An Improved Vector Commitment Construction with Applications to Signatures

Yalan Wang[*⋆], Bryan Kumara[†¶⋆], Harsh Kasyap[†‖], Liqun Chen[*], Sumanta Sarkar[§], Christopher J.P. Newton[*], Carsten Maple[†‡], Ugur Ilker Atmaca[†‡]

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

[†]The Alan Turing Institute, Email: {hkasyap, cmaple, uatmaca}@turing.ac.uk

[‡]University of Warwick

[§]University of Essex, Email: sumanta.sarkar@essex.ac.uk

[¶]University of Oxford, Email: bryan.kumara@exeter.ox.ac.uk

[‖]Indian Institute of Technology (BHU), Varanasi, India

[⋆] Joint first authors

***Abstract*—All-but-one Vector Commitments (AVCs) randomly opens all but one of the committed vector 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. Correlated GGM (cGGM) trees have been proposed to halve the number of hash calls and Batched AVCs (BAVCs) using a single GGM tree were integrated in the FAEST signature scheme, which improves efficiency and reduces the signature sizes. This paper proposes BACON, a BAVC with aborts that leverages a single 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 and show the theoretical efficiency compared to state-of-the-art.**

***Index Terms*—Correlated GGM trees, Batched all-but-one vector commitments, Post-quantum signatures**

I. Introduction

Multi-party Computation-in-the-Head (MPCitH) paradigm [1] is a popular method to create post-quantum digital signatures using a non-interactive zero-knowledge proof (NIZKP), which proves that a signer knows a secret witness to the output of a function. MPCitH-style signatures can be very efficient for small to medium-sized circuits and have been proposed as candidates for the NIST competition on post-quantum signatures [2]–[5]. However, one drawback of the proof size in a MPCitH-style signature is linear with the size of the underlying circuit which renders this approach inefficient for large-sized circuits.

In 2018, Boyle *et al.* [6] 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 [7]–[20] 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.* [21] 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.

A. Related work

In current instantiations of AVC schemes, an initial seed acting as the root of the Goldreich-Goldwasser-Micali (GGM) tree [22] 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.* [23] replaced the length-doubling PRG with the Davies-Meyer circular correlation robust (CCR) hash function [24] 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

[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.

value. The cGGM tree can reduce the number of hash calls by half compared with the GGM tree, therefore this tree is also referred to as a "half tree". To be compatible with the sVOLE approach [7], [25], 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).

There are a number of works focusing on constructing more efficient AVC schemes [26]–[30] or security [2], [3], [3], [31]–[35]. Details of these work can be referred to the full version. Based on the literature review, *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.*

B. 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 [27]. 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 [30] 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/ [29].
4) We compare our BACON with the large GGM tree-based BAVC with aborts scheme [27] in theory and show our BACON is more efficient.

II. Preliminaries

GGM trees start with an initial random value and expand it to numerous pseudorandom values. This is a binary tree that recursively applies a length-doubling Pseudorandom Generator $PRG : \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ to create a pseudorandom left and right child nodes. To halve the computational cost, a cGGM tree [23] instead uses a CCR hash function H where the left node is defined as $H(parent)$ and the right node as $left \oplus parent = H(parent) \oplus parent$ [23]. The hash function calls is halved as a CCR hash function only requires one hash function whilst a length-doubling PRG often need two. 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 $I = (i^{(1)}, ..., i^{(\tau)}) \subset [N_1] \times ... \times [N_\tau]$. Formally, a BAVC with aborts is defined as follows [27]:

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

- $\mathsf{Setup}(1^\lambda) \to \mathsf{pp}$: 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.
- $\mathsf{Commit}(\mathsf{pp}) \to (\mathsf{com}, \mathsf{decom}, \mathbf{m} = (m_1^{(\alpha)}, ..., m_{N_\alpha}^{(\alpha)})_{\alpha\in[\tau]})$: 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 $\mathbf{m} = (m_1^{(\alpha)}, ..., m_{N_\alpha}^{(\alpha)})_{\alpha\in[\tau]} \in \mathcal{M}^{N_1+\cdots+N_\tau}$.
- $\mathsf{Open}(\mathsf{decom}, I) \to \mathsf{decom}_I \vee \perp$: 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.
- $\mathsf{Verify}(\mathsf{com}, \mathsf{decom}_I, I) \to \left(\left(m_j^{(\alpha)}\right)_{j\in[N_\alpha]\setminus\{i_\alpha\}}\right)_{\alpha\in[\tau]} \vee \perp$: 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).

III. BACON: a BAVC with aborts scheme

BACON adopts the cGGM tree [23] and AES-based CCR hash [30] to construct a BAVC with aborts across three security levels. To provide resistance against collision attacks, BACON adds an iv value as a salt when constructing the cGGM tree. Following section II, let L be the number of leaf nodes, d the layer of the cGGM tree, so $L = 2^d$. Let $\pi : [0, 2^d - 1] \to \{(\alpha, i)\}_{1\le i\le N_\alpha}$ be a bijective function mapping each leaf of the big cGGM tree to each AVC, T_{open} be the threshold of the number of internal nodes to be opened.

Let $\mathsf{H}_{\mathsf{CCR}} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a CCR hash function defined in section II, and $\mathsf{H} : \{0,1\}^* \to \{0,1\}^{2\lambda}$ be a collision-resistant hash function. BACON is shown in Fig. 1.

BACON.Setup establishes the public parameters. In BACON.Commit, a single 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 $\mathsf{com}_i^{(\alpha)}$, $\alpha \in [\tau]$.

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 $|S| \leq \mathsf{T}_{open}$, a new decommitment is produced as a new challenge set is requested and the prover incurs a proof-of-work to do so. A discussion this mechanism can be found in [27]. 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.

IV. Security Analysis

We prove the security properties of BACON in this section. We leverage the extractable-binding security property of the $\mathsf{H}_{\mathsf{CCR}}$ [36]) proved in [30] to prove that an adversary is probabilistically bounded in their ability to win in the extractable-binding game in our scheme. For the hiding property, the proof relies on the CCR property proved in [30]. Further, we model the $\mathsf{H}_{\mathsf{CCR}}$ as an ideal cipher oracle.

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 [30].

A. Security Model

A BAVC with aborts scheme must satisfy *correctness*, *extractable-binding* and *hiding (real-or-random)*:

Definition 2: A BAVC with aborts scheme is correct if for all $I \subset [N_1] \times \cdots \times [N_\tau]$, the following outputs True

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

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

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 3: Let the vector commitment scheme constructed by two hash functions, $\mathsf{H}_{\mathsf{CCR}}$ and H, Ext be a PPT algorithm such that:

- $\mathsf{Ext}(Q_{\mathsf{H}}, Q_{\mathsf{H}_{\mathsf{CCR}}}, \mathsf{com}) \rightarrow ((m_j^{(\alpha)})_{j \in [N_\alpha]})_{\alpha \in [\tau]}$, i.e., given two sets of oracles Q_{H} and $Q_{\mathsf{H}_{\mathsf{CCR}}}$ 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) $\mathsf{com} \leftarrow \mathcal{A}^{\mathsf{H},\mathsf{H}_{\mathsf{CCR}}}(\mathsf{pp})$.
3) $\bar{\mathbf{m}} \leftarrow \mathsf{Ext}(Q_{\mathsf{H}}, Q_{\mathsf{H}_{\mathsf{CCR}}}, \mathsf{com})$, where $\bar{\mathbf{m}} = ((\bar{m_1}^{(\alpha)}, ..., \bar{m_{N_\alpha}}^{(\alpha)})_{\alpha \in [\tau]})$ is a message vector output by Ext.
4) $(\mathbf{m}, \mathsf{decom}_I, I) \leftarrow \mathcal{A}^{\mathsf{H},\mathsf{H}_{\mathsf{CCR}}}(\mathsf{open}, \mathsf{com})$, 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 4: (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) $\mathsf{pp} \rightarrow \mathsf{Setup}(1^\lambda)$.
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 $m_{i_\alpha}^{(\alpha)} \rightarrow \left\{ \begin{array}{ll} \text{random from } \mathcal{M} & \text{if } \bar{b} = 0 \wedge \alpha \leq n \\ \bar{m}_{i_\alpha}^{(\alpha)} & \text{otherwise} \end{array} \right\}$.
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 $\mathsf{AdvAdpHide}_{\mathcal{A},i}^{\mathsf{BAVC}}$ ($\mathsf{AdvSelHide}_{\mathcal{A},i}^{\mathsf{BAVC}}$).

B. Security proofs

Following the security model in IV-A, in this section, we give the security proof of BACON.

*Theorem 1 (**Correctness**):* BACON satisfies correctness. We omit the proof here due to the page limit. Please refer to the full version.

*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

- $\mathsf{BACON.Setup}(1^\lambda, L) \to \mathsf{pp}$: 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 $k_{2j}^i = \mathsf{H_{CCR}}(k_j^{i-1})$, $k_{2j+1}^i = k_{2j}^i \oplus k_j^{i-1}$;
  3) Deinterleave the leaves:
$$\left\{\mathsf{sd}_1^{(\alpha)}, \ldots, \mathsf{sd}_{N_\alpha}^{(\alpha)}\right\}_{\alpha\in[\tau]} \xleftarrow{\pi} \left\{k_0^d, \cdots, k_{2^d-1}^d\right\}.$$
  4) Compute $m_i^{(\alpha)} \leftarrow \mathsf{H_{CCR}}(\mathsf{sd}_i^{(\alpha)})$ and $\mathsf{com}_i^{(\alpha)} \leftarrow \mathsf{H_{CCR}}(\mathsf{sd}_i^{(\alpha)} \oplus 1) || \mathsf{H_{CCR}}(\mathsf{sd}_i^{(\alpha)} \oplus 2)$, 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 $k_{\pi^{-1}(\alpha, i^{(\alpha)})}$ 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 $m_i^{(\alpha)} \leftarrow \mathsf{H_{CCR}}(\mathsf{sd}_i^{(\alpha)})$ and $\mathsf{com}_i^{(\alpha)} \leftarrow \mathsf{H_{CCR}}(\mathsf{sd}_i^{(\alpha)} \oplus 1) || \mathsf{H_{CCR}}(\mathsf{sd}_i^{(\alpha)} \oplus 2)$.
  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 $\left(\left(m_i^{(\alpha)}\right)_{i\in[N_\alpha]\setminus\{i^{(\alpha)}\}}\right)_{\alpha\in[\tau]}$.

Fig. 1. BACON

function H. Therefore, as defined in Definition 3, our proposed cGGM-based BAVC with aborts scheme, BACON, satisfies $\mathsf{AdvEB}_{\mathcal{A}}^{\mathsf{BAVC}} \le \frac{1+q_H(q_H-1)/2}{2^{2\lambda}} + L \cdot (\delta + \varepsilon)$ for any adversary $\mathcal{A}$ making q_E queries to the ideal cipher oracle and q_H queries to the random oracle.

Proof 1: Define the extractor as $\mathrm{Ext}(c, \mathcal{Q}_E)$, 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, 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 $\mathsf{H_{RO}}$ 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

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

□

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

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

Proof 2: Let $\mathcal{B}$ be a distinguisher who can access the CCR hash oracle $\mathcal{O}_{\mathsf{CCR}}$, where $\mathcal{O}_{\mathsf{CCR}}(x, b) = \mathsf{H_{CCR}}(x \oplus \Delta) \oplus b \cdot \Delta$. $\mathcal{B}$, 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 $\mathbf{m}$ 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 $k^i_{j^*+\bar{\alpha}_i} = \mathcal{O}^{\mathsf{CCR}}(s^{i^{(\alpha)}}_{i-1}, \bar{\alpha}_i) \oplus \bar{\alpha}_i s^{i^{(\alpha)}}_{i-1}$. 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 $(\mathsf{com} = h, \mathsf{decom}_I = S, \mathbf{m} = (m_1^{(\alpha)}, ..., m_{N_\alpha}^{(\alpha)})_{\alpha \in [\tau]})$.
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

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

Except the CCR oracle, there are two differences between the real adaptive game and the ideal world simulated by $\mathcal{B}$ for $\mathcal{A}$: $\mathcal{B}$ aborts if the simulation of $\mathcal{O}_{\mathsf{H}}$ fails; k^1_0 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:

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

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

$$\Pr[\mathcal{A} \text{ notices } PRG] \leq \mathsf{Adv}^{PRG}_{\mathcal{B}}(\lambda) \tag{1}$$

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

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

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

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

Finally, putting all cases together, we can get

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

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

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

V. Applications

Our BACON, as a building block, can be applied to the FAEST signature. Due to the page limit, we omit details here. Please refer to full version.

VI. Comparisons

Comparisons among a GGM tree, a pcGGM tree and a cGGM tree are shown in Table I. 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. We can have $2^d - 2 = 2^0 + ... + 2^{d-2} + 2^{d-1} - 1 > 2^{d-2} + 2^{d-1} - 1$. Then the efficiency result of these three tree constructions in descending order of cost as GGM tree [22]>pcGGM tree [23]>cGGM tree [23]. Moreover, adding the number of hash calls used in generating the commitments from the leaf layer, we list the results in Table II. We initially implemented BACON on FAEST. However, the NIST has released an updated version, FAEST v2 which replaced the hash function calls and revisited the evaluation of the AES block ciphers to achieve improvements on both security and practical efficiency [29]. We will integrate BACON with FAEST v2 once the latter is publicly available.

TABLE I
Comparisons among tree constructions

Type	Tree construction	Number of CCR hash calls
Small tree	GGM tree [22]	$(2^d - 2)$
	pcGGM tree [23]	$(2^{d-1} + 2^{d-2} - 1)$
	cGGM tree [23]	$(2^{d-1} - 1)$
Big tree	GGM tree [27]	$2^d - 2$
	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 $d = \log_2 \left(\sum_{i=1}^{\tau} N_i\right)$.

VII. Conclusions

This paper presents BACON, a BAVC based on one cGGM tree which is more efficient than state-of-the-art. We proved the security of BACON in the random oracle model and ideal cipher model.

TABLE II
COMPARISONS BETWEEN TWO LARGE TREE-BASED VECTOR COMMITMENT SCHEMES

Scheme	Number of function calls
[27]	$(2^d - 2)\mathsf{H}_{\mathsf{CCR}} + 3 * 2^d\mathsf{H}$
Our BACON	$(2^{d-1} - 1)\mathsf{H}_{\mathsf{CCR}} + 3 * 2^d\mathsf{H}$

VIII. ACKNOWLEDGMENT

This work was supported by the European Union's Horizon research and innovation program for support under grant agreement numbers: 101069688 (CONNECT), 101070627 (REWIRE), 101095634 (ENTRUST) and 101167904 (CASTOR). These projects are funded by the UK government's Horizon Europe guarantee and administered by UKRI. This work is also partially supported by the Gates Foundation [INV-057591]; under the grant conditions of the Foundation, a Creative Commons Attribution 4.0 Generic License has already been assigned to the Author's Accepted Manuscript.

REFERENCES

[1] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai, "Zero-knowledge proofs from secure multiparty computation," SIAM Journal on Computing, vol. 39, no. 3, pp. 1121–1152, 2009.

[2] C. D. de Saint Guilhem, L. De Meyer, E. Orsini, and N. P. Smart, "BBQ: using AES in picnic signatures," in International Conference on Selected Areas in Cryptography, 2019, pp. 669–692.

[3] C. Baum, C. D. de Saint Guilhem, D. Kales, E. Orsini, P. Scholl, and G. Zaverucha, "Banquet: short and fast signatures from AES," in PKC, 2021, pp. 266–297.

[4] C. Delpech de Saint Guilhem, E. Orsini, and T. Tanguy, "Limbo: efficient zero-knowledge MPCitH-based arguments," in ACM CCS, 2021, pp. 3022–3036.

[5] J. Katz, V. Kolesnikov, and X. Wang, "Improved non-interactive zero knowledge with applications to post-quantum signatures," in ACM CCS, 2018, pp. 525–537.

[6] E. Boyle, G. Couteau, N. Gilboa, and Y. Ishai, "Compressing vector OLE," in ACM CCS, 2018, pp. 896–912.

[7] C. Weng, K. Yang, J. Katz, and X. Wang, "Wolverine: fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits," in IEEE Symposium on Security and Privacy (S&P), 2021, pp. 1074–1091.

[8] S. Dittmer, Y. Ishai, and R. Ostrovsky, "Line-point zero knowledge and its applications," Cryptology ePrint Archive, Paper 2020/1446, 2020. [Online]. Available: https://eprint.iacr.org/2020/1446

[9] C. Baum, A. J. Malozemoff, M. B. Rosen, and P. Scholl, "Mac' n'cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions," in CRYPTO, 2021, pp. 92–122.

[10] K. Yang, P. Sarkar, C. Weng, and X. Wang, "Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field," in ACM CCS, 2021, pp. 2986–3001.

[11] N. Franzese, J. Katz, S. Lu, R. Ostrovsky, X. Wang, and C. Weng, "Constant-overhead zero-knowledge for RAM programs," in ACM CCS, 2021, pp. 178–191.

[12] C. Weng, K. Yang, X. Xie, J. Katz, and X. Wang, "Mystique: Efficient conversions for zero-knowledge proofs with applications to machine learning," in USENIX Security, 2021, pp. 501–518.

[13] S. Dittmer, Y. Ishai, S. Lu, and R. Ostrovsky, "Improving line-point zero knowledge: two multiplications for the price of one," in ACM CCS, 2022, pp. 829–841.

[14] C. Weng, K. Yang, Z. Yang, X. Xie, and X. Wang, "AntMan: Interactive zero-knowledge proofs with sublinear communication," in ACM CCS, 2022, pp. 2901–2914.

[15] C. Baum, L. Braun, A. Munch-Hansen, and P. Scholl, "Moz $\mathbb{Z}_{2^k}$ arella: efficient vector-ole and zero-knowledge proofs over $\mathbb{Z}_{2^k}$," in CRYPTO, 2022, pp. 329–358.

[16] Y. Yang, D. Heath, C. Hazay, V. Kolesnikov, and M. Venkitasubramaniam, "Batchman and robin: Batched and non-batched branching for interactive zk," in ACM CCS, 2023, pp. 1452–1466.

[17] S. Dittmer, K. Eldefrawy, S. Graham-Lengrand, S. Lu, R. Ostrovsky, and V. Pereira, "Boosting the performance of high-assurance cryptography: Parallel execution and optimizing memory access in formally-verified line-point zero-knowledge," in ACM CCS, 2023, pp. 2098–2112.

[18] D. Bui, H. Chu, G. Couteau, X. Wang, C. Weng, K. Yang, and Y. Yu, "An efficient ZK compiler from SIMD circuits to general circuits," Cryptology ePrint Archive, Paper 2023/1610, 2023. [Online]. Available: https://eprint.iacr.org/2023/1610

[19] F. Lin, C. Xing, and Y. Yao, "More efficient zero-knowledge protocols over $\mathbb{Z}_{2^k}$ via galois rings," in CRYPTO, 2024, pp. 424–457.

[20] Y. Yang and D. Heath, "Two shuffles make a RAM: Improved constant overhead zero knowledge RAM," in USENIX Security, 2024, pp. 1435–1452.

[21] C. Baum, L. Braun, C. D. de Saint Guilhem, M. Klooß, E. Orsini, L. Roy, and P. Scholl, "Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head," in CRYPTO, 2023, pp. 581–615.

[22] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions," Journal of the ACM (JACM), vol. 33, no. 4, pp. 792–807, 1986.

[23] X. Guo, K. Yang, X. Wang, W. Zhang, X. Xie, J. Zhang, and Z. Liu, "Half-tree: Halving the cost of tree expansion in COT and DPF," in EUROCRYPT, 2023, pp. 330–362.

[24] R. S. Winternitz, "A secure one-way hash function built from DES," in IEEE Symposium on Security and Privacy (S&P), 1984, pp. 88–88.

[25] E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, P. Rindal, and P. Scholl, "Efficient two-round ot extension and silent non-interactive secure computation," in ACM CCS, 2019, pp. 291–308.

[26] D. Bui, K. Cong, and C. D. de Saint Guilhem, "Faster VOLEith signatures from all-but-one vector commitment and half-tree," in Australasian Conference on Information Security and Privacy, 2025, pp. 205–223.

[27] C. Baum, W. Beullens, S. Mukherjee, E. Orsini, S. Ramacher, C. Rechberger, L. Roy, and P. Scholl, "One tree to rule them all: Optimizing GGM trees and OWFs for post-quantum signatures," in ASIACRYPT, 2025, pp. 463–493.

[28] C. Baum, L. Braun, C. Delphech de Saint Guilheme, M. Kloop, C. Majenz, S. Mukherjee, S. Ramacher, C. Rechberger, E. Orsini, L. Roy, and P. 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.

[29] C. Baum, W. Beullens, L. Braun, C. De Saint Guilhem, M. Klooß, C. Majenz, S. Mukherjee, E. Orsini, S. Ramacher, C. Rechberger et al., "Shorter, Tighter, FAESTer: Optimizations and improved (QROM) analysis for VOLE-in-the-Head signatures," in CRYPTO, 2025.

[30] H. Cui, C. Guo, X. Wang, C. Weng, K. Yang, and Y. Yu, "AES-based CCR hash with high security and its application to zero-knowledge proofs," Cryptology ePrint Archive, Paper 2024/1271, 2024. [Online]. Available: https://eprint.iacr.org/2024/1271

[31] I. Dinur and N. Nadler, "Multi-target attacks on the picnic signature scheme and related protocols," in EUROCRYPT, 2019, pp. 699–727.

[32] G. Zaverucha, M. Chase, D. Derler, S. Goldfeder, C. Orlandi, S. Ramacher, C. Rechberger, D. Slamanig, J. Katz, x. Wang, V. Kolesnikov, and D. Kales, "Picnic," 2020, https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

[33] T. Feneuil, A. Joux, and M. Rivain, "Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs," in CRYPTO, 2022, pp. 541–572.

[34] C. Aguilar-Melchor, N. Gama, J. Howe, A. Hülsing, D. Joseph, and D. Yue, "The return of the sdith," in EUROCRYPT, 2023, pp. 564–596.

[35] S. Kim, B. Lee, and M. Son, "Relaxed vector commitment for shorter signatures," IACR Preprint, 2024.

[36] C. Guo, J. Katz, X. Wang, and Y. Yu, "Efficient and secure multiparty computation from fixed-key block ciphers," in IEEE Symposium on Security and Privacy (S&P), 2020, pp. 825–841.