichetti, Tovohery H Randrianarisoa, Matthieu Rivain, et al. SDitH—syndrome decoding in the Head. *Submission to the NIST Post-Quantum Standardization project*, 2023.

[4] Carlos Aguilar-Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. The return of the sdith. In *EUROCRYPT*, pages 564–596, 2023.

[5] Nicolas Aragon, Magali Bardet, Loic Bidoux, Jusus-Javier Chi-Domingues, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, Mattieu Rivain, and Jean-Pierre Tillich. MIRA, 2023. `https://csrc.nist.gov/Projects/pqc-dig-sig/\round-1-additional-signatures`.

[6] Nicolas Aragon, Bardet Magali, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Matthieu Rivain, Jean-Pierre Tillich, and Adrien Vinçotte. RYDE, 2023. `https://csrc.nist.gov/Projects/pqc-dig-sig/\round-1-additional-signatures`.

[7] Carsten Baum, Ward Beullens, Lennart Braun, Cyprien De Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, et al. Shorter, Tighter, FAESTer: Optimizations and improved (QROM) analysis for VOLE-in-the-Head signatures. In *CRYPTO*, 2025.

[8] Carsten Baum, Ward Beullens, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. One tree to rule them all: Optimizing GGM trees and OWFs for post-quantum signatures. In *ASIACRYPT*, pages 463–493, 2025.

[9] Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head. In *CRYPTO*, pages 581–615, 2023.

[10] Carsten Baum, Lennart Braun, Cyprien Delphech de Saint Guilheme, Michael Kloop, Christian Majenz, Shibam Mukherjee, Sebastian Ramacher, Christian Rechberger, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. FAEST: Algorithm specifications version 2.0, 2025. `https://csrc.nist.gov/csrc/media/Projects/pqc-\dig-sig/documents/round-2/spec-files/faest-spec\-round2-web.pdf`.

[11] Carsten Baum, Lennart Braun, Alexander Munch-Hansen, and Peter Scholl. Moz $\mathbb{Z}_{2^k}$ arella: efficient vector-ole and zero-knowledge proofs over $\mathbb{Z}_{2^k}$. In *CRYPTO*, pages 329–358, 2022.

[12] Carsten Baum, Cyprien Delpech de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. Banquet: short and fast signatures from AES. In *PKC*, pages 266–297, 2021.

[13] Carsten Baum, Alex J Malozemoff, Marc B Rosen, and Peter Scholl. Mac‘n‘cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In *CRYPTO*, pages 92–122, 2021.

[14] Luk Bettale, Delaram Kahrobaei, Ludovic Perret, and Javier Verbel. Biscuit, 2023. `https://csrc.nist.gov/Projects/pqc-dig-sig/\round-1-additional-signatures`.

[15] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In *ACM CCS*, pages 896–912, 2018.

[16] Elette Boyle, Geoffroy Couteau, Niv Gilboa, Yuval Ishai, Lisa Kohl, Peter Rindal, and Peter Scholl. Efficient two-round ot extension and silent non-interactive secure computation. In *ACM CCS*, pages 291–308, 2019.

[17] Dung Bui, Eliana Carozza, Geoffroy Couteau, Dahmun Goudarzi, and Antoine Joux. Faster signatures from MPC-in-the-head. *ASIACRYPT*, 2024.

[18] Dung Bui, Haotian Chu, Geoffroy Couteau, Xiao Wang, Chenkai Weng, Kang Yang, and Yu Yu. An efficient ZK compiler from SIMD circuits to general circuits. Cryptology ePrint Archive, Paper 2023/1610, 2023.

[19] Dung Bui, Kelong Cong, and Cyprien Delpech de Saint Guilhem. Faster VOLEith signatures from all-but-one vector commitment and half-tree. In *Australasian Conference on Information Security and Privacy*, pages 205–223, 2025.

[20] Hongrui Cui, Chun Guo, Xiao Wang, Chenkai Weng, Kang Yang, and Yu Yu. AES-based CCR hash with high security and its application to zero-knowledge proofs. Cryptology ePrint Archive, Paper 2024/1271, 2024.

[21] Cyprien Delpech de Saint Guilhem, Lauren De Meyer, Emmanuela Orsini, and Nigel P Smart. BBQ: using AES in picnic signatures. In *International Conference on Selected Areas in Cryptography*, pages 669–692, 2019.

[22] Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. Limbo: efficient zero-knowledge MPCitH-based arguments. In *ACM CCS*, pages 3022–3036, 2021.

[23] Itai Dinur and Niv Nadler. Multi-target attacks on the picnic signature scheme and related protocols. In *EUROCRYPT*, pages 699–727, 2019.

[24] Samuel Dittmer, Karim Eldefrawy, Stéphane Graham-Lengrand, Steve Lu, Rafail Ostrovsky, and Vitor Pereira. Boosting the performance of high-assurance cryptography: Parallel execution and optimizing memory access in formally-verified line-point zero-knowledge. In *ACM CCS*, pages 2098–2112, 2023.

[25] Samuel Dittmer, Yuval Ishai, Steve Lu, and Rafail Ostrovsky. Improving line-point zero knowledge: two multiplications for the price of one. In *ACM CCS*, pages 829–841, 2022.

[26] Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-point zero knowledge and its applications. Cryptology ePrint Archive, Paper 2020/1446, 2020.

[27] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In *CRYPTO*, pages 541–572, 2022.

[28] Thibuld Feneuil and Matthieu Rivain. MQOM, 2023. `https://csrc.nist.gov/Projects/pqc-dig-sig/\round-1-additional-signatures`.

[29] Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO*, pages 186–194, 1986.

[30] Nicholas Franzese, Jonathan Katz, Steve Lu, Rafail Ostrovsky, Xiao Wang, and Chenkai Weng. Constant-overhead zero-knowledge for RAM programs. In *ACM CCS*, pages 178–191, 2021.

[31] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions. *Journal of the ACM (JACM)*, 33(4):792–807, 1986.

[32] Chun Guo, Jonathan Katz, Xiao Wang, and Yu Yu. Efficient and secure multiparty computation from fixed-key block ciphers. In *IEEE Symposium on Security and Privacy (S&P)*, pages 825–841, 2020.

[33] Xiaojie Guo, Kang Yang, Xiao Wang, Wenhao Zhang, Xiang Xie, Jiang Zhang, and Zheli Liu. Half-tree: Halving the cost of tree expansion in COT and DPF. In *EUROCRYPT*, pages 330–362, 2023.

[34] Janik Huth and Antoine Joux. MPC in the head using the subfield bilinear collision problem. In *CRYPTO*, pages 39–70, 2024.

[35] Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM Journal on Computing*, 39(3):1121–1152, 2009.

[36] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *ACM CCS*, pages 525–537, 2018.

[37] Seongkwang Kim, Byeonghak Lee, and Mincheol Son. Relaxed vector commitment for shorter signatures. *IACR Preprint*, 2024.

[38] Vladimir Kolesnikov and Thomas Schneider. Improved garbled circuit: Free XOR gates and applications. In *Automata, Languages and Programming: 35th International Colloquium, ICALP*, pages 486–498, 2008.

[39] Fuchun Lin, Chaoping Xing, and Yizhou Yao. More efficient zero-knowledge protocols over $\mathbb{Z}_{2^k}$ via galois rings. In *CRYPTO*, pages 424–457, 2024.

[40] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *IEEE Symposium on Security and Privacy (S&P)*, pages 1074–1091, 2021.

[41] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. Mystique: Efficient conversions for zero-knowledge proofs with applications to machine learning. In *USENIX Security*, pages 501–518, 2021.

[42] Chenkai Weng, Kang Yang, Zhaomin Yang, Xiang Xie, and Xiao Wang. AntMan: Interactive zero-knowledge proofs with sublinear communication. In *ACM CCS*, pages 2901–2914, 2022.

[43] Robert S Winternitz. A secure one-way hash function built from DES. In *IEEE Symposium on Security and Privacy (S&P)*, pages 88–88, 1984.

[44] Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In *ACM CCS*, pages 2986–3001, 2021.

[45] Yibin Yang and David Heath. Two shuffles make a RAM: Improved constant overhead zero knowledge RAM. In *USENIX Security*, pages 1435–1452, 2024.

[46] Yibin Yang, David Heath, Carmit Hazay, Vladimir Kolesnikov, and Muthuramakrishnan Venkitasubramaniam. Batchman and robin: Batched and non-batched branching for interactive zk. In *ACM CCS*, pages 1452–1466, 2023.

[47] Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, xiao Wang, Vladmir Kolesnikov, and Daniel Kales. Picnic, 2020. `https://csrc.nist.gov/projects/post-quantum\-cryptography/post-quantum-cryptography-\standardization/round-3-submissions`.