One Tree to Rule Them All: Optimizing GGM Trees and OWFs for Post-Quantum Signatures

Carsten Baum[*][1], Ward Beullens[†][2], Shibam Mukherjee[‡][3], Emmanuela Orsini[§][4], Sebastian Ramacher[¶][5], Christian Rechberger[‖][6], Lawrence Roy[**][7], and Peter Scholl[††][8]

team@faest.info

[1,7,8]Aarhus University
[1]Technical University of Denmark
[2]IBM Research Zurich
[3,6]TU Graz
[4]Bocconi University
[5]AIT Austrian Institute of Technology

Abstract

The use of MPC-in-the-Head (MPCitH)-based zero-knowledge proofs of knowledge (ZKPoK) to prove knowledge of a preimage of a one-way function (OWF) is a popular approach towards constructing efficient post-quantum digital signatures. Starting with the Picnic signature scheme, many optimized MPCitH signatures using a variety of (candidate) OWFs have been proposed. Recently, Baum et al. (CRYPTO 2023) showed a fundamental improvement to MPCitH, called VOLE-in-the-Head (VOLEitH), which can generically reduce the signature size by at least a factor of two without decreasing computational performance or introducing new assumptions. Based on this, they designed the FAEST signature which uses AES as the underlying OWF. However, in comparison to MPCitH, the behavior of VOLEitH when using other OWFs is still unexplored.

In this work, we improve a crucial building block of the VOLEitH and MPCitH approaches, the so-called all-but-one vector commitment, thus decreasing the signature size of VOLEitH and MPCitH signature schemes. Moreover, by introducing a small Proof of Work into the signing procedure, we can improve the parameters of VOLEitH (further decreasing signature size) *without* compromising the computational performance of the scheme. Based on these optimizations, we propose three VOLEitH signature schemes FAESTER, KuMQuat, and MandaRain based on AES, MQ, and Rain, respectively. We carefully explore the parameter space for these schemes and implement each, showcasing their performance with benchmarks. Our experiments show that these three signature schemes outperform MPCitH-based competitors that use comparable OWFs, in terms of both signature size and signing/verification time.

1 Introduction

The threat of quantum computing has forced cryptographers to develop digital signatures based on new, supposedly quantum-resistant, hardness assumptions. In order to standardize these new signature schemes, NIST started its first post-quantum (PQ) signature standardization process[1] in 2017, where SPHINCS+ [16, 6], Dilithium [33] and FALCON [46] were standardized. With two out of three standardizations relying on hard lattice problems for their security, NIST deemed it necessary to seek additional candidates for standardization whose security is based on a more diverse set of hardness assump-

[*]cabau@dtu.dk
[†]ward@beullens.com
[‡]shibam.mukherjee@iaik.tugraz.at
[§]emmanuela.orsini@unibocconi.it
[¶]sebastian.ramacher@ait.ac.at
[‖]christian.rechberger@tugraz.at
[**]ldr709@gmail.com
[††]peter.scholl@cs.au.dk

[1]`https://csrc.nist.gov/Projects/post-quantum-cryptography/post-quantum-cryptography-standardization/Call-for-Proposals`.

tions[2].

Signatures from Zero-Knowledge Proofs. A well-known technique to build a digital signature scheme is to compile a (public-coin, honest-verifier) zero-knowledge (ZK) proof of knowledge, used in an identification protocol, with the Fiat-Shamir transformation (FS). In particular, a zero-knowledge proof of knowledge (ZKPoK) for an NP relation $\mathcal{R}$ is an interactive protocol that allows the prover to prove knowledge of a witness w for a statement x such that $(x, w) \in \mathcal{R}$, without revealing any further information. In the context of signature (and identification) schemes, this is a proof of knowledge of a secret key k such that $y = F_k(x)$, for a given one-way function (OWF) $F_k(\cdot)$.
A powerful and efficient technique to build such ZK proofs for arbitrary NP relations is the MPC-in-the-Head (MPCitH) framework due to Ishai et al. [39]. However, a significant limitation of many MPCitH-based proofs lies in their proof size which scales linearly with the size of the circuit representation of the statement being proven. Nevertheless, MPCitH is particularly effective with small to medium-sized circuits and leads to efficient post-quantum signature schemes. These schemes are either based solely on symmetric primitives, such as AES [28, 29, 10, 32, 40] and other MPC-friendly one-way functions (OWFs) like LowMC [4], Rain [32], and AIM [42], or well-studied computational hardness assumptions, including syndrome decoding [37, 3, 5], the multivariate quadratic problem (MQ)[15, 45], the permuted kernel problem [1], and the Legendre PRF [18]. This second approach typically results in a more communication-efficient scheme.

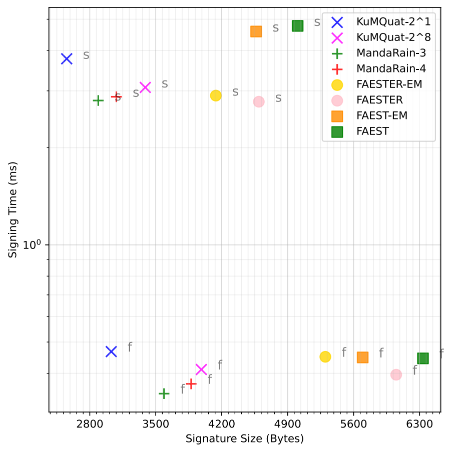
VOLE-ZK and FAEST. In 2018, Boyle et al. [23] proposed a new class of prover-efficient (linear complexity) and scalable ZK proofs, which use commit-and-prove protocols instantiated using vector oblivious linear evaluation (VOLE) correlations. Follow-up works [23, 24, 55, 51, 31, 53, 13, 52, 9] reduced the constants of the linear proof size, surpassing MPCitH schemes in terms of efficiency, in particular when dealing with very large circuits. Compared to MPCitH schemes, the above VOLE-ZK protocols are limited to the designated-verifier setting only. However, recent work by Baum et al. [8] reconciles the advantages of both worlds, resulting in VOLE-ZK proofs that are publicly verifiable. To achieve this, they introduce a technique called VOLE-in-the-Head (VOLEitH) which bears a surprising resemblance to MPCitH-based protocols. Based on VOLEitH, they proposed the FAEST [7] post-quantum signature scheme.
Similarly to MPCitH signature schemes like Banquet [10], BBQ [28], and Helium [40], FAEST relies on AES [2] as its OWF. However, FAEST outperforms MPCitH-based signatures, by having signatures at least twice as small and with similar or better signing and verification times. This makes the VOLEitH-based FAEST as performant as the most optimized MPCitH-based schemes [40], while relying on a very conservative OWF. At the same time, VOLEitH is a relatively new concept, and it remained unexplored to what extent VOLEitH-based signatures can benefit from selecting different OWFs, such as Rain or random multivariate quadratic maps.

1.1 Our Contributions

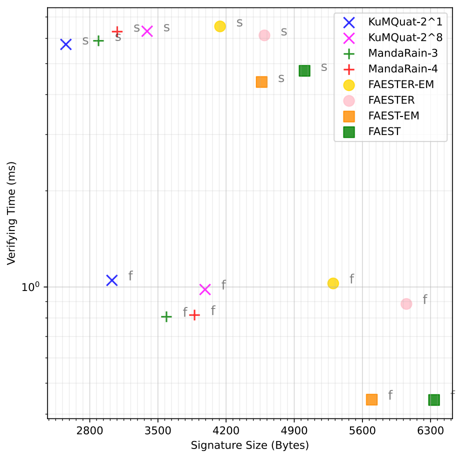
In these work, we present improvements to the core building blocks used in VOLE-in-the-head proof systems, as well as alternative one-way function instantiations that optimize prior approaches and lead to more efficient post-quantum signature candidates.

Improved Batch Vector Commitments. VOLE-in-the-head signatures such as those based on MPC-in-the-head, use multiple GGM-based [38] all-but-one vector commitment schemes to generate correlated randomness for the ZK proofs. These vector commitments are then opened at random challenge points as part of the proof, incurring a decommitment size of $\log(N) \cdot \lambda$ bits per vector commitment that must be sent during the opening phase (where N is the length of the vector and λ is the security parameter). These openings are a substantial part of the setup cost of the ZK proof. We provide a new abstraction, called *batch all-but-one vector commitment* (BAVC) schemes, which captures how multiple vector commitments are used in VOLEitH and MPCitH. We observe that, to instantiate the BAVC abstraction more efficiently, one can interleave multiple vector commitments which drastically reduces the opening size. This batching requires the signer to perform rejection sampling when selecting the points to open, reducing the entropy of the challenge space somewhat. While it might seem that this makes the scheme less secure, one can prove that security is actually preserved: since each rejection sampling step requires the prover to perform a hash function call, we can consider rejection sampling as a *proof of work* done during each signing operation. Any attacker must also perform this proof of work to generate a valid signature. We believe that this technique is of independent interest.

[2]`https://csrc.nist.gov/Projects/pqc-dig-sig/standardization`.



(a) Signing time - Signature Size trade-off, L1 security.



(b) Verification time - Signature Size trade-off, L1 security.

Figure 1: Signature size and runtime trade-off comparison between the proposed signature schemes with FAEST and FAEST-EM. The slow and fast versions are denoted with s and f respectively.

FAESTER. This rejection sampling / proof of work idea can be pushed further, using a technique known as "grinding" [19, 50]. Proof systems naturally have a tradeoff between signature size, computation, and security, and reducing the security can lead to significant improvements in both signature size and computational efficiency. We do this by further reducing the entropy of the challenge space so that some part of the opening process does not even need to be considered. This makes the VOLEitH proof itself slightly less secure, but the overall signature scheme retains the same security level due to the additional proof of work caused by increased rejection probability. It might seem that this trade-off will naturally lead to longer signing times, but the opposite can actually be the case: reducing the challenge entropy significantly reduces the other signing costs, so the scheme is optimized by finding a balance between the costs of the proof of work and those of the rest of the scheme. We applied BAVC and grinding to the FAEST signature scheme, leading to a new digital signature with a signature size of 4KB (an improvement over all signature schemes using AES OWF) while maintaining or improving upon the signing and verification time of FAEST. We name this new improved signature scheme *FAESTER*.

MandaRain & KuMQuat. AES-based OWFs benefit from decades of public scrutiny. However, AES was not designed for use-cases such as VOLEitH which leaves open the possibility that other OWFs may result in faster signing and verification times, and smaller signature sizes. We survey suitable candidate PRFs, ranging from various recent specialized designs in symmetric cryptography [42, 4, 32, 36, 34, 47] to various instances of the MQ problem [15]. We select the Rain [32] and MQ [15] PRFs, from which we construct the new MandaRain and KuMQuat signature schemes using our new commitment optimization. These signature schemes have a signature size as small as 2.6KB, lowest among all VOLEitH and MPCitH-based signature schemes. An overview of our results can be seen in Figure 1.

Allowing Uniform AES Keys in FAEST(ER). In cases where the conservative choice of AES is preferred to alternative OWFs, we show how to tweak the AES proving algorithm so that FAEST and FAESTER can support secret keys that cover the entirety of the AES keyspace. This avoids sampling signing keys via rejection sampling, as done in previous works, so we obtain a simplified key generation algorithm and improve concrete security by 1–2 bits. This improvement comes with no cost in signature size or runtime.

FAEST-d7: Higher-Degree Constraints for AES. We also present a new method of proving AES in VOLE-ZK proof systems, using degree-7 constraints over $\mathbb{F}_2$. Compared with the degree-2 constraints over $\mathbb{F}_{2^8}$ used in the original FAEST (and above), we halve the witness size in the ZK proof. Although proving higher-degree constraints does come with some extra costs, we show that signature sizes can be up to 5% smaller in FAEST-d7. We have not yet implemented this variant, but expect signing and verification times to be similar to FAEST. As a contribution of independent interest, we optimize the method for proving high-degree constraints in the QuickSilver proof system [54], greatly improving the efficiency of the prover.

VOLEitH Parameter Exploration. We systematically investigate the parameter set within the VOLEitH paradigm for constructing a signature scheme, providing insights into the effects of different parameters, including those introduced in this work. These insights contribute to further improvements and trade-offs.

2 Preliminaries

2.1 One-Way Functions

MPCitH and VOLEitH signatures are based on proving knowledge of the preimage to a OWF.[3] In many recent signature schemes like Picnic and FAEST, OWFs are built from a block cipher, according to the following construction.

Construction 1. A one-way-function $\mathsf{F}(k, x)$ can be constructed using a block cipher $E_k(x)$ by setting $\mathsf{F}(k, x) := (x, E_k(x))$, where $E_k(x)$ denotes the encryption of x under the key k. The OWF relation is defined as $((x, y), k) \in R \Leftrightarrow E_k(x) = y$.

2.1.1 The `Rain` OWF

Dobraunig et al. presented a block cipher called `Rain` [32] with a small number of non-linear constraints, designed to optimize the signature size and time when used as a OWF in MPCitH based signature schemes.[4] The resulting signature scheme, Rainier [32], was the first MPCitH signature scheme with less than 5 KB of signature size.
Below we describe the `Rain` round function and we refer to Figure 2 for a graphical overview of `Rain` with 3 rounds.
The `Rain` keyed permutation $f_k(x) : \mathbb{F}_2^\lambda \to \mathbb{F}_2^\lambda$ is defined by the concatenation of a small number r of round functions R_i, $i \in [r]$, i.e. $f_k(x) = R_r \circ \cdots \circ R_2 \circ R_1(x)$. Each R_i, $i \in [r]$, is in turn defined as

$$R_i(x) = \begin{cases} \mathbf{M}_i \cdot S(x + k + \mathbf{c}_i) & i \in [1..r) \\ k + S(x + k + \mathbf{c}_r) & i = r. \end{cases}$$

Here, $S : \mathbb{F}_{2^\lambda} \to \mathbb{F}_{2^\lambda}$ is the field inversion function over $\mathbb{F}_{2^\lambda}$ (mapping 0 to 0), $\mathbf{c}_i \in \mathbb{F}_2^\lambda$ is a round constant, $k \in \mathbb{F}_2^\lambda$ the secret key and $\mathbf{M}_i \in \mathbb{F}_2^{\lambda\times\lambda}$ an invertible matrix.

[3]See Appendix A.1 for definitions.

[4]`Rain` is not a typical block cipher like AES, but rather specifically designed for MPCitH use cases, where it requires that an adversary has access to only one plaintext-ciphertext (*pt-ct*) pair per secret key. When constructing signature schemes, this condition is easily satisfied as `pk` contains the only *pt-ct* pair known to an adversary.

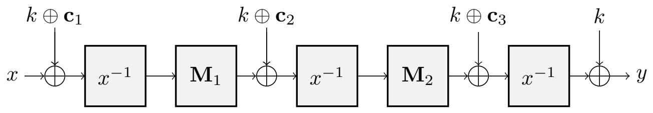


Figure 2: The `Rain` encryption function with r = 3 rounds. $\mathbf{M}_i$ denotes the multiplication with an unstructured invertible matrix over $\mathbb{F}_2$ in the i-th round.

In the VOLEitH setting, similar to MPCitH schemes, the linear layer has a much smaller impact on the performance in comparison to the non-linear layer. Thus to improve diffusion, the authors of `Rain` decided to use different rounds constants $\mathbf{c}_i$ and linear matrices $\mathbf{M}_i$ for each round. Rain comes in two settings, namely `Rain`-3 with 3 rounds and `Rain`-4 with (more conservative) 4 rounds. Despite detailed cryptanalysis carried out by the authors, the best known attacks [44, 56] extend only to two rounds.

2.1.2 Multivariate Quadratic (MQ) OWF

One can also build a OWF from the well-known Multivariate Quadratic problem.

Definition 1. (Multivariate Quadratic Problem). Let $\mathbb{F}_q$ be a finite field and $\mathsf{MQ}_{n,m,q}$ be the set of multivariate maps over $\mathbb{F}_q$ with n variables and m components of the form $\{\mathbf{x}^\mathbf{T} \cdot \mathbf{A}_i \cdot \mathbf{x} + \mathbf{b}_i^\mathbf{T} \cdot \mathbf{x}\}_{i\in[m]}$, where $\mathbf{A}_i \in \mathbb{F}_q^{n\times n}$, are randomly sampled upper triangular matrices and $\mathbf{b}_i \in \mathbb{F}_q^n$ are uniformly sampled vectors. Given $F \in \mathsf{MQ}_{n,m,q}$ and $\mathbf{y} = (y_1, \ldots, y_m) \in \mathbb{F}_q^m$, the MQ problem asks to find $\mathbf{x}$ such that $F(\mathbf{x}) = \mathbf{y}$, i.e. $\big(y_i := \mathbf{x}^\mathbf{T} \cdot \mathbf{A}_i \cdot \mathbf{x} + \mathbf{b}_i^\mathbf{T} \cdot \mathbf{x}\big)_{i\in[m]}$.

The MQ problem has been extensively used in cryptography and used to build both trapdoor [43, 17] and one-way signature schemes [49, 15]. We construct the one-way function $E_\mathbf{x}(\texttt{seed}) = \mathbf{y}$ from the MQ problem, where `seed` is the input to a pseudorandom generator G such that $\mathbf{A}_1, \ldots, \mathbf{A}_m, \mathbf{b}_1, \ldots, \mathbf{b}_m \leftarrow G(\texttt{seed})$. Therefore, when constructing a one-way signature scheme from the MQ problem, ($\mathbf{x}$, `seed`) becomes the `sk` and ($\mathbf{y}$,`seed`) becomes the `pk` (similar to MQOM [15]).

2.2 VOLEitH Signatures

We now give an overview of the VOLEitH framework as the ZK-proof system underlying FAEST.
A *vector oblivious linear evaluation* (VOLE) correlation of length m is a two-party correlation between a prover $\mathcal{P}$ and a verifier $\mathcal{V}$ 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 $q_i \in \mathbb{F}_{2^k}$

such that $q_i = u_i \cdot \Delta - v_i, i = 0, \ldots, m-1$. $\mathcal{P}$ obtains u_i, v_i while $\mathcal{V}$ obtains Δ, q_i. The correlations commit $\mathcal{P}$ to the u_i's as linearly homomorphic commitments, allowing efficient proof systems (see [12] for an overview). One of the main drawbacks of such VOLE-based ZK schemes is that of being inherently designated verifier since the verifier $\mathcal{V}$ needs to know its part of the VOLE correlation to verify the proof, which has to remain secret from the prover for the proof to be sound.

Using VOLEitH, Baum et al. realized a *delayed* VOLE functionality that allows the prover to generate values u_i, v_i of VOLE correlations independently of Δ, q_i and to generate them later instead. This delayed VOLE functionality can in turn be realized from vector commitments (VCs). The main steps of the interactive ZK proof can be computed as before, and only after these, in the last stage of the protocol, the verifier will choose and send to the prover the random value Δ of the correlation. At this point, $\mathcal{P}$ will open the homomorphic commitments and send to $\mathcal{V}$ information which allows it to reconstruct the q_is in the VOLE correlations, check the openings and thus the proof. This guarantees public verifiability, as the final VOLE correlation is defined by the random value Δ chosen as the last step of the proof by the verifier, after all other proof messages have been fixed. Concretely, to obtain the desired soundness, it is necessary to run τ instances of VOLEitH such that $\tau \cdot k = \lambda$. The main steps of the resulting ZK proof using the VOLEitH technique are depicted in Figure 3.

We give a more detailed introduction to the VOLE-in-the-Head approach in Appendix A.

3 Improving Batch Vector Commitments

In this section, we present our result on batch vector commitments (VCs) in the random oracle (RO) model. We start by providing a formal definition of a batch all-but-one vector commitment scheme (BAVC) with abort in the opening phase. This can used in FAEST, and more generally in VOLEitH-based protocols, as well as in most of the known MPC-in-the-head schemes. By making the properties of the used GGM-based instantiation explicit, we manage to achieve an optimized construction that results in shorter signatures.

Informally, a batch all-but-one vector commitment scheme (BAVC) is a two-phase protocol between two PPT machines, a *sender* and a *receiver*. In the first phase, called the *commitment phase*, the sender commits to multiple vectors of messages while keeping them secret; in the second phase, the *decommitment phase*, all but one of the entries of each vector are opened. The vectors may have different lengths. We require the binding and hiding properties of regular commitments, and additionally also that the messages at the *unopened indices* remain hidden, even after opening all other indices of each committed vector. In addition, we do not allow the sender to choose the messages, which instead are just random elements from the message space $\mathcal{M}$. This definition captures how vector commitments are used in MPC-in-the-head or VOLE-in-the-head constructions.

Let τ be the number of vectors, and let the α-th vector have length N_α for $\alpha \in [\tau]$. We will denote by i_τ the index of vector τ that remains unopened and by I the vector $(i_1, \ldots, i_\tau)$ comprising all the indices that remain unopened.

Definition 2 (BAVC)**.** Let H be a random oracle. A *(non-interactive) batch all-but-one vector commitment scheme* BAVC (with message space $\mathcal{M}$) in the RO model is defined by the following PPT algorithms, where all of them have access to a RO, and obtain the security parameter 1^λ as well as $\tau, N_1, \ldots, N_\tau$ as input:

$\mathsf{Commit}() \to (\mathsf{com}, \mathsf{decom}, (m_1^{(\alpha)}, \ldots, m_{N_\alpha}^{(\alpha)})_{\alpha \in [\tau]})$**:** output a commitment com with opening information decom for messages $(m_1^{(\alpha)}, \ldots, m_N^{(\alpha)})_{\alpha \in [\tau]} \in \mathcal{M}^{N_1 + \cdots + N_\tau}$.

$\mathsf{Open}(\mathsf{decom}, I) \to \mathsf{decom}_I \vee \bot$**:** On input an opening decom and the index vector $I \subset [N_1] \times \cdots \times [N_\tau]$, output $\bot$ or an opening decom_I for I.

$\mathsf{Verify}(\mathsf{com}, \mathsf{decom}_I, I) \to ((m_j^{(\alpha)})_{j \in [N_\alpha] \setminus \{i_\alpha\}})_{\alpha \in [\tau]} \vee \bot$**:** Given a commitment com, an opening decom_I, for an index vector I, as well as the index vector I, either output all messages $(m_j^{(\alpha)})_{j \in [N_\alpha] \setminus \{i^\alpha\}}$ (accept the opening) or $\bot$ (reject the opening).

We now define correctness for the commitment scheme. We allow the sender to potentially abort for certain choices of I during Open. Note that this does not pose any problem if the abort probability is low, as aborts only happen during signature generation.

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

$$(\mathsf{com}, \mathsf{decom}, M) \leftarrow \mathsf{Commit}()$$
$$\forall\, \mathsf{decom}_I \leftarrow \mathsf{Open}(\mathsf{decom}, I)$$
$$\text{output } \mathsf{decom}_I = \bot \vee \mathsf{Verify}(\mathsf{com}, \mathsf{decom}_I, I) = M$$

with all but a negligible probability, where $M = (m_1^{(\alpha)}, \ldots, m_N^{(\alpha)})_{\alpha \in [\tau]}$.

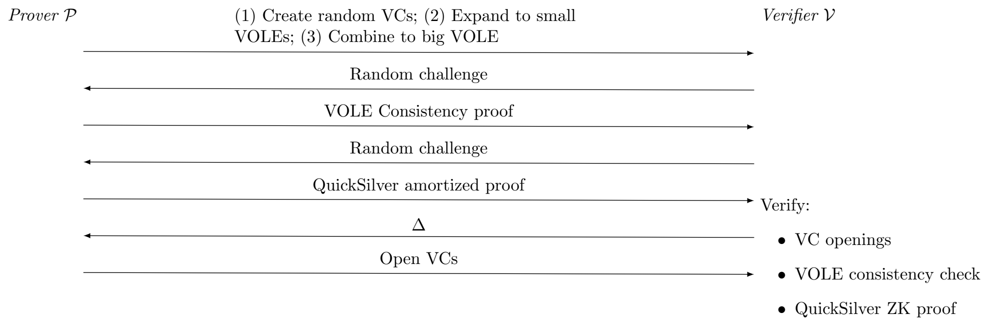


Figure 3: Main steps of the VOLEitH-based Zero-Knowledge proof in FAEST

Informally, we say that a commitment scheme is extractable-binding if there exists an extractor Ext such that for any commitment opening, the extracted message is equal to the opened message. More formally, we have the following definition.

Definition 4 (Extractable-Binding)**.** Let BAVC be defined as above in the RO-model with RO H. Let Ext be a PPT algorithm such that

- $\mathsf{Ext}(Q, \mathsf{com}) \to ((m_j^{(\alpha)})_{j\in[N_\alpha]})_{\alpha\in[\tau]}$, i.e., given a set Q of query-response pairs of random oracle queries, and a commitment com, Ext outputs the committed messages. (Ext may output $m_j^{(\alpha)} = \bot$, e.g. if committed value at this index is invalid.)

For any $\tau, N_\alpha = \mathit{poly}(\lambda)$, define the straightline extractable-binding game for BAVC and stateful adversary $\mathcal{A}^H$ with oracle access to the random oracle H as follows:

1. $\mathsf{com} \leftarrow \mathcal{A}^H(1^\lambda)$
2. $((\overline{m}_1^{(\alpha)}, \dots, \overline{m}_N^{(\alpha)})_{\alpha\in[\tau]}) \leftarrow \mathsf{Ext}(Q, \mathsf{com})$, where Q is the set $\{(x_i, H(x_i))\}$ of query-response pairs of queries $\mathcal{A}$ made to H.
3. $(((m_j^{(\alpha)})_{j\in[N_\alpha]\setminus\{i_\alpha\}})_{\alpha\in[\tau]}, \mathsf{decom}_I, I) \leftarrow \mathcal{A}^H(\mathtt{com})$.
4. Output 1 (success) if:
   $\mathsf{Verify}(\mathsf{com}, \mathsf{decom}_I, I) = ((m_j^{(\alpha)})_{j\in[N_\alpha]\setminus\{i_\alpha\}})_{\alpha\in[\tau]}$,
   but $m_j^{(\alpha)} \neq \overline{m}_j^{(\alpha)}$ for some $\alpha \in [\tau], j \in [N_\alpha]\setminus\{i_\alpha\}$.
   Else output 0 (failure).

We say BAVC is *straightline extractable* w.r.t. Ext if any PPT adversary $\mathcal{A}$ has a negligible probability of winning the extractable binding game. We denote the advantage, i.e. probability to win, by $\mathsf{AdvEB}_{\mathcal{A}}^{\mathsf{BAVC}}$.

We define the n-hiding real-or-random game where $0 < n \leq \tau$. Here, the attacker has to guess if claimed committed values for the first n commitments at the hidden index are correct or not. We allow for a parameter n to permit hybrids in security proofs.

Definition 5 (Hiding (real-or-random))**.** Let BAVC be a vector commitment scheme in the RO-model with random oracle H. The *selective hiding* experiment for BAVC with $\tau, N_\alpha = \mathit{poly}(\lambda)$, parameter n and stateful $\mathcal{A}$ is defined as follows.

1. $\bar{b} \leftarrow \{0,1\}$
2. $(\mathsf{com}, \mathsf{decom}, (\overline{m}_1^{(\alpha)}, \dots, \overline{m}_N^{(\alpha)})_{\alpha\in[\tau]}) \leftarrow \mathsf{Commit}()$
3. $I \leftarrow \mathcal{A}^H(1^\lambda, \mathsf{com})$, where $I \in [N_1] \times \cdots \times [N_\tau]$.
4. $\mathsf{decom}_I \leftarrow \mathsf{Open}(\mathsf{decom}, I)$
5. $m_j^{(\alpha)} \leftarrow \overline{m}_j^{(\alpha)}$ for $j \in [N_\alpha] \setminus \{i_\alpha\}, \alpha \in [\tau]$.
6. Set $m_{i_\alpha}^{(\alpha)} \leftarrow \begin{cases} \text{random from } \mathcal{M} & \text{if } \bar{b} = 0 \wedge \alpha \leq n \\ \overline{m}_{i_\alpha}^{(\alpha)} & \text{otherwise} \end{cases}$
7. $b \leftarrow \mathcal{A}((m_j^{(\alpha)})_{j\in[N_\alpha]}, \mathsf{decom}_i)$.
8. Output 1 (success) if: $b = \bar{b}$, else 0 (failure).

The advantage $\mathsf{AdvSelHide}_{\mathcal{A},i}^{\mathsf{BAVC}}$ of an adversary $\mathcal{A}$ is defined by $\Pr[\mathcal{A} \text{ wins and } n = i] - \frac{1}{2}$ in the hiding experiment. We say BAVC is selectively *hiding* if every PPT adversary $\mathcal{A}$ has a negligible advantage of winning $\mathsf{AdvSelHide}_{\mathcal{A},i}^{\mathsf{BAVC}}$

Note that the GGM-based VC scheme of [8] can be defined using our definitions as well. We show this in Appendix B.

3.1 Using BAVC in FAEST

We now describe how to integrate the previous BAVC definition in FAEST in a black-box way, using rejection sampling to handle aborts and a proof-of-work optimization to reduce the number and length of the

vectors.
FAEST, as described in [8], uses a GGM-based VC scheme to replace a specified number τ of oblivious transfers (OTs) in OT-based zero-knowledge proofs. This is achieved through a compilation step that transforms these proofs into publicly verifiable ones. To be more specific, the compiler treats all OTs as a single functionality, where the sender and the receiver simultaneously query all τ OT instances. Consequently, by syntactically substituting their VC scheme with Definition 2, the compiled protocol will still correctly sign whenever Open does not output $\perp$. We will now address this and demonstrate how our modifications to Open, as well as our security definitions, fit into their framework.

Handling aborts. During the FAEST signing algorithm, the sequence of indices $I \in [N_1] \times \cdots \times [N_\tau]$ for opening the batch all-but-one vector commitment are derived from a λ-bit challenge chal_3 using an injective decoding function $\mathsf{DecodeChallenge}$, where the challenge chal_3 is the output of a hash function H_2^3. To handle aborts in the Open algorithm, we add a counter value ctr to the input of H_2^3. If the challenge chal_3 decodes to a sequence of indices I for which Open fails, then the signing algorithm repeatedly increases ctr and hashes again until it reaches a challenge for which Open succeeds. The counter ctr is included in the signature to allow for efficient verification.

We now argue why this change does not affect the security of FAEST. The proof of [8, Lemma 4] says that for every query to the H_2^3 there are at most 2 out of 2^λ challenge responses that can lead to a forgery, because challenges correspond one-to-one with field elements $\Delta \in \mathbb{F}_{2^\lambda}$, and to cheat, the adversary needs Δ to be a root of a nonzero quadratic polynomial in the Quicksilver check. The proof then considers a union bound over all Q queries to H_2^3 to obtain the term $Q/2^{\lambda-1}$ in the bound on the forgery probability of the adversary. The same proof strategy still works for the signing algorithm with counter, because for every query to H_2^3 there are still at most two challenges that map to the roots of the Quicksilver polynomial.

Using fewer and shorter vector commitments. In the original FAEST scheme we need to have $\prod_{\alpha=1}^{\tau} N_\alpha \geq 2^\lambda$, because the λ-bit challenges need to map injectively to index sequences $I \in [N_1] \times \cdots \times [N_\tau]$. In the setting with aborts, we only need the *non-aborting* challenges to map injectively to index sequences I. Therefore, as an additional optimization, we can choose to reduce the number and/or the length of some of the vectors (reducing the signature size or the signing and verification time respectively), at the cost of increasing the probability of a restart (which slows down signing). Concretely, we set parameters such that $\sum_{\alpha=1}^{\tau} \log N_\alpha = \lambda - w$, and let $I \leftarrow \mathsf{DecodeChallenge}(\mathsf{chal}_3)$ injectively decode the first $\lambda - w$ bits of chal_3. If some of the remaining w bits of chal_3 are nonzero, or if $\mathsf{Open}(I)$ aborts, then the signing algorithm tries again with the next counter. The verifier rejects the signature if the last w bits of chal_3 are not all zero. Since there are still at most two challenges that map to the roots of the Quicksilver polynomial, this optimization does not affect the security proof. The relevant part of the original FAEST and the optimized FAEST signing algorithm are given in Algorithm 1 and Algorithm 2 (Figure 4). Another way to look at this optimization is that we increase efficiency by giving up w bits of security and that we regain security by making the prover solve a proof of 2^w work for each forgery attempt.

New binding and hiding definitions. The security proof of the compilation from OTs to GGM-based VCs is established in [8, Lemma 3]. This proof shows a reduction of special honest-verifier zero-knowledge (SHVZK) of the compiled protocol to SHVZK of the compiled protocol itself and security against the selective hiding game. It uses a hybrid argument to iteratively replace the unopened index value with random values. The same proof technique is applicable using our n-hiding real-or-random security requirement from Definition 5 and showing a hybrid argument progressively selecting $1 \leq n \leq \tau$. Note that, in [8] an adaptive version of the hiding security game is defined, but only selective hiding is employed in the proof.
Furthermore, [8] reduces knowledge soundness to the knowledge soundness of the compiled protocol and the extractable binding security definition. The proof again uses a hybrid argument with abort if the malicious prover successfully opens one of the τ VC instances to a vector of messages differing from those extracted previously. The proof can be seamlessly adapted by replacing the FAEST VC scheme with Definition 2, resulting in essentially the same security proof and bounds.

3.2 Optimizing BAVCs for Signatures

The GGM-based [38] VC construction has been extensively used both in MPCitH based signature schemes like Picnic [25], BBQ [28], Banquet [10], Helium [40] and also VOLEitH-based FAEST to construct the commitment scheme. It expands a ran-

Algorithm 1 FAEST Signing

```
...
$\mathsf{chal}_3 \leftarrow \mathsf{H}_2^3(\mathsf{chal}_2||\tilde{a}||\tilde{b}\,;\,\lambda)$
$I \leftarrow \mathsf{DecodeChallenge}(\mathsf{chal}_3)$
$\mathsf{decom}_I \leftarrow \mathsf{BAVC.Open}(I)$
$\sigma \leftarrow \sigma||\mathsf{decom}_I$
return $\sigma$
```

Algorithm 2 FAESTER Signing

```
...
$\mathsf{ctr} \leftarrow 0$
retry:
$\mathsf{chal}_3 \leftarrow \mathsf{H}_2^3(\mathsf{chal}_2||\tilde{a}||\tilde{b}||\mathsf{ctr}\,;\,\lambda)$
$I \leftarrow \mathsf{DecodeChallenge}(\mathsf{chal}_3[0:\lambda - w - 1])$
$\mathsf{decom}_I \leftarrow \mathsf{BAVC.Open}(I)$
if $\mathsf{decom}_I = \perp$ or $\mathsf{chal}_3[\lambda - w : \lambda] \neq 0^w$ then
    $\mathsf{ctr} \leftarrow \mathsf{ctr} + 1$
    go to retry
end if
$\sigma \leftarrow \sigma||\mathsf{decom}_I||\mathsf{ctr}$
return $\sigma$
```

Figure 4: Signing with FAEST vs signing with FAESTER.

dom seed into a tree of Pseudorandom values by recursively applying a length-doubling Pseudo Random Generator (PRG) to each seed. To obtain a VC, the prover commits to the tree leaves to represent one vector commitment towards the verifier. Then, at a later stage, it can reveal parts of the leaves by opening intermediate seeds (i.e. inner nodes of the tree), allowing the verifier to check the opening against the VC. MPCitH-based signatures usually generate a forest of τ such trees in parallel, whose roots are generated from a single seed. This approach (which we recap in Appendix B) allows expressing τ VCs as one BAVC.

One big tree instead of τ small ones. We now describe an optimization of this construction, where instead of generating a forest of τ trees with $N_1, \ldots, N_\tau$ leaves each, we generate a single GGM tree with $L = \sum_{i=1}^{\tau} N_i$ leaves. Opening all but τ leaves of the big tree is more efficient than opening all but one leaf in each of the τ smaller trees, because with high probability some of the active paths in the tree will merge relatively close to the leaves, which reduces the number of internal nodes that need to be revealed. Importantly, we map entries of the individual vector commitments to the leaves of the tree in an interleaved fashion. The first τ leaves of the tree correspond to the first entry of the τ vector commitments, the next leaves correspond to the second entries, and so on. The other way around would force the τ unopened leaves to be spaced far apart, which is detrimental to the number of nodes that need to be revealed. The number of internal nodes that need to be revealed depends on I, which would cause some variability in the size of the signature. To prevent this, we fix a threshold $\mathsf{T}_{\mathsf{open}}$ for the number of internal nodes in an opening, and we let the Open algorithm abort if the number of nodes exceeds $\mathsf{T}_{\mathsf{open}}$. The value of $\mathsf{T}_{\mathsf{open}}$ controls a trade-off between the opening size of BAVC and the success probability of $\mathsf{BAVC.Open}$.

Towards formalizing our optimized BAVC scheme $\mathsf{BAVC}_{\mathsf{opt}}$, let $\mathsf{PRG}\colon \{0,1\}^\lambda \rightarrow \{0,1\}^{2\lambda}$ be a PRG, $\mathsf{H}\colon \{0,1\}^* \rightarrow \{0,1\}^{2\lambda}$ be a collision-resistant hash function (CRHF) and $\mathsf{G}\colon \{0,1\}^\lambda \rightarrow \{0,1\}^\lambda \times \{0,1\}^{2\lambda}$ be a PRG and CRHF. We define the scheme $\mathsf{BAVC}_{\mathsf{opt}}$, which is parameterized by the number of vectors τ, the lengths of the vectors $N_1, \ldots, N_\tau$, and the opening size threshold $\mathsf{T}_{\mathsf{open}}$. Let $\pi : [L-1, 2L-2] \rightarrow \{(\alpha, i)\}_{1 \le i \le N_\alpha}$ be a bijective mapping from roots of the GGM tree to positions in the vector commitment.

<u>Commit</u>():

1. Set $k \leftarrow \{0,1\}^\lambda$ and let $k_0 \leftarrow k$.
2. For $i \in [0, L-2]$, compute $(k_{2i+1}, k_{2i+2}) \leftarrow \mathsf{PRG}(k_i)$ to create a tree with L leaves $k_{L-1}, \ldots, k_{2L-2}$.
3. Deinterleave the leaves: $\{\mathsf{sd}_1^{(\alpha)}, \ldots, \mathsf{sd}_{N_\alpha}^{(\alpha)}\}_{\alpha \in [\tau]} \overset{\pi}{\leftarrow} \{k_{L-1}, \cdots, k_{2L-2}\}$.
4. Compute $(m_i^{(\alpha)}, \mathsf{com}_i^{(\alpha)}) \leftarrow \mathsf{G}(\mathsf{sd}_i^{(\alpha)})$, for $\alpha \in [\tau]$ and $i \in [N_\alpha]$.
5. Compute $h^{(\alpha)} \leftarrow \mathsf{H}(\mathsf{com}_1^{(\alpha)}, \ldots, \mathsf{com}_{N_\alpha}^{(\alpha)})$ for $\alpha \in [\tau]$ and $h \leftarrow \mathsf{H}(h^{(1)}, \ldots, h^{(\tau)})$.
6. Output the commitment $\mathsf{com} = h$, the opening $\mathsf{decom} = k$ and the messages $(m_1^{(\alpha)}, \ldots, m_{N_\alpha}^{(\alpha)})_{\alpha \in [\tau]}$.

<u>Open</u>($\mathsf{decom} = k, I = (i^{(1)}, \ldots, i^{(\tau)})$):

1. Recompute k_j for and $j \in [0, \ldots, 2L-2]$ from k as in Commit.

2. Let $S = \{k_{L-1}, \dots, k_{2L-2}\}$.

3. For each $\alpha \in [\tau]$, remove $k_{\pi^{-1}(\alpha, i^{(\alpha)})}$ from S.

4. For i from $i = L-2$ to 0:

   If $k_{2i+1} \in S$ and $k_{2i+2} \in S$ then replace both with k_i.

5. If $|S| \leq \mathsf{T}_{\mathsf{open}}$ output the opening information $\mathsf{decom}_I = ((\mathsf{com}^{(\alpha)}_{i^{(\alpha)}})_{\alpha\in[\tau]}, S)$, otherwise output $\bot$.

$\underline{\mathsf{Verify}}(\mathsf{com} = h, \mathsf{decom}_I = (\{\mathsf{com}_{i^{(\alpha)}}\}^{(\alpha)}_{\alpha\in[\tau]}, S), I = (i^{(1)}, \dots, i^{(\tau)}))$**:**

1. Recompute $\mathsf{sd}^{(\alpha)}_i$ from decom_I, for each $\alpha \in [\tau]$ and $i \neq i^{(\alpha)}$ using the available keys in S, and compute $(m^{(\alpha)}_i, \mathsf{com}^{(\alpha)}_i) \leftarrow \mathsf{G}(\mathsf{sd}^{(\alpha)}_i)$.

2. Compute $h^{(\alpha)} = \mathsf{H}(\mathsf{com}^{(\alpha)}_1, \dots, \mathsf{com}^{(\alpha)}_{N_\alpha})$ for each $\alpha \in [\tau]$.

3. If $h \neq \mathsf{H}(h^{(1)}, \dots, h^{(\tau)})$ output $\bot$.

4. Output $((m^{(\alpha)}_i)_{i\in[N_\alpha]\setminus\{i^{(\alpha)}\}})_{\alpha\in[\tau]}$.

Lemma 6 (Extractable Binding)**.** Decompose $\mathsf{G}\colon \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ into $\mathsf{G}(x) := (\mathsf{G}_1(x), \mathsf{G}_2(x))$ and suppose G_2, H are straight-line extractable. Then $\mathsf{BAVC}_{\mathsf{GGM}}$ is straight-line extractable-binding according to Definition 4: Given any adversary $\mathcal{A}$ breaking the extractable-binding of $\mathsf{BAVC}_{\mathsf{opt}}$ with advantage AdvEB we can construct a PPT adversary breaking extractability on G_2, H with advantage

$$\mathsf{AdvEB} \leq L \cdot \mathsf{Adv}_{\mathsf{G}_2} + (\tau+1) \cdot \mathsf{Adv}_{\mathsf{H}}.$$

Proof. The proof is similar to [8, Lemma 1]. We extract Ext after obtaining $\mathsf{com} = h$ using the straight-line extractability of G_2, H. For this, we first find $h^{(1)}$, $\dots, h^{(\tau)}$ which hash to h, and then $\mathsf{com}^{(\alpha)}_i$ for each $i \in [N_\alpha], \alpha \in [\tau]$, in both cases using extractability of H. Then, we extract $\mathsf{sd}^{(\alpha)}_i$ from $\mathsf{com}^{(\alpha)}_i$ using the extractability of G_2, and compute $m^{(\alpha)}_i$ using G_1.
Assume $\mathcal{A}$ breaks extractable binding, i.e. provides values during Open which differ from the extracted $h^{(\alpha)}, \mathsf{com}^{(\alpha)}_i$, $\mathsf{sd}^{(\alpha)}_i$. Then, our constructed adversary will simply guess in advance at which index $\mathcal{A}$ will break extractability of G_2, H and play the extractability game at that index. This guess leads to the loss outlined in the statement. □

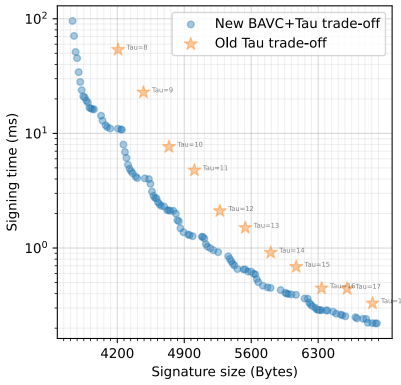
Lemma 7 (Selectively Hiding)**.** Given any adversary $\mathcal{A}$ breaking the selective hiding of $\mathsf{BAVC}_{\mathsf{GGM}}$ for parameter n with advantage $\mathsf{AdvSelHide}_n$ we can construct a PPT adversary breaking the pseudorandomness of G, PRG with advantage

$$\mathsf{AdvSelHide}_n \leq \lceil \log_2(L) \rceil \cdot \mathsf{Adv}_{\mathsf{PRG}} + \mathsf{Adv}_{\mathsf{G}}.$$

Proof. The proof is similar to [8, Lemma 2]. By using that the GGM construction is a puncturable PRF according to [20] and since we know the unopened index I for each commitment vector, and in particular for vector n, in advance, one can iteratively replace the unopened PRG seeds k_i on the path from the root to $\mathsf{sd}^{(n)}_{i^{(n)}}$ *which are not seeds on paths to* $\mathsf{sd}^{(1)}_{i^{(1)}}$, $\dots, \mathsf{sd}^{(n-1)}_{i^{(n-1)}}$ as well as the output of $\mathsf{G}(\mathsf{sd}^{(n)}_{i^{(n)}})$ with uniformly random values. For this to be possible, we fully randomize the seeds on the paths to $\mathsf{sd}^{(1)}_{i^{(1)}}, \dots,$ $\mathsf{sd}^{(n-1)}_{i^{(n-1)}}$ first, to allow for any hybrids distinguishing at indices n to $n+1$ to be meaningful. The bound then follows from the maximal number of hybrids possible. □

3.3 Optimized FAEST and FAEST-EM

This section discusses our optimized FAEST and FAEST-EM signature schemes, namely FAESTER and FAESTER-EM respectively, which benefit from the improved BAVC constructions discussed in Section 3.1 and 3.2. When considering the non-optimized BAVC, the previous VOLEitH signatures FAEST and the recently proposed ReSolved [26] are limited to the signature size and signing/verification runtime trade-off only with respect to τ, the number of "small" VOLEs. Even though flexible, such a trade-off provides an exponential correlation between the signature size and signing time as shown in Figure 5.
With the optimized BAVC, our proposed signature schemes, including FAESTER, enjoy both improved performance and an improved signature size-runtime trade-off. Our experiments show an improvement in the signature size of around 10% for FAESTER when compared to FAEST, in the L1 setting, while maintaining a similar runtime, as shown in the trade-off plot in Figure 5. As a direct consequence of this improvement, FAESTER is the first signature scheme using standard AES with a signature size of 4.5KB. Similarly, FAESTER-EM enjoys a signature size of less than 4KB, with similar signing times. We refer to Appendix D for FAESTER performance for the L3 and L5 security levels.
Figure 8 shows the benefits of our new optimized BAVC for different signature schemes. Table 2 presents our recommended parameter choices for different signature schemes. In the FAEST NIST submission [7], the slow and the fast versions represented


(a) FAEST-128.

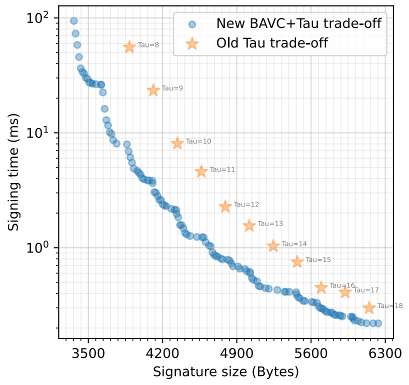

(b) FAEST-EM-128.

Figure 5: FAEST(-EM) τ-signature size and signing time trade-off.

by (s) and (f) respectively were only determined by τ as shown in the first 4 rows. However, for the optimized FAESTER and FAESTER-EM, along with the proposed new signature schemes, we also consider the optimal w and $\mathsf{T}_{\mathsf{open}}$ parameter as described in Section 3.1. We refer to Table 5 for the FAESTER optimized implementation benchmarks.

4 New VOLEitH Signature Schemes

We present three new signature schemes constructed following the footsteps of FAESTER using the optimized BAVC, however, instantiated with different OWFs. The first two variants take advantage of the Rain and MQ OWFs, discussed in Section 2.1.1 and 2.1.2 respectively, to achieve the lowest signature sizes (less than 3 KB) among all MPCitH and VOLEitH signature schemes. The third variant uses AES but with a different approach to proving the S-box, which reduces signature sizes by up to around 5%. We also show how to tweak the original AES proof in FAEST, to allow use of the full AES keyspace, instead of restricting to a subset of all keys.

4.1 MandaRain: VOLEitH + Rain

The MandaRain signature scheme uses two instantiations of the Rain OWF, namely Rain-3 and Rain-4 which use 3 and 4 rounds respectively. Rain has the same block size as its security parameter λ, thus unlike FAEST and FAESTER, Rain can circumvent the need for multiple evaluations of the OWF. The parameters of Rain that we use for MandaRain can be found in Table 3.

We prove Rain using the VOLEitH NIZK proof as described in Section 2.2, with the optimized BAVC (Section 3.2). The prover uses as a witness the secret key k together with the internal state after each round, except for the last round which can be derived from the public key. This gives a total witness length of $l = r\lambda$ bits for r rounds, and proving consistency requires r multiplication checks in $\mathbb{F}_{2^\lambda}$. See Table 1 for a summary of the non-linear complexity of the Rain-3 and Rain-4 OWFs. Compared to the other OWFs, Rain has the smallest number of non-linear constraints that must be checked in ZK resulting in not only a very small signature size but also a competitive signing and verification time. Refer to Table 2 for details on the MandaRain parameters. Similarly to FAEST, Figure 6 presents the parameter set exploration to find the most suitable parameter sets for signature size/runtime trade-offs with and without the BAVC optimization. We see that the signature size can be as small as around 2.8KB for the same or better signing runtime. Refer to Table 5 for the MandaRain optimized implementation benchmarks at the L1 security level. For L3 and L5 benchmarks, refer to Appendix D.

4.2 KuMQuat: VOLEitH + MQ

Using a OWF relying on the MQ problem (Section 2.1.2), we obtain the smallest witness size, and hence the smallest signature size among all VOLEitH and MPCitH signature schemes.

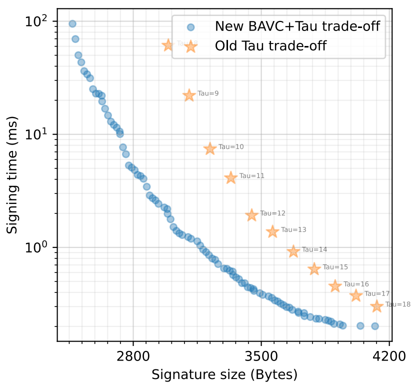
Proving an MQ evaluation in VOLEitH is conceptually straightforward: the witness is the solution $\mathbf{x} \in \mathbb{F}_q^n$ to the system of equations, and there are m quadratic constraints to verify. One challenge is

Table 1: Non-linear complexity of VOLEitH signature schemes using different OWFs.

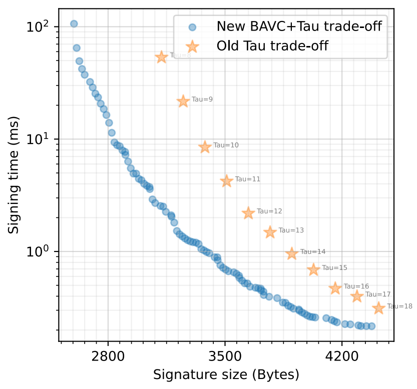
Description	FAEST			FAEST-EM		
λ	AES-128	AES-192	AES-256	AES-EM-128	AES-EM-192	AES-EM-256
No. of S-Boxes in key expansion	40	32	52	0	0	0
No. of S-Boxes in encryption	160	192	224	160	288	448
Total no. of $\mathbb{F}_{2^8}$ constraints	200	416	500	160	288	448
	FAESTER			FAESTER-EM		
λ	AES-128	AES-192	AES-256	AES-EM-128	AES-EM-192	AES-EM-256
No. of S-Boxes in key expansion	40	32	52	0	0	0
No. of S-Boxes in encryption	160	192	224	160	288	448
Total no. of $\mathbb{F}_{2^8}$ constraints	200	416	500	160	288	488
	MandaRain-3			MandaRain-4		
λ	Rain-3-128	Rain-3-192	Rain-3-256	Rain-4-128	Rain-4-192	Rain-4-256
No. of S-Boxes in encryption	3	3	3	4	4	4
Total no. of $\mathbb{F}_{2^\lambda}$ constraints	3	3	3	4	4	4
	KuMQuat-2^1			KuMQuat-2^8		
λ	MQ-$\mathbb{F}_{2^1}$-L1	MQ-$\mathbb{F}_{2^1}$-L3	MQ-$\mathbb{F}_{2^1}$-L5	MQ-$\mathbb{F}_{2^8}$-L1	MQ-$\mathbb{F}_{2^8}$-L3	MQ-$\mathbb{F}_{2^8}$-L5
Total no. of $\mathbb{F}_{2^n}$ constraints	152	224	320	48	72	96

Table 2: VOLEitH signature schemes and their parameters. We denote the signature schemes as SCHEME-$\lambda_{s/f}$. l is the number of VOLE correlations required for the NIZK proof. w and $\mathsf{T}_{\mathsf{open}}$ are the values for the optimized BAVC as described in Section 3.1. τ is the number of VOLE repetitions determining the choice between s (slow) and f (fast) versions. k_0 and k_1 are bit lengths of small VOLEs. B is the padding parameter affecting the security of the VOLE check. Secret key (sk), public key (pk) and signature sizes are in bytes.

Signature Scheme	OWF $E_{sk}(x)$	l	w	$\mathsf{T}_{\mathsf{open}}$	τ	τ_0	τ_1	k_0	k_1	sk size	pk size	sig. size
FAEST-128_s	AES$128_{sk}(x)$	1600	–	–	11	7	4	12	11	16	32	5006
FAEST-128_f	AES$128_{sk}(x)$	1600	–	–	16	0	16	8	8	16	32	6336
FAEST-EM-128_s	AES$128_x(sk) \oplus sk$	1280	–	–	11	7	4	12	11	16	32	4566
FAEST-EM-128_f	AES$128_x(sk) \oplus sk$	1280	–	–	16	0	16	8	8	16	32	5696
FAEST-d7-128_s	AES$128_{sk}(x)$	800	–	–	11	7	4	12	11	16	32	4790
FAEST-d7-128_f	AES$128_{sk}(x)$	800	–	–	16	0	16	8	8	16	32	6020
FAESTER-128_s	AES$128_{sk}(x)$	1600	7	102	11	0	11	11	11	16	32	4594
FAESTER-128_f	AES$128_{sk}(x)$	1600	8	110	16	8	8	8	7	16	32	6052
FAESTER-EM-128_s	AES$128_x(sk) \oplus sk$	1280	7	103	11	0	11	11	11	16	32	4170
FAESTER-EM-128_f	AES$128_x(sk) \oplus sk$	1280	8	112	16	8	8	8	7	16	32	5444
FAESTER-d7-128_s	AES$128_{sk}(x)$	800	5	102	11	0	11	11	11	16	32	4374
FAESTER-d7-128_f	AES$128_{sk}(x)$	800	6	110	16	8	8	8	7	16	32	5732
MandaRain-3-128_s	Rain-3-$128_{sk}(x)$	384	7	100	11	7	4	12	11	16	32	2890
MandaRain-3-128_f	Rain-3-$128_{sk}(x)$	384	8	108	16	0	16	8	8	16	32	3588
MandaRain-4-128_s	Rain-4-$128_{sk}(x)$	512	7	101	11	7	4	12	11	16	32	3082
MandaRain-4-128_f	Rain-4-$128_{sk}(x)$	512	8	110	16	0	16	8	8	16	32	3876
KuMQuat-2^1-$L1_s$	MQ-2^1-$L1_{sk}(x)$	152	7	99	11	7	4	12	11	19	35	2555
KuMQuat-2^1-$L1_f$	MQ-2^1-$L1_{sk}(x)$	152	4	102	16	0	16	8	8	19	35	3028
KuMQuat-2^8-$L1_s$	MQ-2^8-$L1_{sk}(x)$	384	7	100	11	7	4	12	11	48	64	2890
KuMQuat-2^8-$L1_f$	MQ-2^8-$L1_{sk}(x)$	384	4	108	16	0	16	8	8	48	64	3588



(a) MandaRain-3-128.



(b) MandaRain-4-128.

Figure 6: MandaRain τ-signature size and signing runtime trade-off.

Table 3: Rain Parameters

Instance	Seclvl	State	Rounds
`Rain-3-128`	L1	$\mathbb{F}_2^{128}$	3
`Rain-3-192`	L3	$\mathbb{F}_2^{192}$	3
`Rain-3-256`	L5	$\mathbb{F}_2^{256}$	3
`Rain-4-128`	L1	$\mathbb{F}_2^{128}$	4
`Rain-4-192`	L3	$\mathbb{F}_2^{192}$	4
`Rain-4-256`	L5	$\mathbb{F}_2^{256}$	4

Table 4: MQ Parameters

Instance	Seclvl	Field	$m = n$
`MQ-2`^{1}`-L1`	L1	$\mathbb{F}_{2^1}$	152
`MQ-2`^{8}`-L1`	L1	$\mathbb{F}_{2^8}$	48
`MQ-2`^{1}`-L3`	L3	$\mathbb{F}_{2^1}$	224
`MQ-2`^{8}`-L3`	L3	$\mathbb{F}_{2^8}$	72
`MQ-2`^{1}`-L5`	L5	$\mathbb{F}_{2^1}$	320
`MQ-2`^{8}`-L5`	L5	$\mathbb{F}_{2^8}$	96

that a naive approach using QuickSilver would require $O(mn^2)$ multiplications in $\mathbb{F}_{2^\lambda}$. In Section 2.1.2, we describe some optimizations that reduce this to just $O(mn^2 q/\lambda)$ multiplications.

Although the runtime of KuMQuat is not as fast as MandaRain, it still has signing and verification speeds comparable to those of FAEST, for signatures of around half the size. Table 4 shows the MQ parameter choices for our experiments chosen according to the security estimation from [35, 14]. We set $m = n$ (as in MQOM) and choose a field $\mathbb{F}_{2^k}$ for a power k. The field size of the MQ problem and security level determines the choice of n (see Section 2.1.2), which in turn influences the key and signature sizes and the runtime as shown in Table 5. We refer to Table 2 for the recommended parameter choice for the L1 security level. For L3 and L5, parameter choices, we refer to Appendix D. Note that the signature size of KuMQuat depends only mildly on the MQ parameters m, n. One could therefore choose to increase n, m to massively increase the margin of safety against MQ-solving attacks without growing the signature size much.

4.2.1 Optimizations

One implementation difficulty with KuMQuat is the computational cost of the OWF. The MQ function itself has $mn(n+3)/2$ terms[5] (see Definition 1), each with coefficients in $\mathbb{F}_q$, and evaluating the constraints with QuickSilver requires calculating the same number of terms over $\mathbb{F}_{2^\lambda}$. While this seems to require $\tilde{\Theta}(mn^2\lambda)$ work, we used an optimization to reduce this back to just $\tilde{\Theta}(mn^2 \log_2 q)$.

Instead of these m constraints (for $i \in [m]$) over $\mathbb{F}_q$:

$$0 = \sum_{jk} \mathbf{A}_{ijk}\, x_j x_k + \sum_j b_{ij}\, x_j - y_i,$$

we require that $\mathbb{F}_{2^\lambda}$ is a degree $r = \frac{\lambda}{\log_2(q)}$ field extension of $\mathbb{F}_q$, and group the constraints into blocks of r:

$$0 = \sum_{i=ri'}^{ri'+r-1} \alpha^{i-ri'} \left(\sum_{jk} \mathbf{A}_{ijk}\, x_j x_k + \sum_j b_{ij}\, x_j - y_i \right),$$

where α is a generator of $\mathbb{F}_{2^\lambda}$ over $\mathbb{F}_q$. These constraints are equivalent to the original ones, because $\alpha^0, \alpha^1, \ldots, \alpha^{r-1}$ are linearly independent over $\mathbb{F}_q$ since $\mathbb{F}_{2^\lambda}$ is a degree r vector space over $\mathbb{F}_q$. Now, we can precompute this linear combination of constraints

$$\mathbf{A}'_{i'jk} = \sum_{i=0}^{r-1} \alpha^i \mathbf{A}_{(ri'+i)jk}$$

$$b'_{i'j} = \sum_{i=0}^{r-1} \alpha^i b_{(ri'+i)j}$$

$$y'_{i'} = \sum_{i=0}^{r-1} \alpha^i y_{ri'+i}$$

to get $\lceil m/r \rceil$ constraints over $\mathbb{F}_{2^\lambda}$:

$$0 = \sum_{jk} \mathbf{A}'_{i'jk}\, x_j x_k + \sum_j b'_{i'j}\, x_j - y'_{i'}.$$

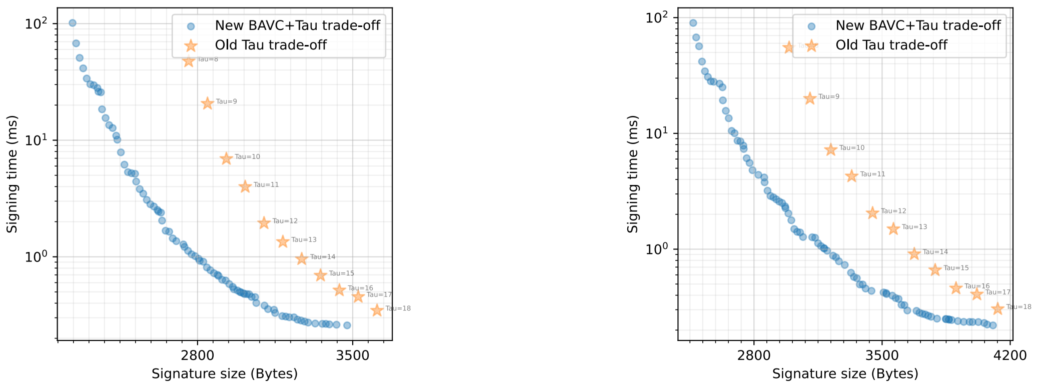
Note that evaluating these constraints for QuickSilver now requires only $\Theta(mn^2/r)$ operations over $\mathbb{F}_{2^\lambda}$. Assuming $\mathbb{F}_{2^\lambda}$ multiplication can be done in $\tilde{\Theta}(\lambda)$ time, this is $\tilde{\Theta}(mn^2 \log_2 q)$ time.

As a final optimization, note that if $r \leq m/r$ then there are exactly r $\mathbf{A}_{ijk}$ elements that get mapped into a single $\mathbf{A}'_{i'jk}$, and that the transformation between them is bijective (and similarly for $\mathbf{b}$ and y). Therefore, sampling all $\mathbf{A}'_{i'}$ uniformly at random from the subset of upper triangular matrices in $\mathbb{F}_{2^\lambda}^{n\times n}$ is equivalent to sampling the original $\mathbf{A}_i$ elements uniformly from the upper triangular matrices in $\mathbb{F}_q^{n\times n}$, for all except very last i'. To save computing this transformation, other than for the last i' we sample the $\mathbf{A}'_{i'}$ and $\mathbf{b}'_{i'}$ directly, instead of going through $\mathbf{A}_{ri'}, \ldots, \mathbf{A}_{ri'+r-1}$. Similarly, for $i' \leq m/r$ we also use $y'_{i'}$ directly in the public key, rather than converting between them and the $\mathbf{y}_i$s.

4.3 Uniform AES Keys in FAEST

When using one-way functions based on AES or Rijndael, as in FAEST(ER) and FAEST(ER)-EM, the

[5]Or $mn(n+1)/2$ in $\mathbb{F}_2$, since then $x^2 = x$ which makes the diagonal of $\mathbf{A}_i$ redundant.



(a) KuMQuat-2^1-L1. (b) KuMQuat-2^8-L1.

Figure 7: KuMQuat τ-signature size and runtime trade-off.

main challenge is proving consistency of the non-linear part of the S-box. We denote this by the function

$$S : x \mapsto x^{254} \in \mathbb{F}_{2^8}$$

When proving AES in zero-knowledge, the committed witness is typically used to derive an input/output pair $(x, y) \in \mathbb{F}_{2^8}^2$ for each S-box, and the prover shows that $y = S(x)$ by proving the degree-2 constraint $xy = 1$. However, this only works when x, y are non-zero; this meant that prior works [30, 11, 8] had to restrict the set of AES keys to those where the input to every S-box is non-zero. This requires adding a rejection sampling step to key generation, and slightly reduces entropy of the signing key, effectively reducing security by 1–2 bits [7, Sec. 10.3.4].

We observe that instead, $y = S(x)$ can be proven for all values of $x, y \in \mathbb{F}_{2^8}$, by the pair of constraints:

$$xy^2 = y \quad \wedge \quad x^2y = x \tag{1}$$

where the first constraint guarantees that we cannot have $x = 0 \wedge y \neq 0$, and the second ensures against $y = 0 \wedge x \neq 0$.

While these constraints have degree-3 over $\mathbb{F}_{2^8}$, when viewed over $\mathbb{F}_2$, their degree is 2 (since squaring is $\mathbb{F}_2$-linear). In FAEST, the witness is initially committed to over $\mathbb{F}_2$, and only lifted to $\mathbb{F}_{2^8}$ for proving the S-box. So, we can easily modify it to prove (1) by linearly computing commitments to the bits of x^2 and y^2 over $\mathbb{F}_2$, before lifting and proving the pair of degree-2 constraints over $\mathbb{F}_{2^8}$. This doubles the number of constraints that are proven, however, in the end, only a random linear combination of all constraints is checked. This means that we can support uniform AES keys with no impact on proof size.

We implemented this tweaked AES proof by modifying the FAEST implementation, and noticed no change in performance when running benchmarks. This is because the main cost of FAEST is the PRG and hashing operations in the BAVC, so merely doubling the number of constraints does not noticeably affect performance.

4.4 FAEST-d7: Proving AES via Degree-7 Constraints

We have also investigated an alternative approach to proving knowledge of a preimage for the AES-based OWFs, using higher degree constraints over $\mathbb{F}_2$, rather than quadratic constraints over $\mathbb{F}_{2^8}$. This allows us to use an AES witness of half the size, which in some cases reduces signature size.

FAEST-d7 is based on the variant of the QuickSilver proof system [54] that allows for proving arbitrary degree-d constraints on the committed witness. In particular, we use degree-7 constraints, since the AES S-box and its inverse can both be expressed as degree-7 circuits over $\mathbb{F}_2$.[6] We combine this with a meet-in-the-middle idea: instead of committing to the AES state after every round, the prover only commits to the state of every other round. Given committed states s_i, s_{i+2}, we can now prove consistency by verifying that $R_i(s_i) = R_{i+1}^{-1}(s_{i+2})$, where R_i is the i-th round function. Each pair of neighbouring AES states can thus be verified with a single degree-7 QuickSilver check. The same idea can be applied to the S-boxes in the key schedule.

[6]The non-linear part of the S-box maps $x \mapsto x^{254}$ in $\mathbb{F}_{2^8}$. Since 254 has Hamming weight 7, and squaring in $\mathbb{F}_{2^8}$ is $\mathbb{F}_2$-linear, we get degree 7 overall.

Computational Efficiency. In QuickSilver, proving a degree-d circuit $C(x_1, \ldots, x_n)$ requires expressing C as a sum of polynomials $\sum_{i=0}^{d} f_i(x_1, \ldots, x_n)$, where each f_i contains monomials only of degree i. While the f_i's need not be computed explicitly, the prover is required to evaluate each f_i. It's not clear how efficiently this can be done for a complex function like the AES S-box.

We observe that it's not necessary to compute the f_i's at all. Instead, to prove the degree-d circuit C, it suffices for the prover to compute the coefficients of a degree-d *univariate* polynomial, given by $g(y) = C(a_1+b_1y, \ldots, a_n+b_ny)$, for values $a_i, b_i \in \mathbb{F}_{2^\lambda}$ known to the prover. Meanwhile, the verifier only needs to evaluate C at a single point. When C is the AES S-box, we estimate the cost for the prover is around 150 multiplications in $\mathbb{F}_{2^\lambda}$. While this is a lot more than the cost of proving 1 multiplication in $\mathbb{F}_{2^8}$, it is still insignificant when compared with all of the PRG and hash calls used in the other components of FAEST. We will include further details of this method in the full version of this paper.

Signature Size. The main advantage of this approach is that the total witness size is halved, from e.g. $l = 1600$ to $l = 800$ at the 128-bit security level. However, this does not come for free, since proving degree-d relations with QuickSilver incurs a cost of $d\tau\lambda$ bits in the signature size. Overall, when applied to FAEST variants with an l-bit witness, we reduce the signature size by $\tau l/2 - 5\tau\lambda$ bits. For the Even-Mansour 128-bit variants, we have $l/2 = 5\lambda$, so there is no change in size. However, for the standard AES variants and the higher security Even-Mansour variants, we see a reduction of up to around 5%.

We have not implemented FAEST-d7, but show in Table 2 the signature sizes it obtains, as well as those of the FAESTER-d7 variant incorporating our GGM tree optimizations.

5 Broader Discussion

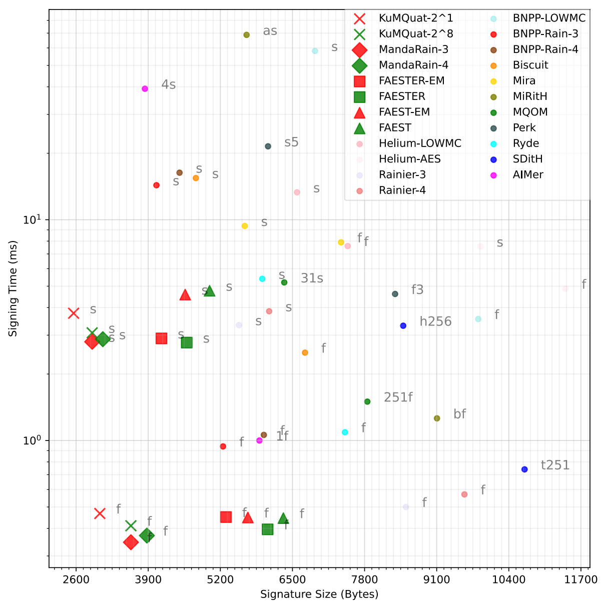
This section compares the existing VOLEitH and MPCitH signature schemes, including the candidates of NIST's call for Additional Signatures, with our proposed optimized signature schemes.

Benchmark platform. To benchmark and compare all the implementations fairly, we run only the most optimized implementation of the signature schemes that is openly available. For the NIST candidates, we refer to the submitted optimized implementations. We measure all the run times on a system with an AMD Ryzen 9 7900X 12-Core CPU, 128 GB memory and running Ubuntu 22.04.

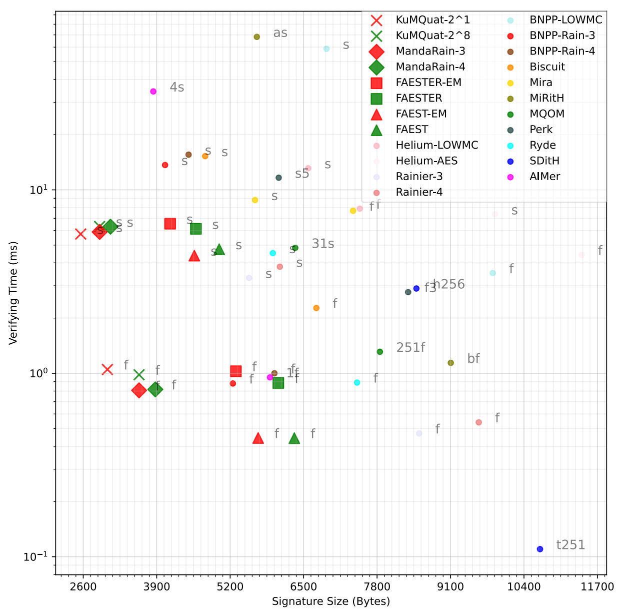
Table 5: Signing Time (ms), Verification Time (ms), and Signature Size (bytes) of different VOLEitH-based signature schemes (optimized implementations). Slow and fast versions are denoted with s and f respectively.

Scheme	Runtime in ms			Size in bytes		
	Keygen	Sign	Verify	sk	pk	Signature
FAEST-128_s	0.0006	4.381	4.102	16	32	5006
FAEST-128_f	0.0005	0.404	0.395	16	32	6336
FAEST-EM-128_s	0.0005	4.151	4.415	16	32	4566
FAEST-EM-128_f	0.0005	0.446	0.474	16	32	5696
FAESTER-128_s	0.0006	3.282	4.467	16	32	4594
FAESTER-128_f	0.0005	0.433	0.610	16	32	6052
FAESTER-EM-128_s	0.0005	3.005	4.386	16	32	4170
FAESTER-EM-128_f	0.0005	0.422	0.609	16	32	5444
MandaRain-3-128_s	0.0018	2.800	5.895	16	32	2890
MandaRain-3-128_f	0.0018	0.346	0.807	16	32	3588
MandaRain-4-128_s	0.0026	2.876	6.298	16	32	3052
MandaRain-4-128_f	0.0026	0.371	0.817	16	32	3876
KuMQuat-2^1-$L1_s$	0.173	4.305	4.107	19	35	2555
KuMQuat-2^1-$L1_f$	0.172	0.539	0.736	19	35	3028
KuMQuat-2^8-$L1_s$	0.174	3.599	4.053	48	64	2890
KuMQuat-2^8-$L1_f$	0.172	0.400	0.623	48	64	3588

Security assumption. The choice of different OWFs allows for a wide variety of security assumptions one can choose from when constructing a VOLEitH signature scheme. For example, using an AES-based OWF results in a highly conservative security guarantee at the cost of a performance penalty in terms of signature size and runtime. This trade-off is similar to the previous state-of-the-art MPCitH signature schemes like BBQ, Banquet, Helium which relied on the standard AES OWF and naturally possessed larger signature size and runtime than their competing schemes which relied on optimized but non-standard OWFs like Rainier [32] or Picnic [25]. Switching to AES-EM construction for VOLEitH signature does not give us the most conservative security guarantees like standard AES, however, the general EM construction is already more than two decades old, thus guaranteeing security in a similar ballpark as of AES while still improving both the signature size and runtime considerably. On the other side, optimized OWFs like Rain and AIM [42] are rather new and not that well studied. For example, in 2023, AIM already witnessed two full round attacks [44, 56] which were later fixed in AIM2 [41]. Due to the mitigation, as per the authors, the signature scheme AIMER using AIM OWF suffers around 10% runtime penalty. This work does not consider using the AIM OWF for constructing a VOLEitH signature scheme as we conjecture that it will lead to worse runtime due large number of Mersenne exponentiation while still giving a signature size similar to Rain. On the other hand, when considering the KuMQuat signature scheme, we benefit from the MQ problem which relies on a different hardness problem, giving us more choices, when compared to the symmetric primitives



(a) L1 Signing.



(b) L1 Verify.

Figure 8: Signature size and runtime comparison between state-of-the-art MPCitH and VOLEitH signature schemes. The slow and fast versions are denoted with s and f respectively. Other special versions are denoted by their short names as per their publicly available specification.

like AES, Rain, or AIM. Similarly, in the recently proposed VOLitH signature scheme ReSolveD [26], their OWF relies on the syndrome decoding problem.

Symmetric Key Primitives. FAEST's zero-knowledge proofs are built out of pseudorandom generators and hash functions, and their instantiation is important for efficiency and security. For consistency with with the FAEST and FAEST-EM proposal [7], we use AES-CTR everywhere a PRG is required, and the SHAKE hash function for all random oracle calls, including those at the leaves of the GGM tree.

Parameters. A careful choice of parameters, including the choice of OWF, is crucial for achieving the best performance of the signature scheme. In the previous sections, we extensively demonstrated the impact of w, $\mathsf{T}_{\mathsf{open}}$, and τ on the signature size and runtime. Additionally, when considering the MQ OWF, the operational field ($\mathbb{F}_n$) is also a crucial factor determining the performance. For example, KuMQuat-2^1-λ operating in $\mathbb{F}_2$ leads to the smallest signature size, however, has the largest number of non-linear constraints among the other proposed VOLEitH signature schemes leading to a long signing and verification runtime. Alternatively, KuMQuat-2^8-λ leads to a larger signature size, due to more witness bits, however, the number of constraints is roughly 70% smaller, leading to a faster runtime than KuMQuat-2^1-λ.

Key Sizes. The key sizes only depend on the underlying OWF and are not affected by the VOLEitH parameters. With the MQ OWF, for example, the operational field $\mathbb{F}_2^n$ and λ determine the size of sk and pk. The key sizes of MandaRain are determined only by λ.

Signature Size and Runtime. FAEST-EM compared to FAEST requires 20-30% less non-linear constraints, which directly influences both the signature size and the runtime, especially for the slow signature variant with a smaller signature size as shown in Table 5. This holds also true for MandaRain which has the smallest number of non-linear constraints enabling it to enjoy the smallest signature runtime along with the smallest signature size after our proposed KuMQuat signature scheme. Looking at the signature size runtime trade-off, in terms of performance we conclude that MandaRain provides a better signature size runtime trade-off, as it has a slightly larger signature size than KuMQuat, however, to the best of our knowledge, it has the smallest runtime among all VOLEitH and MPCitH based signature schemes. We also looked into the possibility of using NIST standardised Ascon[7] as a OWF for constructing VOLEitH signature scheme. However, due to the design structure of Ascon, our estimates showed us

[7]https://csrc.nist.gov/news/2023/lightweight-cryptography-nist-selects-ascon

that the signature size will be much worse than that of standard AES even if we can design an Ascon-style permutation for the OWF.[8] One may also question the fitness of other symmetric primitives which are especially used (friendly optimal design) in MPC, Homomorphic Encryption (HE) and ZKP use-cases. Even though several of these primitives focus on reducing the number of multiplications and their multiplicative depth, such primitives are designed while considering adversaries with higher adversary data complexity. The higher the number of rounds required to guarantee security from a key recovery attack increases the number of witness bits that must be communicated to the verifier. For MPCitH or VOLEitH signature schemes, an adversary knows only the public key or one plaintext-ciphertext pair, though. Hence, VOLEitH- or MPCitH-friendly symmetric primitives like Rain and AIM assume that an adversary knows only the public key, requiring them to have as low as only 3 rounds to guarantee security against key recovery attacks.

For fairness, we compare only the optimized implementations of the signature schemes and thus could not include the recent VOLEitH signature ReSolved [26], as to the best of our knowledge, there exists no optimized implementation for it at the time of writing. However, when comparing the reference implementations of ReSolved with FAEST and FAEST-EM, we conjecture that the optimized implementation of ReSolved should be slower than Rain and FAESTER-EM atleast, if not also FAESTER.

In Figure 8, we compare our proposed VOLEitH signature schemes with other competitive MPCitH and VOLEitH (FAEST) signature schemes. Here, KuMQuat provides the smallest signature size at a high runtime cost. Whereas, MandaRain provides the best signature size runtime trade-off, where it enjoys the best runtime and gives a signature size only second to KuMQuat. Notably, both MandaRain and KuMQuat are the first VOLEitH signature schemes with signature sizes less than 3 KB. This is also the lowest among all the MPCitH signature schemes. FAESTER, using the optimized BAVC, for the first time achieves a signature size of 4.5 KB while still relying on standard AES. Similarly, FAESTER-EM also enjoys a considerably smaller signature size of just 4.1 KB while relying on AES combined with the EM construction.

[8]It might be also interesting to have an analysis on the minimum number of rounds required for security guarantees with Ascon given only one plaintext-ciphertext pair, similar to the security assumptions of Rain or AIM. For AES, this should be conservatively at least 6 rounds as the attack [32, 21] costs 2^{120} time and 2^{120} memory for 4.5 AES rounds, which is still worse than Rain-4 non-linear complexity.

Acknowledgements

This work has been supported by: the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR001120C0085, research grant VIL53029 from VILLUM FONDEN, the Independent Research Fund Denmark under project number 0165-00107B (C3PO), the Digital Europe Program under project number 101091642 ("QCI-CAT"), the Horizon Europe Program under grant agreement number 101096435 (“CONFIDENTIAL6G”), the "DDAI" COMET Module managed by Austrian Research Promotion Agency (FFG).

Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of DARPA. Distribution Statement “A” (Approved for Public Release, Distribution Unlimited).

References

[1] Najwa Aaraj, Slim Bettaieb, Loïc Bidoux, Alessandro Budroni, Victor Dyseryn, Andre Esser, Philippe Gaborit, Mukul Kulkarni, Victor Mateu, Marco Palumbi, et al. Perk. 2023.

[2] Advanced Encryption Standard (AES). National Institute of Standards and Technology, NIST FIPS PUB 197, U.S. Department of Commerce, November 2001.

[3] Carlos Aguilar Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. The return of the SDitH. In Carmit Hazay and Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 564–596. Springer, Heidelberg, April 2023.

[4] Martin R. Albrecht, Christian Rechberger, Thomas Schneider, Tyge Tiessen, and Michael Zohner. Ciphers for MPC and FHE. In Elisabeth Oswald and Marc Fischlin, editors, *Advances in Cryptology - EUROCRYPT 2015 - 34th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Sofia, Bulgaria, April 26-30, 2015, Proceedings, Part I*, volume 9056 of *Lecture Notes in Computer Science*, pages 430–454. Springer, 2015.

[5] Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Matthieu Rivain, Jean-Pierre Tillich, et al. Ryde specifications. 2023.

[6] Jean-Philippe Aumasson, Daniel J. Bernstein, Ward Beullens, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Andreas Hülsing, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, and Bas Westerbaan. Sphincs+ – submission to the 3rd round of the nist post-quantum project, 2022.

[7] Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Sebastian Ramacher, Christian Rechberger, Emmanuela Orsini, Lawrence Roy, et al. FAEST: Algorithm specifications (version 1.1). 2023. `https://faest.info/faest-spec-v1.1.pdf`.

[8] 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 Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 581–615. Springer, Heidelberg, August 2023.

[9] 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 Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference, CRYPTO 2022, Santa Barbara, CA, USA, August 15-18, 2022, Proceedings, Part IV*, volume 13510 of *Lecture Notes in Computer Science*, pages 329–358. Springer, 2022.

[10] Carsten Baum, Cyprien Delpech de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. Banquet: Short and fast signatures from AES. In Juan A. Garay, editor, *Public-Key Cryptography - PKC 2021 - 24th IACR International Conference on Practice and Theory of Public Key Cryptography, Virtual Event, May 10-13, 2021, Proceedings, Part I*, volume 12710 of *Lecture Notes in Computer Science*, pages 266–297. Springer, 2021.

[11] Carsten Baum, Cyprien Delpech de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. Banquet: Short and fast signatures from AES. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 266–297. Springer, Heidelberg, May 2021.

[12] Carsten Baum, Samuel Dittmer, Peter Scholl, and Xiao Wang. Sok: Vector ole-based zero-knowledge protocols. Cryptology ePrint Archive, Paper 2023/857, 2023. `https://eprint.iacr.org/2023/857`.

[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 Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part IV*, volume 12828 of *Lecture Notes in Computer Science*, pages 92–122. Springer, 2021.

[14] Emanuele Bellini, Rusydi H. Makarim, Carlo Sanna, and Javier A. Verbel. An estimator for the hardness of the MQ problem. In Lejla Batina and Joan Daemen, editors, *Progress in Cryptology - AFRICACRYPT 2022: 13th International Conference on Cryptology in Africa, AFRICACRYPT 2022, Fes, Morocco, July 18-20, 2022, Proceedings*, Lecture Notes in Computer Science, pages 323–347. Springer Nature Switzerland, 2022.

[15] Ryad Benadjila, Thibauld Feneuil, and Matthieu Rivain. Mq on my mind: Post-quantum signatures from the non-structured multivariate quadratic problem. *Cryptology ePrint Archive*, 2023.

[16] Daniel J. Bernstein, Andreas Hülsing, Stefan Kölbl, Ruben Niederhagen, Joost Rijneveld, and Peter Schwabe. The sphincs$^+$ signature framework. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 2129–2146. ACM, 2019.

[17] Ward Beullens. MAYO: practical post-quantum signatures from oil-and-vinegar maps. In Riham AlTawy and Andreas Hülsing, editors, *Selected Areas in Cryptography - 28th International Conference, SAC 2021, Virtual Event, September 29 - October 1, 2021, Revised Selected Papers*, volume 13203 of *Lecture Notes in Computer Science*, pages 355–376. Springer, 2021.

[18] Ward Beullens and Cyprien Delpech de Saint Guilhem. Legroast: Efficient post-quantum signatures from the legendre prf. In *International Conference on Post-Quantum Cryptography*, pages 130–150. Springer, 2020.

[19] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. Csi-fish: Efficient isogeny based signatures through class group computations. In Steven D. Galbraith and Shiho Moriai, editors, *Advances in Cryptology - ASIACRYPT 2019 - 25th International Conference on the Theory and Application of Cryptology and Information Security, Kobe, Japan, December 8-12, 2019, Proceedings, Part I*, volume 11921 of *Lecture Notes in Computer Science*, pages 227–247. Springer, 2019.

[20] Dan Boneh and Brent Waters. Constrained pseudorandom functions and their applications. In Kazue Sako and Palash Sarkar, editors, *ASIACRYPT 2013, Part II*, volume 8270 of *LNCS*, pages 280–300. Springer, Heidelberg, December 2013.

[21] Charles Bouillaguet, Patrick Derbez, and Pierre-Alain Fouque. Automatic search of attacks on round-reduced AES and applications. In Phillip Rogaway, editor, *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, volume 6841 of *Lecture Notes in Computer Science*, pages 169–187. Springer, 2011.

[22] Joan Boyar and René Peralta. A new combinational logic minimization technique with applications to cryptology. In *Experimental Algorithms: 9th International Symposium, SEA 2010, Ischia Island, Naples, Italy, May 20-22, 2010. Proceedings 9*, pages 178–189. Springer, 2010.

[23] Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, CCS 2018, Toronto, ON, Canada, October 15-19, 2018*, pages 896–912. ACM, 2018.

[24] 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 Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security, CCS 2019, London, UK, November 11-15, 2019*, pages 291–308. ACM, 2019.

[25] Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, and Greg Zaverucha. Post-quantum zero-knowledge and signatures from symmetric-key primitives. In Bhavani Thuraisingham, David Evans, Tal Malkin, and Dongyan Xu, editors, *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security, CCS 2017, Dallas, TX, USA, October 30 - November 03, 2017*, pages 1825–1842. ACM, 2017.

[26] Hongrui Cui, Hanlin Liu, Di Yan, Kang Yang, Yu Yu, and Kaiyi Zhang. Resolved: Shorter signatures from regular syndrome decoding and vole-in-the-head. Cryptology ePrint Archive, Paper 2024/040, 2024. `https://eprint.iacr.org/2024/040`.

[27] Joan Daemen and Vincent Rijmen. The block cipher rijndael. In Jean-Jacques Quisquater and Bruce Schneier, editors, *Smart Card Research and Applications, This International Conference, CARDIS '98, Louvain-la-Neuve, Belgium, September 14-16, 1998, Proceedings*, volume 1820 of *Lecture Notes in Computer Science*, pages 277–284. Springer, 1998.

[28] Cyprien Delpech de Saint Guilhem, Lauren De Meyer, Emmanuela Orsini, and Nigel P. Smart. BBQ: using AES in picnic signatures. *IACR Cryptol. ePrint Arch.*, page 781, 2019.

[29] Cyprien Delpech de Saint Guilhem, Emmanuela Orsini, and Titouan Tanguy. Limbo: Efficient zero-knowledge mpcith-based arguments. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 3022–3036. ACM, 2021.

[30] Cyprien Delpech de Saint Guilhem, Lauren De Meyer, Emmanuela Orsini, and Nigel P. Smart. BBQ: Using AES in picnic signatures. In Kenneth G. Paterson and Douglas Stebila, editors, *SAC 2019*, volume 11959 of *LNCS*, pages 669–692. Springer, Heidelberg, August 2019.

[31] Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-point zero knowledge and its applications. In Stefano Tessaro, editor, *2nd Conference on Information-Theoretic Cryptography, ITC 2021, July 23-26, 2021, Virtual Conference*, volume 199 of *LIPIcs*, pages 5:1–5:24. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021.

[32] Christoph Dobraunig, Daniel Kales, Christian Rechberger, Markus Schofnegger, and Greg Zaverucha. Shorter signatures based on tailor-made minimalist symmetric-key crypto. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 843–857. ACM, 2022.

[33] Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, Peter Schwabe, Gregor Seiler, and Damien Stehlé. Crystals-dilithium: A lattice-based digital signature scheme. *IACR Trans. Cryptogr. Hardw. Embed. Syst.*, 2018(1):238–268, 2018.

[34] Orr Dunkelman, Nathan Keller, and Adi Shamir. Minimalism in cryptography: The even-mansour scheme revisited. In David Pointcheval and Thomas Johansson, editors, *Advances in Cryptology - EUROCRYPT 2012 - 31st Annual International Conference on the Theory and Applications of Cryptographic Techniques, Cambridge, UK, April 15-19, 2012. Proceedings*, volume 7237 of *Lecture Notes in Computer Science*, pages 336–354. Springer, 2012.

[35] Andre Esser, Javier A. Verbel, Floyd Zweydinger, and Emanuele Bellini. Cryptographic estimators: a software library for cryptographic hardness estimation. *{IACR} Cryptol. ePrint Arch.*, page 589, 2023.

[36] Shimon Even and Yishay Mansour. A construction of a cipher from a single pseudorandom permutation. *J. Cryptol.*, 10(3):151–162, 1997.

[37] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 541–572. Springer, Heidelberg, August 2022.

[38] Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th Annual Symposium on Foundations of Computer Science, West Palm Beach, Florida, USA, 24-26 October 1984*, pages 464–479. IEEE Computer Society, 1984.

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

[40] Daniel Kales and Greg Zaverucha. Efficient lifting for shorter zero-knowledge proofs and post-quantum signatures. *IACR Cryptol. ePrint Arch.*, page 588, 2022.

[41] Seongkwang Kim, Jincheol Ha, Mincheol Son, and ByeongHak Lee. Mitigation on the AIM cryptanalysis. *IACR Cryptol. ePrint Arch.*, page 1474, 2023.

[42] Seongkwang Kim, Jincheol Ha, Mincheol Son, ByeongHak Lee, Dukjae Moon, Joohee Lee, Sangyup Lee, Jihoon Kwon, Jihoon Cho, Hyojin Yoon, and Jooyoung Lee. AIM: symmetric primitive for shorter signatures with stronger security. *IACR Cryptol. ePrint Arch.*, page 1387, 2022.

[43] Aviad Kipnis, Jacques Patarin, and Louis Goubin. Unbalanced Oil and Vinegar signature schemes. In Jacques Stern, editor, *EUROCRYPT'99*, volume 1592 of *LNCS*, pages 206–222. Springer, Heidelberg, May 1999.

[44] Fukang Liu and Mohammad Mahzoun. Algebraic attacks on RAIN and AIM using equivalent representations. *IACR Cryptol. ePrint Arch.*, page 1133, 2023.

[45] Ludovic Perret. Biscuit: Shorter mpc-based signature from posso.

[46] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. Falcon. *Post-Quantum Cryptography Project of NIST*, 2020.

[47] Ron Rivest. Desx. *Unpublished*, 1984.

[48] Lawrence Roy. SoftSpokenOT: Quieter OT extension from small-field silent VOLE in the minicrypt model. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 657–687. Springer, Heidelberg, August 2022.

[49] Simona Samardjiska, Ming-Shing Chen, Andreas Hulsing, Joost Rijneveld, and Peter Schwabe. MQDSS. Technical report, National Institute of Standards and Technology, 2019. available at `https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-2-submissions`.

[50] StarkWare. ethSTARK documentation. Cryptology ePrint Archive, Report 2021/582, 2021. `https://eprint.iacr.org/2021/582`.

[51] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *42nd IEEE Symposium on Security and Privacy, SP 2021, San Francisco, CA, USA, 24-27 May 2021*, pages 1074–1091. IEEE, 2021.

[52] Chenkai Weng, Kang Yang, Zhaomin Yang, Xiang Xie, and Xiao Wang. Antman: Interactive zero-knowledge proofs with sublinear communication. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, CCS 2022, Los Angeles, CA, USA, November 7-11, 2022*, pages 2901–2914. ACM, 2022.

[53] Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. Quicksilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In Yongdae Kim, Jong Kim, Giovanni Vigna, and Elaine Shi, editors, *CCS '21: 2021 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, Republic of Korea, November 15 - 19, 2021*, pages 2986–3001. ACM, 2021.

[54] Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2986–3001. ACM Press, November 2021.

[55] Kang Yang, Chenkai Weng, Xiao Lan, Jiang Zhang, and Xiao Wang. Ferret: Fast extension for correlated OT with small communication. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *CCS '20: 2020 ACM SIGSAC Conference on Computer and Communications Security, Virtual Event, USA, November 9-13, 2020*, pages 1607–1626. ACM, 2020.

[56] Kaiyi Zhang, Qingju Wang, Yu Yu, Chun Guo, and Hongrui Cui. Algebraic attacks on round-reduced RAIN and full AIM-III. *IACR Cryptol. ePrint Arch.*, page 1397, 2023.

A A detailed description of the VOLE-in-the-Head approach

A.1 Definitions

Signature Schemes. We first recall the standard definition of a signature scheme.

Definition 8 (Signature Scheme)**.** A *signature scheme* Sig is a tuple of algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ such that:

1. The *key-generation* algorithm $\mathsf{Gen}(1^\lambda)$ takes as input a security parameter λ in unary representation and outputs a key pair $(\mathsf{sk}, \mathsf{pk})$.
2. The (randomized) *signature* algorithm $\mathsf{Sign}(\mathsf{sk}, \mu)$ takes as input a secret key sk and a message μ and outputs a signature σ.
3. The (deterministic) *verification* algorithm $\mathsf{Verify}(\mathsf{pk}, \mu, \sigma)$ takes as input a public key pk, a message μ and a signature σ and outputs 1 (or accept) or 0.

For correctness, it is required that, for any message μ, the following probability is negligible:

$$\Pr_{\mathsf{Gen},\mathsf{Sign}}\left[\mathsf{Verify}(\mathsf{pk}, \mu, \sigma) = 0 \;\middle|\; \begin{array}{c}(\mathsf{sk}, \mathsf{pk}) \leftarrow \mathsf{Gen} \\ \sigma \leftarrow \mathsf{Sign}(\mathsf{sk}, \mu)\end{array}\right].$$

The standard security notion for digital signature schemes is that of existential unforgeability under adaptive chosen-message attacks (EUF-CMA): an adversary $\mathcal{A}$ given pk and oracle access to $\mathsf{Sign}(\mathsf{sk}, \cdot)$ should not be able to produce a pair (σ, μ) satisfying $\mathsf{Verify}(\mathsf{pk}, \mu, \sigma) = 1$ (for a message μ which was not queried to the signing oracle).

One-Way Functions. VOLE-in-the-head signatures are based on proving knowledge of the preimage to a OWF. We quickly recap their definition.

Definition 9 (One-way function)**.** A polynomial-time function $\mathsf{F}\colon K_\lambda \to C_\lambda$ is called *one-way*, if for every PPT algorithm $\mathcal{A}$ the advantage

$$\mathsf{AdvOWF}^{\mathsf{F}}_{\mathcal{A}} := \Pr\left[\mathsf{F}(k^*) = y \;\middle|\; \begin{array}{c} k \leftarrow K_\lambda \\ y := \mathsf{F}_k(k) \\ k^* \leftarrow \mathcal{A}(1^\lambda, y)\end{array}\right]$$

is negligible in λ.

A.2 VOLE-in-the-Head Proof System

We now explain the VOLE-in-the-head proof system and FAEST signature scheme [8, 7] in more detail.

Generating GGM-based Vector Commitments. VOLE-in-the-Head, like MPC-in-the-Head, uses all-but-one Vector Commitments (VCs) as a starting point. Informally, an all-but-one vector commitment scheme is a two-phase protocol between a *sender* and a *receiver*. In the first phase, also called *commitment phase*, it enables the sender to generate a vector of N random messages $\mathbf{r}_0, \dots, \mathbf{r}_{N-1}$, while keeping them secret; in the second phase, called *decommitment phase*, all but one of the $\mathbf{r}_j$ vectors are opened to the receiver. We require two properties of the scheme: the whole vector must be hidden before the decommitment phase, and the message at the *unopened index* j remains hidden, even after decommitting all other indices of the vector. In addition, the scheme must be binding, meaning that after the commitment phase none of the $\mathbf{r}_j$ in the vector can be changed anymore.

The starting point of the GGM-based [38] all-but-one VC scheme used in FAEST is a binary tree with $N = 2^k$ leaves, which is built from a random seed r as root node and then recursively applying a length-doubling PRG at any node with input the corresponding seed to obtain the two children seeds. The resulting tree has N leaves, each of which are now hashed to obtain 2 values: a message $\mathbf{r}_j$ as well as $\overline{\mathsf{com}}_j$, $j \in \{0, \dots, N-1\}$. To generate the VC, one hashes all $\overline{\mathsf{com}}_j$ using a collision-resistant hash function whose output is the commitment h. To open all messages except $\mathbf{r}_j$, consider the path from r to $\mathbf{r}_j$ through the tree. The sender reveals the seeds corresponding to all siblings of nodes that are on the path (including $\overline{\mathsf{com}}_j$). Given all these revealed values, the receiver can apply the same PRG and reconstruct all $\mathbf{r}_{\bar{j}}, \overline{\mathsf{com}}_{\bar{j}}$ for $\bar{j} \neq j$. By hashing all $\overline{\mathsf{com}}_{\bar{j}}$ as well as the additional $\overline{\mathsf{com}}_j$ and comparing the output with h, correctness of the opening can be checked. Moreover, since $\overline{\mathsf{com}}_j$ is an output of a PRG, it reveals no information about the unopened $\mathbf{r}_j$. The full algorithm is described in Appendix B.

From GGM-based VC to VOLE correlations. To obtain a VOLE correlation $\mathbf{u}, \mathbf{v}, \mathbf{q}, \Delta$ of length m, the prover first generates a VC as described above. Then, following [48], the vector commitment is transformed into a length-m, small-field VOLE correlation in $\mathbb{F}_{2^k}$ in the following way: First, denote the N messages committed in the VC as $\mathbf{r}_0, \dots, \mathbf{r}_{N-1} \in \mathbb{F}_{2^k}^m$. We can write

$$\mathbf{u} = \sum_{i=0}^{N-1} \mathbf{r}_i, \quad \mathbf{v} = \sum_{i=0}^{N-1} i \cdot \mathbf{r}_i.$$

Note that when the verifier asks to open the commitments later, it will learn messages $\mathbf{r}_j$ for all $j \neq \bar{j}$, for some index $\bar{j} \in \{0, \dots, N-1\}$ viewed as an $\mathbb{F}_{2^k}$ element. In this way, $\mathcal{V}$ can compute

$$\begin{aligned} \mathbf{q} &= \sum_{j=0}^{N-1} (\bar{j} - j) \cdot \mathbf{r}_j \\ &= \bar{j} \cdot \sum_{j=0}^{N-1} \mathbf{r}_j - \sum_{j=0}^{N-1} j \cdot \mathbf{r}_j \\ &= \bar{j} \cdot \mathbf{u} - \mathbf{v} \end{aligned} \tag{2}$$

giving the desired VOLE correlation over $\mathbb{F}_{2^k}$. By performing this for τ independent VC instances, the prover has τ independent VOLE correlation vectors for challenges from $[0..N-1]$. Denote these VOLEs as $(\mathbf{u}^{(\alpha)}, \mathbf{V}^{(\alpha)})$, where now $\mathbf{V}^{(\alpha)}$ is a matrix in $\{0,1\}^{m \times k}$ instead of a vector in $\mathbb{F}_{2^k}^m$ for $\alpha \in [\tau]$. The prover commits to its input by forcing all the VOLEs to use the same value $\mathbf{u}$. Towards this, the prover sends the correction value $\mathbf{d}^{(\alpha)} = \mathbf{u}^{(\alpha)} - \mathbf{u}$, for each $\alpha \in [\tau]$ to the verifier, which can then use it later to adjust the small VOLEs. By concatenating the τ VOLE instances created in this way, $\mathcal{P}$ obtains $\mathbf{V} \in \mathbb{F}_2^{m \times \lambda}$, where each row can be seen as an element of $\mathbb{F}_{2^\lambda}$, obtaining the desired VOLE values over $\mathbb{F}_{2^\lambda}$.

In order to ensure that the prover does not cheat during this phase by committing to different secrets in each VOLE instance, the protocol then runs a consistency check. We refer to [7, 8] for further details.

VOLE commitments. As described above, VOLE correlations are lists of tuples (u_i, v_i, q_i) such that the VOLE relation holds for the global key Δ. One such tuple is referred to as an *information-theoretic message authentication code (MAC)* on the value u_i under the global key Δ, since u_i cannot be modified (for a fixed q_i) without knowledge of Δ, and can thus be considered as a designated-verifier commitment to u_i. Since the VOLE relation is linear in Δ, such commitments are trivially additive, and any public linear function can be applied by both parties on the committed value by performing local computation on their respective values.

QuickSilver proof. We now describe how the input of a proof will be committed and how $\mathcal{P}$ and $\mathcal{V}$ execute the information-theoretic VOLE-based QuickSilver proof on the secret input, as applied to VOLE-in-the-head [8].

Let C denote an arithmetic circuit over $\mathbb{F}_2$, containing t multiplication gates, for which the prover knows an input (i.e. the witness) $\mathbf{w} \in \mathbb{F}_2^{\mathbf{n}}$ of length n, such that $\mathsf{C}(\mathbf{w}) = \mathbf{1}$. To prove its knowledge of the witness, the prover will interact with the verifier to evaluate C. Note that all of this will happen before the verifier even learns his outputs of the VOLE correlation. We nevertheless additionally mention the steps that the verifier later takes to "finish" his part of the circuit evaluation in the following.

1. The prover requests $n + t$ MACs from the VOLE protocol. This provides the prover with $(u_i, v_i)_{i\in[n+t]}$. The verifier will later compute $(q_i)_{i\in[n+t]}$ and Δ when performing this step. This has been outlined above.

2. For every input element w_i, for $i \in [n]$, the prover computes $d_i := w_i - u_i$ and sends $(d_i)_{i\in[n]}$ to the verifier. This allows the verifier later to update the commitments of u_i to w_i locally using the linearity of the commitment scheme.

3. For every gate in the circuit C, with input values w_α, w_β, the prover proceeds as follows:

   - Linear gate: the prover uses the linear property of the commitment scheme to compute his shares of the output commitment locally. This does not require any communication towards the verifier, which will later use the linear property to compute his shares of the output commitment.

   - i-th multiplication gate, for $i \in [t]$: the prover computes $w_\gamma := w_\alpha \cdot w_\beta$ and sends $d_{n+i} := w_\gamma - u_{n+1} \in \mathbb{F}_2$ to the verifier. The verifier can later, when holding his shares of the VOLE correlation as well as d_{n+i}, update his shares of the commitment (turning it from a commitment of u_{n+i} to w_γ).

4. Finally, the prover opens the commitment to the output wire of the evaluated circuit by sending v_i to the verifier. It can then later check that $q_i = \Delta - v_i$, i.e. that the opening is correct.

For each multiplication, the prover has generated three valid VOLE MACs $(w_\alpha, v_\alpha, q_\alpha), (w_\beta, v_\beta, q_\beta)$ and $(w_\gamma, v_\gamma, q_\gamma)$ for the t multiplication gates $(\alpha, \beta, \gamma)_i$ contained in the execution of $\mathsf{C}(\mathbf{w})$. But the prover may be malicious, meaning that one has to check that $w_\alpha \cdot w_\beta = w_\gamma$. Therefore, the verifier must check that the prover did not behave maliciously when it sent the t values d_{n+i}.

Checking multiplications. The QuickSilver protocol performs a check of the multiplication values based on the observation that the verifier can compute a value $b_i \in \mathbb{F}_{2^\lambda}$ for each multiplication gate $(\alpha, \beta, \gamma)_i$, for $i \in [t]$, as follows:

$$\begin{aligned} b_i &:= q_\alpha \cdot q_\beta - q_\gamma \cdot \Delta \\ &= v_\alpha \cdot v_\beta + (w_\alpha \cdot v_\beta + w_\beta \cdot v_\alpha - v_\gamma) \cdot \Delta \\ &+ (w_\alpha \cdot w_\beta - w_\gamma) \cdot \Delta^2 \end{aligned} \tag{3}$$

If the prover was honest in the computation of d_i, then the Δ^2 coefficient $w_\alpha \cdot w_\beta - w_\gamma$ disappears, and the verifier needs only to check that

$$\begin{aligned} b_i &\overset{?}{=} a_{0,i} + a_{1,i} \cdot \Delta \qquad (4) \\ \text{for} \quad a_{0,i} &:= v_\alpha \cdot v_\beta \\ \text{and} \quad a_{1,i} &:= w_\alpha \cdot v_\beta + w_\beta \cdot v_\alpha - v_\gamma. \end{aligned}$$

To perform this check, the verifier requires the $a_{0,i}$ and $a_{1,i}$ values which the prover can compute, since they only depend on the w and v values for the multiplication gate $(\alpha, \beta, \gamma)_i$. The prover sends these values (appropriately masked) to the verifier as part of the proof.

After receiving the $(a_{0,i}, a_{1,i})$-pairs from the prover, the verifier evaluates $\mathsf{C}(\mathbf{w})$ by first locally generating his VOLE shares and applying $\mathbf{d}$ to them to commit to the witness and multiplication outputs. It then performs the linear operations and checks if (1) eq. (4) holds for all $i \in [t]$; and (2) if the opened output commitment of $\mathsf{C}(\mathbf{w})$ is a commitment to 1 as outlined above. It rejects if any of the tests fail. It is in these checks that the secrecy of the global key Δ, or in other words the binding property of the VOLE MACs, guarantees the soundness of the proof: to cheat in the proof, the prover would need to modify values u and tags v such that the test of eq. (4) still passes; this requires guessing Δ.

Batch checking multiplications. Since the relation tested in eq. (4) is linear, the QuickSilver protocol optimises the checking procedure by only revealing and thus checking a random linear combination of all the $a_{0,i}, a_{1,i}$. It also modifies the check slightly so that no information about $\mathbf{w}$ is leaked in the process. Both of these modifications are described in [8, 7] in detail.

Checking extension field multiplications. FAEST generates VOLE correlations for values $u \in \mathbb{F}_2$, but correctness of multiplications will be checked in $\mathbb{F}_{2^8}$ since this is the field over which the AES S-box is defined. Since $\mathbb{F}_2$ is a subfield of $\mathbb{F}_{2^8}$, which is itself a subfield of $\mathbb{F}_{2^\lambda}$, one can combine

VOLE MACS for 8 values in $\mathbb{F}_2$ into a VOLE MAC for a single value in $\mathbb{F}_{2^8}$, with the corresponding tag and key still satisfying the VOLE relation for the original global key Δ.
The advantage of the QuickSilver protocol is that one can still commit to the witness bits using $d_i \in \mathbb{F}_2$, which costs only 1 bit of proof size (and therefore signature size) per bit of the witness, and then prove $\mathbb{F}_{2^8}$-multiplications at no extra cost, since $(a_{0,i}, a_{1,i})$ are already in $\mathbb{F}_{2^\lambda}$.

Putting things together. We finally note that the only interactions in the aforementioned protocol now happen when $\mathcal{V}$ sends uniformly random values, such as Δ, to $\mathcal{P}$. This can be made fully non-interactive using the Fiat-Shamir transform.

A.3 Optimizing QuickSilver for Higher Degree Constraints

We now present an optimized method of proving general, degree-d constraints in QuickSilver, which improves upon [54], and which we use for the FAEST-d7 construction from Section 4.4. We show how to prove constraints represented as arithmetic circuits over $\mathbb{F}_q$, where in FAEST-d7 we will use $q = 2$.
We start with some additional notation that generalizes the MACs used in the standard QuickSilver approach defined previously.

Notation. We write $[\![x]\!]^{(d)}$ to mean that a value $x \in \mathbb{F}_q$ held by the prover is committed through VOLE, as follows:

- $\mathcal{P}$ holds coefficients $(a_0, \ldots, a_{d-1}, x) \in \mathbb{F}_{q^r}^d \times \mathbb{F}_q$, representing the polynomial $p_x(\gamma) = a_0 + a_1\gamma + \cdots + a_d\gamma^d$, where a_d equals x lifted into $\mathbb{F}_{q^r}$.
- $\mathcal{V}$ holds $q_x = p_x(\Delta) \in \mathbb{F}_{2^\lambda}$.

Notice that a degree-1 commitment, $[\![x]\!]^{(1)}$, is exactly a standard VOLE commitment, which is how the prover's witness is initially committed. The prover and verifier can then perform the following homomorphic operations on commitments.

<u>Add:</u> $[\![z]\!]^{(d)} = [\![x]\!]^{(d_1)} + [\![y]\!]^{(d_2)}$, where $d = d_2$ and $d_1 \le d_2$:

- $\mathcal{P}$: Let $p_z(\gamma) = p_x(\gamma)\gamma^{d_2-d_1} + p_y(\gamma)$
- $\mathcal{V}$: Let $q_z = q_x\Delta^{d_2-d_1} + q_y$

<u>Multiply:</u> $[\![z]\!]^{(d)} = [\![x]\!]^{(d_1)}[\![y]\!]^{(d_2)}$, where $d = d_1 + d_2$

- $\mathcal{P}$: Output the coefficients of $p_z(\gamma) = p_x(\gamma)p_y(\gamma)$
- $\mathcal{V}$: Output $q_z = q_xq_y$

It's straightforward to these that these operations preserve that invariant that the message is stored in the degree-$d-1$ coefficient and the verifier holds $q_z = p_z(\Delta)$.
Proving a general constraint that is represented by an arithmetic circuit $C : \mathbb{F}_q^n \to \mathbb{F}_q$ can then be done by performing the appropriate additions and multiplications on the commitments. If the circuit has degree d (when viewed as a polynomial) then the final commitment to the output wire will be degree d. It remains to check that the output is a commitment to zero, which can be done as follows.

<u>$\mathsf{CheckZero}([\![x]\!]^{(d)})$:</u>

1. Let $p_x(\gamma) = a_0 + \cdots + a_{d-1}\gamma^{d-1}$, for $a_i \in \mathbb{F}_{q^r}$ (recall that the degree-d coefficient should be zero). $\mathcal{V}$ holds $q_x = p_x(\Delta)$.
2. Sample $r(d-1)$ additional random VOLEs, to obtain masks $[\![s_0]\!]^{(1)}, \ldots, [\![s_{d-2}]\!]^{(1)}$, each represented by a polynomial $p_{s_i}(\gamma) = r_i + s_i\gamma$, where r_i, s_i are both uniform in $\mathbb{F}_{q^r}$.
3. $\mathcal{P}$ computes the degree-$(d-1)$ mask polynomial $p_{s'}(\gamma) = \sum_{i=0}^{d-2} \gamma^i \cdot p_{s_i}(\gamma)$
4. $\mathcal{P}$ sends $p_x(\gamma) + p_{s'}(\gamma)$ to the verifier
5. $\mathcal{V}$ computes the corresponding MAC $q' = q_x + \sum_{i=0}^{d-2} \Delta^i \cdot q_{s_i}$ and check that $(p_x + p_{s'})(\Delta) = q'$

Overall, this protocol is essentially the same check as in the high-degree variant of QuickSilver [54], except with an optimized method of computing the commitment to the circuit output, since we no longer need to express the circuit as a multi-variate polynomial. One other difference is that the mask in step 2 above only needs $r(d-1)$ additional VOLEs, while the method from QuickSilver uses $r(2d-1)$. This is because QuickSilver protects against the case where a malicious verifier may choose its own VOLE outputs or Δ, but this is not possible in VOLE-in-the-head. Indeed, if r_i, s_i are all chosen uniformly and $\Delta \neq 0$, then the coefficients of the mask $p_{s'}$ are uniform, even to a verifier who learns $r_i + s_i\Delta$.
As in QuickSilver, note that when checking a large batch of t constraints, each of degree up to d, it suffices to check that a random linear combination of the t outputs gives a commitment to zero.

Efficiency for AES in FAEST-d7 . Using higher-degree constraints saves communication by allowing for a smaller witness, compared with using only degree-2 constraints and committing to the output of every multiplication gate. However, it comes with some computational overhead due to working with polynomials of higher degree, and also more communication in the final zero check. When adapting this to VOLE-in-the-head, $\mathsf{CheckZero}$ requires sending $(\tau - 1)rd$ $\mathbb{F}_q$ elements to generate the random masks, and a further rd elements to perform the check. If $r \log q = \lambda$ (as in FAEST), then moving from degree-2 to degree-d checks incurs an additional $\tau(d-2)\lambda$ bits of communication. It turns out that for AES in FAEST-d7, this extra cost is either matched, or more than compensated for, by the saving from halving the witness size.

The computational cost of this approach depends on the circuit representation of the constraint. For the non-linear component of the AES S-box, using the degree-7 circuit from [22], the prover would need to do the following univariate polynomial multiplications over $\mathbb{F}_{2^\lambda}$:

- 9 of degree-1 $\times$ degree-1
- 3 of degree-2 $\times$ degree-2
- 4 of degree-2 $\times$ degree-4
- 18 of degree-6 $\times$ degree-1

If we ignore the cost of additions in $\mathbb{F}_{2^\lambda}$, and take into account the fact that the highest-degree term of each polynomial is always 0/1, these multiplications can be done with $9 \cdot 1^2 + 3 \cdot 2^2 + 4 \cdot 2 \cdot 4 + 18 \cdot 6 \cdot 1 = 161$ multiplications in $\mathbb{F}_{2^\lambda}$. This can be reduced slightly further, down to 150, using Karatsuba multiplication. Although this is still much higher than the cost of verifying the S-box with one $\mathbb{F}_{2^8}$ multiplication, the total cost for all S-boxes should still be fairly small, compared with the PRGs and hashing in the remainder of FAEST.

B The GGM-based BAVC scheme

In this appendix, we describe the classical GGM-based VC scheme as an BAVC. Security can be shown identically to our proofs for $\mathsf{BAVC}_{\mathsf{opt}}$.

As before, let $\mathsf{PRG}\colon \{0,1\}^\lambda \to \{0,1\}^{2\lambda}$ be a PRG, $\mathsf{H}\colon \{0,1\}^* \to \{0,1\}^{2\lambda}$ be a collision-resistant hash function (CRHF), $\mathsf{G}\colon \{0,1\}^\lambda \to \{0,1\}^\lambda \times \{0,1\}^{2\lambda}$ be a PRG and CRHF. There are τ VCs of length $N_1, \ldots, N_\tau$ which we wish to generate, where $L = \sum_\alpha N_\alpha$. Let $\pi : [L-1, 2L-2] \to \{(\alpha, i)\}_{1 \le i \le N_\alpha}$ be a bijective mapping from roots of the GGM tree to positions in the vector commitment. We set π so that it maps $L-2+i+\sum_{j=1}^{\alpha-1} N_j$ to (α, i) for each $\alpha \in [\tau], i \in [N_\alpha]$. We define the scheme $\mathsf{BAVC}_{\mathsf{GGM}}$ as follows:

$\underline{\mathsf{Commit}}()$:

1. Set $k \leftarrow \{0,1\}^\lambda$ and let $k_0 \leftarrow k$
2. For $i \in [0, L-2]$, compute $(k_{2i+1}, k_{2i+2}) \leftarrow \mathsf{PRG}(k_i)$ to create a tree with L leaves $k_{L-1}, \ldots, k_{2L-2}$.
3. Assign the leaves: $\{\mathsf{sd}_1^{(\alpha)}, \ldots, \mathsf{sd}_{N_\alpha}^{(\alpha)}\}_{\alpha \in [\tau]} \overset{\pi}{\leftarrow} \{k_{L-1}, \cdots, k_{2L-2}\}$.
4. Compute $(m_i^{(\alpha)}, \mathsf{com}_i^{(\alpha)}) \leftarrow \mathsf{G}(\mathsf{sd}_i^{(\alpha)})$, for $\alpha \in [\tau]$ and $i \in [N_\alpha]$.
5. Compute $h^{(\alpha)} \leftarrow \mathsf{H}(\mathsf{com}_1^{(\alpha)}, \ldots, \mathsf{com}_{N_\alpha}^{(\alpha)})$ for $\alpha \in [\tau]$ and $h \leftarrow \mathsf{H}(h^{(1)}, \ldots, h^{(\tau)})$.
6. Output the commitment $\mathsf{com} = h$, the opening $\mathsf{decom} = k$ and the messages $(m_1^{(\alpha)}, \ldots, m_{N_\alpha}^{(\alpha)})_{\alpha \in [\tau]}$.

$\underline{\mathsf{Open}}(\mathsf{decom} = k, I = (i^{(1)}, \ldots, i^{(\tau)}))$:

1. Recompute k_j for and $j \in [0, \ldots, 2L-2]$ from k as in Commit.
2. Let $S = \{k_{L-1}, \ldots, k_{2L-2}\}$.
3. For each $\alpha \in [\tau]$, remove $k_{\pi^{-1}(\alpha, i^{(\alpha)})}$ from S.
4. For i from $i = L-2$ to 0:

   If $k_{2i+1} \in S$ and $k_{2i+2} \in S$ then replace both with k_i.
5. Output the opening information $\mathsf{decom}_I = ((\mathsf{com}_{i^{(\alpha)}}^{(\alpha)})_{\alpha \in [\tau]}, S)$.

$\underline{\mathsf{Verify}}(\mathsf{com} = h, \mathsf{decom}_I = ((\mathsf{com}_{i^{(\alpha)}}^{(\alpha)})_{\alpha \in [\tau]}, S), I = (i^{(1)}, \ldots, i^{(\tau)}))$:

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
$$(m_i^{(\alpha)}, \mathsf{com}_i^{(\alpha)}) \leftarrow \mathsf{G}(\mathsf{sd}_i^{(\alpha)}).$$
2. Compute $h^{(\alpha)} = \mathsf{H}(\mathsf{com}_1^{(\alpha)}, \ldots, \mathsf{com}_{N_\alpha}^{(\alpha)})$ for each $\alpha \in [\tau]$.
3. If $h \neq \mathsf{H}(h^{(1)}, \ldots, h^{(\tau)})$ output $\perp$.
4. Output $((m_i^{(\alpha)}))_{i \in [N_\alpha] \setminus \{i^{(\alpha)}\}})_{\alpha \in [\tau]}$.

For the given parameters, it is easy to see that $\mathsf{BAVC}_{\mathsf{GGM}}$ has decommitments of size $\approx \tau \cdot (\log L + 2) \cdot \lambda$ bits, while a commitment has exactly 2λ bits.

Table 6: AES and AES-EM OWFs and parameters for FAEST and FAEST-EM as described in [7]. We denote the signature schemes as FAEST-$\lambda_{s/f}$ or FAEST-EM-$\lambda_{s/f}$. l is the number of VOLE correlations required for the NIZK proof. τ is the number of repetition determining the choice between s and f. k_0 and k_1 are bit lengths of small VOLEs. B is the padding parameter affecting the security of the VOLE check. Secret key, public key and signature sizes are in bytes.

Signature Scheme	OWF $E_{sk}(x)$	l	τ	τ_0	τ_1	k_0	k_1	sk size	pk size	sig. size
FAEST-128$_s$	AES128$_{sk}(x)$	1600	11	7	4	12	11	16	32	5006
FAEST-128$_f$	AES128$_{sk}(x)$	1600	16	0	16	8	8	16	32	6336
FAEST-192$_s$	AES192$_{sk}(x_0)\|\|$AES192$_{sk}(x_1)$	3264	16	0	16	12	12	24	64	12744
FAEST-192$_f$	AES192$_{sk}(x_0)\|\|$AES192$_{sk}(x_1)$	3264	24	0	24	8	8	24	64	16792
FAEST-256$_s$	AES256$_{sk}(x_0)\|\|$AES256$_{sk}(x_1)$	4000	22	14	8	12	11	32	64	22100
FAEST-256$_f$	AES256$_{sk}(x_0)\|\|$AES256$_{sk}(x_1)$	4000	32	0	32	8	8	32	64	28400
FAEST-EM-128$_s$	AES128$_x(sk) \oplus sk$	1280	11	7	4	12	11	16	32	4566
FAEST-EM-128$_f$	AES128$_x(sk) \oplus sk$	1280	16	0	16	8	8	16	32	5696
FAEST-EM-192$_s$	Rijndael192$_x(sk) \oplus sk$	2304	16	0	16	12	12	24	64	10824
FAEST-EM-192$_f$	Rijndael192$_x(sk) \oplus sk$	2304	24	0	24	8	8	24	64	13912
FAEST-EM-256$_s$	Rijndael256$_x(sk) \oplus sk$	3584	22	14	8	12	11	32	64	20956
FAEST-EM-256$_f$	Rijndael256$_x(sk) \oplus sk$	3584	32	0	32	8	8	32	64	26736

C Faest design and parameters

Here we recall the main design choices of [7].

VOLEitH Parameters. To efficiently construct VOLE correlations over $\mathbb{F}_{2^\lambda}$, [7] executes parallel instances of the VOLEitH protocol on smaller fields. The resulting tags and keys are then concatenated to yield a correlation in a larger field. Concretely, for each repetition parameter $\tau \in \mathbb{N}$ and security parameter λ, we can fix integers k_0, k_1 and τ_0, τ_1 such that $\lambda = k_0\tau_0 + k_1\tau_1$, where

$$k_0 := \lceil \lambda/\tau \rceil \qquad k_1 := \lfloor \lambda/\tau \rfloor, \qquad \text{and}$$
$$\tau_0 := (\lambda \bmod \tau) \qquad \tau_1 := \tau - \tau_0.$$

In this way, the protocol produces τ_0 (resp. τ_1) VOLEs in $\mathbb{F}_{2^{k_0}}$ (resp. $\mathbb{F}_{2^{k_1}}$), that concatenated produce a correlation in $\mathbb{F}_2^\lambda$. In table 6, we report the set of parameters used in [7].

OWF. FAEST signature follows the same construction as described in Section 2.2, where the OWF F is the circuit description of AES and AES-EM proved with publicly verifiable non-interactive honest verifier ZKPoK using FS transformation using VOLEitH described in Section 2.2. When constructing OWF with AES or AES-EM, it is required that the input block size is equal to the security parameter λ. However, in case of the standard AES, the block size is fixed to 128 bits which requires two separate instantiations when $\lambda = 192, 256$. More concretely, similar to the FAEST NIST submission, we also use similar constructions AES192$_{sk}(x_0)\|$AES192$_{sk}(x_1)$ and AES256$_{sk}(x_0)\|$AES256$_{sk}(x_1)$ respectively, where x_0 and x_1 are two separate plaintexts. When $\lambda = 192$, the additional bits are truncated.

The single key Even-Mansour scheme [36, 34, 47] is a way to constructs a block cipher f from a cryptographic permutation π by adding a key sk to the plaintext and then adding sk again to the output of the permutation function, $f_{sk}(x) := sk + \pi(x + sk)$. Rainier [32] proposed a tweaked version of the standard AES, namely AES-EM, which uses the single key EM construction as a OWF where AES and Rijndael are the permutations. By leveraging this particular construction, the authors successfully removed the AES key schedule and made the round keys public to the verifier, reducing the number of constraints that needs to be proven in MPCitH NIZK. Such a optimization is also applicable to the VOLEitH NIZK. In contrast to AES, when using AES-EM as OWF for $\lambda = 192, 256$, one requires using Rijndael block cipher [27], where the state size is equal to λ bits, thus not requiring multiple instantiations. Refer to Table 6 for more details on the OWF instantiation and the FAEST parameters. In AES and AES-EM, SBox inverse is the only non-linear operation operating in $\mathbb{F}_{2^8}$. Refer to Table 1 for details on the non-linear complexity of AES and AES-EM OWFs.

D Additional Benchmarks

Here we put the additional benchmark numbers for the signature schemes and the OWFs used for all security levels. Table 7 describes the benchmark results of FAEST as in the NIST submission and Table 6 describes the parameters used for the same. Table 8 and Table 7 describe the parameter set and the benchmark results of FAESTER for all security levels. Similarly in Table 10,11,12,13, we describe the parameter sets and the performance numbers for our new signature schemes MandaRain and KuMQuat.

Table 7: Signing Time (ms), Verification Time (ms) and Signature Size (bytes) of the FAEST and FAEST-EM optimized implementation.

Scheme	Runtime in ms			Size in Bytes		
	Keygen	Sign	Verify	sk	pk	Signature
FAEST-128$_{\mathsf{s}}$	0.0006	4.381	4.102	16	32	5006
FAEST-128$_{\mathsf{f}}$	0.0005	0.404	0.395	16	32	6336
FAEST-192$_{\mathsf{s}}$	0.0021	10.855	10.85	24	64	12744
FAEST-192$_{\mathsf{f}}$	0.0022	1.185	1.177	24	64	16792
FAEST-256$_{\mathsf{s}}$	0.003	14.373	14.365	32	64	22100
FAEST-256$_{\mathsf{f}}$	0.0035	1.639	1.583	32	64	28400
FAEST-EM-128$_{\mathsf{s}}$	0.0005	4.151	4.415	16	32	4566
FAEST-EM-128$_{\mathsf{f}}$	0.0005	0.446	0.474	16	32	5696
FAEST-EM-192$_{\mathsf{s}}$	0.0012	10.577	10.882	24	48	10824
FAEST-EM-192$_{\mathsf{f}}$	0.0012	1.081	1.083	24	48	13912
FAEST-EM-256$_{\mathsf{s}}$	0.0024	14.046	14.089	32	64	20956
FAEST-EM-256$_{\mathsf{f}}$	0.0025	1.568	1.583	32	64	26736

Table 8: AES and AES-EM OWFs and parameters for FAESTER and FAESTER-EM. We denote the signature schemes as FAESTER-$\lambda_{\mathsf{s/f}}$ or FAESTER-EM-$\lambda_{\mathsf{s/f}}$. l is the number of VOLE correlations required for the NIZK proof. τ is the number of repetition determining the choice between slow (s) and fast (f). k_0 and k_1 are bit lengths of small VOLEs. Secret key, public key and signature sizes are in bytes.

Signature Scheme	OWF $E_{sk}(x)$	l	w	$\mathsf{T}_{\mathsf{open}}$	τ	τ_0	τ_1	k_0	k_1	sk size	pk size	sig. size
FAESTER-128$_{\mathsf{s}}$	$\text{AES128}_{sk}(x)$	1600	7	102	11	0	11	11	11	16	32	4594
FAESTER-128$_{\mathsf{f}}$	$\text{AES128}_{sk}(x)$	1600	8	110	16	8	8	8	7	16	32	6052
FAESTER-192$_{\mathsf{s}}$	$\text{AES192}_{sk}(x_0)\|\text{AES192}_{sk}(x_1)$	3264	12	162	16	4	12	12	11	24	64	12028
FAESTER-192$_{\mathsf{f}}$	$\text{AES192}_{sk}(x_0)\|\text{AES192}_{sk}(x_1)$	3264	8	163	24	16	8	8	7	24	64	16100
FAESTER-256$_{\mathsf{s}}$	$\text{AES256}_{sk}(x_0)\|\text{AES256}_{sk}(x_1)$	4000	6	245	22	8	14	12	11	32	64	21752
FAESTER-256$_{\mathsf{f}}$	$\text{AES256}_{sk}(x_0)\|\text{AES256}_{sk}(x_1)$	4000	8	246	32	24	8	8	7	32	64	28084
FAESTER-EM-128$_{\mathsf{s}}$	$\text{AES128}_{x}(sk) \oplus sk$	1280	7	103	11	0	11	11	11	16	32	4170
FAESTER-EM-128$_{\mathsf{f}}$	$\text{AES128}_{x}(sk) \oplus sk$	1280	8	112	16	8	8	8	7	16	32	5444
FAESTER-EM-192$_{\mathsf{s}}$	$\text{Rijndael192}_{x}(sk) \oplus sk$	2304	8	162	16	8	8	12	11	24	48	10108
FAESTER-EM-192$_{\mathsf{f}}$	$\text{Rijndael192}_{x}(sk) \oplus sk$	2304	8	176	24	16	8	8	7	24	48	13532
FAESTER-EM-256$_{\mathsf{s}}$	$\text{Rijndael256}_{x}(sk) \oplus sk$	3584	6	218	22	8	14	12	11	32	64	19744
FAESTER-EM-256$_{\mathsf{f}}$	$\text{Rijndael256}_{x}(sk) \oplus sk$	3584	8	234	32	24	8	8	7	32	64	26036

Table 9: Signinig Time (ms), Verification Time (ms) and Signature Size (bytes) of the FAESTER and FAESTER-EM optimized implementation.

Scheme	Runtime in ms			Size in Bytes		
	Keygen	Sign	Verify	sk	pk	Signature
FAESTER-128$_{\mathsf{s}}$	0.0006	3.282	4.467	16	32	4594
FAESTER-128$_{\mathsf{f}}$	0.0005	0.433	0.610	16	32	6052
FAESTER-192$_{\mathsf{s}}$	0.0021	8.930	16.783	24	64	12028
FAESTER-192$_{\mathsf{f}}$	0.0022	1.093	2.177	24	64	16100
FAESTER-256$_{\mathsf{s}}$	0.003	11.708	24.512	32	64	21752
FAESTER-256$_{\mathsf{f}}$	0.0035	1.453	2.801	32	64	27899
FAESTER-EM-128$_{\mathsf{s}}$	0.0005	3.005	4.386	16	32	4170
FAESTER-EM-128$_{\mathsf{f}}$	0.0005	0.422	0.609	16	32	5444
FAESTER-EM-192$_{\mathsf{s}}$	0.0012	7.845	18.509	24	48	10108
FAESTER-EM-192$_{\mathsf{f}}$	0.0012	0.969	2.134	24	48	13532
FAESTER-EM-256$_{\mathsf{s}}$	0.0024	11.676	22.811	32	64	19744
FAESTER-EM-256$_{\mathsf{f}}$	0.0025	1.542	2.946	32	64	26036

Table 10: Rain OWF and parameters for MandaRain. We denote the signature scheme as MandaRain-N-$\lambda_{s/f}$, where N is the number of Rain rounds. l is the number of VOLE correlations (length of extended witnesses in bits) required for the NIZK proof. τ is the number of repetition determining the choice between slow (s) and fast (f). k_0 and k_1 are bit lengths of small VOLEs. Secret key, public key and signature sizes are in bytes.

Scheme	OWF $E_k(x)$	l	w	$\mathsf{T_{open}}$	τ	τ_0	τ_1	k_0	k_1	sk size	pk size	sig. size
MandaRain-3-128$_s$	Rain-3-128$_{sk}(x)$	384	7	100	11	7	4	12	11	16	32	2890
MandaRain-3-128$_f$	Rain-3-128$_{sk}(x)$	384	8	108	16	0	16	8	8	16	32	3588
MandaRain-4-128$_s$	Rain-4-128$_{sk}(x)$	512	7	101	11	7	4	12	11	16	32	3082
MandaRain-4-128$_f$	Rain-4-128$_{sk}(x)$	512	8	110	16	0	16	8	8	16	32	3876
MandaRain-3-192$_s$	Rain-3-192$_{sk}(x)$	576	8	183	16	16	0	12	0	24	48	7132
MandaRain-3-192$_f$	Rain-3-192$_{sk}(x)$	576	8	184	24	24	0	8	0	24	48	8540
MandaRain-4-192$_s$	Rain-4-192$_{sk}(x)$	768	8	181	16	16	0	12	0	24	48	7492
MandaRain-4-192$_f$	Rain-4-192$_{sk}(x)$	768	7	184	24	24	0	8	0	24	48	9116
MandaRain-3-256$_s$	Rain-3-256$_{sk}(x)$	768	6	246	22	14	8	12	11	32	64	12896
MandaRain-3-256$_f$	Rain-3-256$_{sk}(x)$	768	7	248	32	32	0	8	0	32	64	15220
MandaRain-4-256$_s$	Rain-4-256$_{sk}(x)$	1024	6	240	22	14	8	12	11	32	64	13408
MandaRain-4-256$_f$	Rain-4-256$_{sk}(x)$	1024	7	248	32	32	0	8	0	32	64	16244

Table 11: Signing Time (ms), Verification Time (ms) and Signature Size (bytes) of MandaRain-3 and MandaRain-4 optimized implementation.

Scheme	Runtime in ms			Size in Bytes		
	Keygen	Sign	Verify	sk	pk	Signature
MandaRain-3-128$_s$	0.0018	2.800	5.895	16	32	2890
MandaRain-3-128$_f$	0.0018	0.346	0.807	16	32	3588
MandaRain-4-128$_s$	0.0026	2.876	6.298	16	32	3052
MandaRain-4-128$_f$	0.0026	0.371	0.817	16	32	3876
MandaRain-3-192$_s$	0.0047	7.275	19.043	24	48	7132
MandaRain-3-192$_f$	0.0047	0.879	1.968	24	48	8540
MandaRain-4-192$_s$	0.0061	7.025	18.006	24	48	7492
MandaRain-4-192$_f$	0.0061	1.012	2.142	24	48	9116
MandaRain-3-256$_s$	0.0064	9.016	21.217	32	64	12896
MandaRain-3-256$_f$	0.0064	1.357	2.751	32	64	15220
MandaRain-4-256$_s$	0.0084	9.438	21.412	32	64	13408
MandaRain-4-256$_f$	0.0084	1.630	3.088	32	64	16244

Table 12: MQ OWF and parameters for KuMQuat. We denote the signature scheme as KuMQuat-P-$\lambda_{s/f}$, where P defines the 2 prime power field ($\mathbb{F}_{2^n}$). l is the number of VOLE correlations required for the ZK proof. τ is the number of repetition determining the choice between slow (s) and fast (f). k_0 and k_1 are bit lengths of small VOLEs. Secret key, public key and signature sizes are in bytes.

Scheme	OWF $E_k(x)$	l	w	$\mathsf{T_{open}}$	τ	τ_0	τ_1	k_0	k_1	sk size	pk size	sig. size
KuMQuat-2^1-L1$_s$	MQ-2^1-L1$_{sk}(x)$	152	7	99	11	7	4	12	11	19	35	2555
KuMQuat-2^1-L1$_f$	MQ-2^1-L1$_{sk}(x)$	152	4	102	16	0	16	8	8	19	35	3028
KuMQuat-2^8-L1$_s$	MQ-2^8-L1$_{sk}(x)$	384	7	100	11	7	4	12	11	48	64	2890
KuMQuat-2^8-L1$_f$	MQ-2^8-L1$_{sk}(x)$	384	4	108	16	0	16	8	8	48	64	3588
KuMQuat-2^1-L3$_s$	MQ-2^1-L3$_{sk}(x)$	224	8	181	16	16	0	12	0	28	52	6404
KuMQuat-2^1-L3$_f$	MQ-2^1-L3$_{sk}(x)$	224	8	182	24	0	16	8	8	28	52	7436
KuMQuat-2^8-L3$_s$	MQ-2^8-L3$_{sk}(x)$	576	8	184	16	16	0	12	0	72	96	7180
KuMQuat-2^8-L3$_f$	MQ-2^8-L3$_{sk}(x)$	576	8	164	24	0	16	8	8	72	96	8060
KuMQuat-2^1-L5$_s$	MQ-2^1-L5$_{sk}(x)$	320	6	248	22	14	8	12	11	40	72	11728
KuMQuat-2^1-L5$_f$	MQ-2^1-L5$_{sk}(x)$	320	8	247	32	0	32	8	8	40	72	13396
KuMQuat-2^8-L5$_s$	MQ-2^8-L5$_{sk}(x)$	768	6	244	22	14	8	12	11	96	128	12823
KuMQuat-2^8-L5$_f$	MQ-2^8-L5$_{sk}(x)$	768	8	247	32	0	32	8	8	96	128	15092

Table 13: Signinig Time (ms), Verification Time (ms) and Signature Size (bytes) of the KuMQuat optimized implementation.

Scheme	Runtime in ms			Size in Bytes		
	Keygen	Sign	Verify	sk	pk	Signature
KuMQuat-2^1-L1_s	0.172	4.305	4.107	19	35	2555
KuMQuat-2^1-L1_f	0.173	0.539	0.736	19	35	3028
KuMQuat-2^8-L1_s	0.174	3.599	4.053	48	64	2890
KuMQuat-2^8-L1_f	0.172	0.400	0.623	48	64	3588
KuMQuat-2^1-L3_s	0.545	15.601	26.076	28	52	6404
KuMQuat-2^1-L3_f	0.545	2.316	4.724	28	52	7436
KuMQuat-2^8-L3_s	0.163	14.986	25.366	72	96	7180
KuMQuat-2^8-L3_f	0.163	1.963	2.801	72	96	8060
KuMQuat-2^1-L5_s	1.606	24.541	31.564	40	72	11728
KuMQuat-2^1-L5_f	1.606	4.934	6.336	40	72	13396
KuMQuat-2^8-L5_s	0.424	21.062	26.440	96	128	12823
KuMQuat-2^8-L5_f	0.424	2.529	3.443	96	128	15092