Concretely Efficient Blind Signatures Based on VOLE-in-the-Head Proofs and the MAYO Trapdoor

Carsten Baum[1], Marvin Beckmann[1], Ward Beullens[2], Shibam Mukherjee[3,4], Christian Rechberger[3,5]

Denmark Technical University[1], IBM, Zürich[2], Graz University of Technology[3], Know Center, Graz[4], TACEO, Graz[5]

Abstract

Blind signatures (Chaum, CRYPTO 82) are important building blocks in many privacy-preserving applications, such as anonymous credentials or e-cash schemes. Recent years saw a strong interest in building Blind signatures from post-quantum assumptions, primarily from lattices. While performance has improved, no construction has reached practical efficiency in terms of computation and communication. The state of the art requires at least 20 KB size of communication for each showing of a lattice-based Blind signature to a verifier, and more than 100 ms in prover time.

In this work, we propose an alternative direction with a plausibly post-quantum Blind signature scheme called PoMFRIT. It builds on top of the VOLE-in-the-head Zero-Knowledge proof system (Baum et al. CRYPTO 2023), which we combine with the MAYO digital signature scheme (Beullens, SAC 2021). We implement multiple versions of PoMFRIT to demonstrate security and performance trade-offs, and provide detailed benchmarks of our constructions. Signature issuance requires 0.45 KB communication for Blind signatures of size 6.7 KB. Showing a Blind signature can be done in < 76 ms even for a conservative construction with 128 bit security. As a building block for our Blind signature scheme, we implement the first VOLE-in-the-head proof for hash functions in the SHA-3 family, which we consider of independent interest.

1 Introduction

A digital signature scheme allows a signer to sign a message μ using their private signing key, such that a verifier can later authenticate the signature σ using the signer's public key. This requires the signer to know μ, which is often unsuitable for privacy-preserving applications. A Blind signature scheme [18] introduces an additional party, namely a user, who "owns" the message μ and sends a blinded version of it to the signer. The signer then returns information to the user, enabling it to generate a signature σ on the message μ to the verifier. The signature σ must verify under the signer's public key, and the user must only be able to produce signatures on messages which have been signed by the signer (who, nevertheless, did not actually learn them). Blind signatures find their use-cases in applications like e-voting, privacy preserving e-cash, digital identity, surveys, among others.

Classical constructions of Blind signatures have been known for some time, starting from the RSA-based construction of Chaum [18] or Schnorr signatures [42]. These are, of course, not secure against quantum attackers. With recent increased interest into privacy-preserving (anonymous) credential schemes, it is necessary to build efficient post-quantum alternatives which can be deployed and remain secure in the quantum age. However, only recently, some lattice-based proposals showcased practicality with signatures of around 20KB. They usually instantiate a version of Fischlin's round optimal Blind signatures [24]. Except for [3, 38], no implementations of lattice-based (or any other post-quantum) Blind signatures have so far been provided. Worse, for other post-quantum assumptions, proposed Blind signature schemes mostly generalize Schnorr's Blind signatures to the post-quantum setting. However, Schnorr-based Blind signatures have been shown to only be secure for a bounded parallel signing sessions [10, 33] and are therefore of limited practical use.

1.1 Our contributions

In this work, we build and implement a plausibly post-quantum Blind signature scheme without lattice assumptions, outperforming lattice-based constructions in communication and computation. Our approach generally follows Fischlin's paradigm [24], although we make modifications towards more practical efficiency. To make it easier to describe our approach, we quickly recap [24] now: Let $\mathsf{SIG} := (\mathsf{Setup}, \mathsf{KG}, \mathsf{Sig}, \mathsf{Ver})$ be the algorithms of a signature scheme. In addition, we assume the existence of a commitment scheme $\mathsf{COM} := (\mathsf{Setup}, \mathsf{Com})$ and a Non-interactive Zero-Knowledge (NIZK) proof system $\mathsf{ZK} := (\mathsf{Setup}, \mathsf{Prove}, \mathsf{Ver})$.

During key generation of the Blind signature algorithm, the

signer generates a signature key pair $(\mathtt{sk}, \mathtt{vk})$ and $\mathtt{vk}$ becomes the verification key of the Blind signature scheme, known to the user and the verifier. In the signing process, the user samples randomness r and produces a commitment $c = \mathsf{COM.Com}(\mu, r)$ to his message μ, and sends the commitment to the signer. The signer returns a signature σ on c that the user can verify using $\mathtt{vk}$. Finally, the user creates the Blind signature by proving knowledge of a commitment c that can be opened to μ, r and a signature σ on c that verifies under $\mathtt{vk}$ using the NIZK proof system ZK. To verify this proof, the verifier will only use $\mu, \mathtt{vk}$. More formally:

BSig.KG(): Signer runs and outputs $(\mathtt{sk}, \mathtt{vk}) \leftarrow \mathsf{SIG.KG}()$.

BSig.Sig($\mathtt{sk}, \mu$):

1. Sample r and compute $c = \mathsf{COM.Com}(\mu, r)$. Then send c to the signer.
2. Signer computes $\sigma \leftarrow \mathsf{SIG.Sig}(\mathtt{sk}, c)$ and sends it to the user.
3. User calls ZK.Prove to create a proof of knowledge π of (r, σ) such that $\mathsf{SIG.Ver}(\mathtt{vk}, \mathsf{COM.Com}(\mu, r), \sigma) = 1$.
4. User outputs μ, π to verifier.

BSig.Ver($\mathtt{vk}, \mu, \pi$): Run ZK.Ver to check if π is a valid proof of knowledge of (r, σ) such that $\mathsf{SIG.Ver}(\mathtt{vk}, \mathsf{COM.Com}(\mu, r), \sigma) = 1$. Output what ZK.Ver outputs.

Our main idea. We instantiate this paradigm using the MAYO signature scheme [11, 13] as well as the VOLE-in-the-head (VOLEitH) NIZK proof system [6, 8]. We exploit that verifying a solution to a multivariate quadratic system such as in the MAYO signature scheme is highly efficient using VOLEitH both in terms of computation and communication, as demonstrated first in [6]. However, the MAYO signature follows a hash-and-sign paradigm where the signed message $\mathtt{m}$ (which is c in our setting) is first hashed using SHAKE256 while the actual signature is a preimage of $\mathsf{SHAKE256}(\mathtt{m})$. Since we instantiate the commitment function Com with SHAKE256, the Zero-Knowledge (ZK) proof needs to prove a statement involving two evaluations of SHAKE256, one for the commitment and one for the MAYO signature verification. A straightforward application of ZK proof systems to prove two evaluations of SHAKE256 would be rather costly, due to the large number of rounds (24) and the large state size (1600 bits) of the underlying Keccak-f[1600] permutation.

Optimizing Hash function evaluation. We explore two solutions to this problem. First, we exploit that VOLEitH supports proving constraints of higher degree than 2 without substantial increase in proof size (and only modest increase in prover time). We optimize the proof of correct Keccak-f[1600] evaluation on a secret input state, leading to a 75% reduction in proof size for such circuits. In addition, we explore the use of an alternative hash function based on the Rain One-Way Function (OWF) [21]. As Rain shares similarities with the AES block cipher and recent works have explored how to evaluate it efficiently in VOLEitH [6, 8], those optimizations can be applied in our context. Compared to our optimized proof of SHAKE256 evaluation, the use of RainHash reduces the proof size further by approximately 50%.

Avoiding Hash function calls in the ZK proof. The main purpose of the commitment scheme COM is to hide the message μ from the signer. An alternative approach is to additively mask the hash of the message $c = \mathsf{SHAKE256}(\mu) \oplus r$ with some randomness r. If the user has control over r, then the Blind signature would not be secure, since the "commitment" c could be opened to any message μ', allowing the user to produce signatures for arbitrary messages given a signature for a single c. To solve this problem we leverage the fact that VOLEitH proof systems can be split into two parts. The first part outputs π_1 which is a binding and hiding commitment to some randomness r, known to the prover, and the second part takes the witness as input, and completes the proof of some statement involving the witness and the randomness r. The user first runs the first part of the proof, and uses $c = \mathsf{SHAKE256}(\mu, \pi_1) \oplus r$ as commitment to μ. Since μ and π_1 are public, the hash $h = \mathsf{SHAKE256}(\mu, \pi_1)$ can be computed in the clear. So, proving that c opens to μ just boils down to proving that $c = h \oplus r$, which is extremely cheap to prove in a VOLEitH proof system.

In addition, we can also avoid the use of SHAKE256 inside MAYO, by using a version of MAYO where the preimage is directly computed on the message $c = h \oplus r$ and not $\mathsf{SHAKE256}(\mathtt{c})$. This requires a stronger, but justifiable, security assumption on MAYO.

Implementation and Benchmarks. We implement two Fischlin-like Blind signatures using SHAKE256 and SHAKE256 with optimized evaluation. Additionally, we implement our optimized scheme that does not evaluate any hash function inside the NIZK, significantly improving the computation time and signature size. Our implementation builds upon the publicly available NIST PQ DSA Round 2 submissions of FAEST[1] and MAYO[2]. We implement modifications to MAYO and SHAKE256, with the NIZK proofs implemented in C++, and integrate them in Rust frameworks via foreign function interfaces. We implement and benchmark signatures with security levels equivalent to NIST levels 1, 3, and 5. For each level, we propose a small (s) and fast (f) variant optimized for proof size and prover time, respectively. We also implement RainHash inside the NIZK and construct an additional

[1] https://github.com/faest-sign/faest-avx
[2] https://github.com/PQCMayo/MAYO-C

Fischlin-like Blind signature for NIST L1 security, showcasing the performance benefit compared to constructions using SHAKE256.

1.2 Technical Overview

MAYO, on a high level, is a trapdoor OWF used in a hash-and-sign signature scheme. The public key $\mathtt{vk}$ describes a multivariate quadratic map $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, while the signing key $\mathtt{sk}$ allows to sample random preimages $\mathbf{s} \in \mathbb{F}_q^n$ from $\mathcal{T}^{-1}(\mathbf{c})$. Simply put, a signature on a message μ is a vector $\mathbf{s}$ such that

$$\mathcal{T}(\mathbf{s}) = \mathsf{SHAKE256}(\mu)$$

which can easily be verified using $\mathtt{vk}$[3]. Following an observation in [6], we note that the proof of correct evaluation of $\mathcal{T}(\mathbf{s})$ inside VOLEitH is communication- and computation-efficient due to very low multiplicative depth of the circuit and special properties of the proof system.

Optimizations to SHAKE256 evaluation inside VOLEitH. We reduce the cost of proving correct evaluations of SHAKE256 by carefully optimizing how its circuit is evaluated inside the VOLEitH proof system. The naive approach is to commit to the Keccak states $\mathtt{st}_0, \mathtt{st}_1, \ldots, \mathtt{st}_{24}$ after every round, and prove that all the states are consistent, i.e. $\mathsf{Round}(\mathtt{st}_i) = \mathtt{st}_{i+1}$, where Round is the Keccak-f[1600] round function. We observe that the multiplicative degree of the Keccak round function is only 2, and that the inverse only has degree 3. This allows us to improve the proof size by committing only every sixth Keccak states $\mathtt{st}_0, \mathtt{st}_6, \ldots, \mathtt{st}_{24}$. Then we prove that these states are consistent, by proving that $\mathsf{Round}^4(\mathtt{st}_i) = \mathsf{Round}^{-2}(\mathtt{st}_{i+6})$, which is a polynomial constraint of degree 16. This gives a reduction in proof size by almost a factor 4 (see Table 2).

We also investigate the use of the recently proposed Rain OWF, when modified as a hash function, to replace SHAKE256 both as commitment and for hash-and-sign. This leads to another substantial decrease in proof size. However, PoMFRIT can be made even more efficient if proving statements about hash functions can be avoided altogether.

More efficiently committing to μ. In the scheme described above, c and r are secret input of the ZK proof and demonstrate (among other things) that $c = \mathsf{Com}(\mu, r)$. We observe that computing c inside the ZK proof is actually not necessary if one generates the NIZKs using VOLEitH: VOLEitH is a two-stage proof system where, in the first stage, the prover generates a proof π_1 which contains linearly homomorphic commitments to random values r. In the second proof stage, the prover commits to the actual input by derandomizing commitments to r, and proving statements about committed values using the homomorphic property of the commitment scheme.

We use this as follows: Instead of fully generating a ZK proof after the user obtains σ from the signer, the user will start the VOLEitH ZK proof (which is the Blind signature) before interacting with the signer. The user generates π_1 committing to the randomness $r = (r_1, r_2)$. For a message μ and VOLEitH-committed randomness r_1, we then use $c = r_1 \oplus H(\mu, \pi_1)$ as the "commitment" which is signed by the signer. This works because r_1 is the output of a random oracle and unpredictable to the signer, even given π_1. Upon receiving σ on c from the signer, the user can then finish the proof by sending $r_2 \oplus \sigma$, i.e. committing to σ using r_2 (the remaining randomness) and showing that the committed σ is a valid signature on $r_1 \oplus H(\mu, \pi_1)$. Note that $H(\mu, \pi_1)$ is a public value that the verifier can compute based on μ and the NIZK proof.

Hash-free MAYO. To verify a MAYO signature, the verifier checks that $\mathcal{T}(\mathbf{s}) = \mathsf{SHAKE256}(\mu)$ where $\mathcal{T}$ is determined by $\mathtt{vk}$. SHAKE256 is used to hash to the codomain of $\mathcal{T}$, to make it possible to sign messages of arbitrary length. In our setting, if we match the length of the commitments $c = h \oplus r$ with the size of the codomain, we can avoid using the hash, and simply sample preimages for the commitment directly, i.e. $\mathcal{T}(\mathbf{s}) = c = h \oplus r$. The resulting Blind signature can be proven secure based on the conjectured one-more preimage resistance property of the MAYO trapdoor $\mathcal{T}$ (see Section 3).

1.3 Related Work

We now compare with the state-of-the-art of Blind signatures from lattices and multivariate quadratic polynomials, with a focus on efficiency and security.

Lattices. While there were many early works on lattice-based Blind signatures following Schnorr's paradigm, they all relied on a flawed security proof as identified by Hauck et al. [29]. They present the first provably secure construction, which however has signatures of size 7.7MB

For simplicity, we have summarized the state-of-the-art of all recent lattice-based Blind signature schemes in Table 1. The only implementations are due to Lyubashevsky et al. [38] and Argo et al. [3]. [38] relies on a non-standard lattice assumption. Even using Fischlin's generic construction and regular MAYO signatures we improve around 50% on their showing runtime and issue Blind signatures of comparable size. Using a non-standard assumption for MAYO, we achieve < 45 ms of showing time (improving $\approx 2\times$) and ≤ 7 KB of Blind signature size (improving $\approx 4\times$). Our construction based on regular MAYO has $5\times$ faster showing time than [3] and a $\approx 2.5\times$ improvement in Blind signature size.

[3] Note that in the MAYO scheme some random salt is included in the hash. This is only as a countermeasure against side-channel and fault injection attacks, and is not necessary for the security of the scheme. So we can omit the salt to simplify the design of our Blind signature scheme.

Scheme	Assumptions	Signing (in KB)	Showing (in KB)	Runtime Showing (in ms)	Security (in bits)
Agrawal et al. [2]	One-More-ISIS	1.4	45	–	109
del Pino[a] & Katsumata [19]	MSIS/MLWE/NTRU	851	102.6	–	128
Lyubashevsky[b] et al. [37]	MSIS/MLWE	$\gg 1000$	150	–	128
Bootle et al. [15]	ISIS-f/NTRU	> 100	> 100	–	128
Lyubashevsky[c]et al. [38]	ISIS-f/NTRU	1.3	29	< 200	128
Argo et al. [3] (implementing [30])	MSIS/MLWE	45	79.6	< 500	128
Beullens[d] et al. [14]	MSIS/MLWE/NTRU	60	22	–	128
Jeudy and Sanders [31]	MSIS/MLWE	60	41	–	128
Baldimtsi[e] et al. [5]	One-More-ISIS	1	68	–	128
SHAKE256-deg16+MAYO-$128_{s/f}$	UOV/WMQ	0.486	24.3/41.6	178.1/75.6	128
RainHash+MAYO-$128_{s/f}$	UOV/WMQ	0.486	16.5/27.8	114.4/49.8	128
One-More-MAYO-$128_{s/f}$	UOV/One-More-WMQ	0.469	6.7/10.4	42/40	128

[a] May be improved upon using more recent lattice NIZKs. [b] Bounded number of signatures, 1.3MB public key. [c] Implements a version of [15].
[d] Uses a generic NIZK scheme. [e] Non-interactive construction with pseudorandom messages.

Table 1: Efficiency of various Blind signatures using lattice-based assumptions compared to ours (SHAKE256-deg16+MAYO-$128_{s/f}$, RainHash+MAYO-$128_{s/f}$, One-More-MAYO-$128_{s/f}$). A more detailed version of all our constructions, including a finer distinction between the short (s) and fast (f) version, can be found in Table 3. Here, and in previous work, "signing" denotes the issuer's workload and "showing" refers to the user's workload when running the NIZK.

Multivariate Quadratic Polynomials. In 2017, Petzoldt et al. [41] constructed a post-quantum secure Blind signature based on multivariate trapdoor and ZK proof system. However, their construction was shown to be insecure [12]. As an improvement, [12] also suggested to switch to VOLEitH proof systems [6] for better signing and verification performance with reduced signature size.

Recently and concurrent to us, Bouillaguet et al. [16] use a generalized commit-append-and-prove paradigm to model the NIZK proofs necessary for Blind signatures based on MPC-in-the-head. In comparison, we use a different generalization which directly applies to VOLEitH. As we do, the authors investigate multiple hash functions for the signatures and commitments, and investigate signatures based on UOV and the code-based WAVE while we only focus on MAYO (a specific UOV signature design). While they achieve showing sizes comparable to ours, they make use of an expensive generic Succinct Non-Interactive Argument of Knowledge (SNARK) in the signing protocol, which makes signing slow and bandwidth-heavy. No implementation is available, but the authors estimate that producing a signature can take anywhere between 37 seconds and multiple days, with the Blind signature sizes between 300 KB and 2 MB, depending on parameters and SNARK. In contrast, Table 3 shows that PoMFRIT signing using regular MAYO, including all steps, requires < 80 ms of runtime with ≈ 30 KB of communication for comparable security and assumptions.

2 Preliminaries

Notation. We use λ to denote the computational security parameter and write vectors using bold lower-case characters such as $\mathbf{r}$, while bold upper-case letters such as $\mathbf{M}$ denote matrices. $[n]$ denotes the set $\{1,\dots,n\}$. We write $\xleftarrow{\$} S$ to sample uniformly at random from a finite set S, while $\leftarrow A()$ denotes the output of a possibly randomized algorithm A. Let $\mathsf{negl}(n)$ denote any positive function that is negligible in n, i.e. $< 1/|p(n)|$ for any polynomial p and large enough n.

2.1 Signatures

A signature scheme consists of the following algorithms:

- $\mathsf{KG}(1^\lambda) \to (\mathtt{sk},\mathtt{pk})$: Takes the security parameter as input and outputs a secret key $\mathtt{sk}$ and a public key $\mathtt{pk}$.
- $\mathsf{Sig}(\mathtt{sk},\mu) \to \mathsf{sig}$: Takes a signing key $\mathtt{sk}$ and a message $\mu \in \{0,1\}^*$ as input and outputs a signature sig.
- $\mathsf{Ver}(\mathtt{pk},\mu,\mathsf{sig}) \to \{0,1\}$: A verification algorithm that takes as input a public key $\mathtt{pk}$, a message μ and a signature sig and outputs a bit to indicate whether the signature is deemed valid (1) or invalid (0).

All the algorithms take the security parameter λ as input (sometimes implicitly). We require that a signature satisfies correctness (honestly generated signatures are always valid) and existential unforgeability, which says that if an attacker obtains Q signatures for an honestly generated $\mathtt{pk}$ for which it doesn't know the signing key (and where it could choose the messages itself), then it cannot produce a signature for

non-queried other message.

Definition 2.1 (Correctness)**.** A signature $(\mathsf{KG}, \mathsf{Sig}, \mathsf{Ver})$ is correct if there exists a negligible function negl, such that for any message μ we have

$$\Pr\left[\mathsf{Ver}(\mathtt{pk}, \mu, \mathsf{sig}) = 1 \;\middle|\; \begin{array}{c}(\mathtt{sk}, \mathtt{pk}) \leftarrow \mathsf{KG}(1^\lambda), \\ \mathsf{sig} \leftarrow \mathsf{Sig}(\mathtt{sk}, \mu)\end{array}\right] \geq 1 - \mathsf{negl}(\lambda).$$

Definition 2.2 (EUF-CMA)**.** A signature $(\mathsf{KG}, \mathsf{Sig}, \mathsf{Ver})$ is existentially unforgeable if for every polynomial-time adversary $\mathcal{A}$ that makes at most Q queries to the signing oracle there exists a negligible function negl such that

$$\Pr\left[\begin{array}{c}(\mu, \cdot) \notin L, \\ \mathsf{Ver}(\mathtt{pk}, \mu, \mathsf{sig}) = 1\end{array} \;\middle|\; \begin{array}{c}(\mathtt{sk}, \mathtt{pk}) \leftarrow \mathsf{KG}(1^\lambda), \\ (\mu, \mathsf{sig}) \leftarrow \mathcal{A}^{\mathsf{Sig}(\mathtt{sk}, \cdot))}(\mathtt{pk})\end{array}\right] \leq \mathsf{negl}(\lambda).$$

where $L = (\mu_i, \mathsf{sig}_i)_{i \in [Q]}$ is the list of all queries and responses of $\mathcal{A}$ to the signing oracle.

2.2 Blind Signatures

We provide the correctness and security definitions of Blind signatures [18]. A round-optimal Blind signature scheme consists of the following five algorithms:

- $\mathsf{KG}(1^\lambda) \rightarrow (\mathtt{sk}, \mathtt{pk})$: Takes the security parameter as input and outputs a secret key sk and a public key pk.
- $\mathsf{Sig}_1(\mathtt{pk}, \mu) \rightarrow (\rho_1, S_{\mathcal{U}})$: This is the first part of a signing protocol between a user $\mathcal{U}$ and a signer $\mathcal{S}$, where $\mathcal{U}$ uses a public key and a message $\mu \in \{0,1\}^*$, to compute a state $S_{\mathcal{U}}$ and a first message ρ_1, which they send to $\mathcal{S}$.
- $\mathsf{Sig}_2(\mathtt{sk}, \rho_1) \rightarrow \rho_2$: The second part of the interaction, where $\mathcal{S}$ uses their signing key sk and a first message ρ_1 and outputs a second message ρ_2, which they send to $\mathcal{U}$.
- $\mathsf{Sig}_3(\rho_2, S_{\mathcal{U}}) \rightarrow \mathsf{sig}$: The last part of the interaction, where $\mathcal{U}$ uses ρ_2 and $S_{\mathcal{U}}$ to derive a signature sig or $\perp$ if the signing protocol failed.
- $\mathsf{Ver}(\mathtt{pk}, \mu, \mathsf{sig}) \rightarrow \{0,1\}$: A verification algorithm that takes as input a public key pk, a message μ and a signature sig and outputs a bit to indicate whether the signature is deemed valid (1) or invalid (0).

All the algorithms take the security parameter λ as input (sometimes implicitly). We require Blind signatures to satisfy three properties: correctness, which says that honestly generated signatures should verify with probability close to 1, blindness, which says that the signer cannot link signatures to the interaction with which they were created, and existential unforgeability, which says that if a user is allowed to execute the signing protocol Q times with chosen messages it cannot obtain valid signatures for any other messages.

Definition 2.3 (Correctness)**.** A Blind signature $(\mathsf{KG}, (\mathsf{Sig}_i)_{i \in [3]}, \mathsf{Ver})$ is correct if there exists a negligible function negl, such that for any message μ we have

$$\Pr\left[\mathsf{Ver}(\mathtt{pk}, \mu, \mathsf{sig}) = 1 \;\middle|\; \begin{array}{c}(\mathtt{sk}, \mathtt{pk}) \leftarrow \mathsf{KG}(1^\lambda), \\ (\rho_1, S_{\mathcal{U}}) \leftarrow \mathsf{Sig}_1(\mathtt{pk}, \mu), \\ \rho_2 \leftarrow \mathsf{Sig}_2(\mathtt{sk}, \rho_1), \\ \mathsf{sig} \leftarrow \mathsf{Sig}_3(\rho_2, S_{\mathcal{U}})\end{array}\right]$$

is 1 except with probability $\mathsf{negl}(\lambda)$.

Definition 2.4 (Blindness)**.** A Blind signature $(\mathsf{KG}, (\mathsf{Sig}_i)_{i \in [3]}, \mathsf{Ver})$ has blindness if for every polynomial-time three-part stateful adversary $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2, \mathcal{A}_3)$ there exists a negligible function negl such that for any two messages μ_0, μ_1 we have

$$\left|\Pr\left[\mathcal{A}_3(\mathsf{sig}^0, \mathsf{sig}^1) = b \;\middle|\; \begin{array}{c}\mathtt{pk} \leftarrow \mathcal{A}_1(1^\lambda), b \xleftarrow{\$} \{0,1\}, \\ (\rho_1^0, S_{\mathcal{U}}^0) \leftarrow \mathsf{Sig}_1(\mathtt{pk}, \mu_0), \\ (\rho_1^1, S_{\mathcal{U}}^1) \leftarrow \mathsf{Sig}_1(\mathtt{pk}, \mu_1), \\ \rho_2^b, \rho_2^{1-b} \leftarrow \mathcal{A}_2(\rho_1^b, \rho_1^{1-b}), \\ \mathsf{sig}^0 \leftarrow \mathsf{Sig}_3(\rho_2^0, S_{\mathcal{U}}^0), \\ \mathsf{sig}^1 \leftarrow \mathsf{Sig}_3(\rho_2^1, S_{\mathcal{U}}^1), \\ \text{If } \perp \in \{\mathsf{sig}^0, \mathsf{sig}^1\} \\ \text{then } (\mathsf{sig}^0, \mathsf{sig}^1) \leftarrow (\perp, \perp)\end{array}\right] - \frac{1}{2}\right|$$

is bounded by $\mathsf{negl}(\lambda)$.

Definition 2.5 (One-more unforgeability)**.** A Blind signature $(\mathsf{KG}, (\mathsf{Sig}_i)_{i \in [3]}, \mathsf{Ver})$ is one-more unforgeable if for every polynomial-time adversary $\mathcal{A}$ making at most Q queries (Q is a function of λ, bounded by a polynomial) to the signing oracle Sig_2 there exists a negligible function negl such that

$$\Pr\left[\begin{array}{c}\forall 1 \leq i < j \leq Q+1 : \\ (\mu_i \neq \mu_j), \\ \forall 1 \leq i \leq Q+1 : \\ (\mathsf{Ver}(\mathtt{pk}, \mu_i, \mathsf{sig}_i) = 1)\end{array} \;\middle|\; \begin{array}{c}(\mathtt{sk}, \mathtt{pk}) \leftarrow \mathsf{KG}(1^\lambda), \\ \{(\mu_i, \mathsf{sig}_i)\}_{i \in [Q+1]} \leftarrow \\ \mathcal{A}^{\mathsf{Sig}_2(\mathtt{sk}, \cdot))}(\mathtt{pk})\end{array}\right] \leq \mathsf{negl}(\lambda).$$

2.3 Commitments

A commitment scheme consists of the following algorithms:

- $\mathsf{Setup}(1^\lambda) \rightarrow \mathtt{cpk}$: Takes the security parameter as input and outputs a commitment public key cpk.
- $\mathsf{Com}(\mathtt{cpk}, \mu, r) \rightarrow c$: Takes a commitment public key cpk, a message $\mu \in \{0,1\}^*$ and randomness $r \in \{0,1\}^\lambda$ as input and outputs a commitment $c \in \{0,1\}^*$.

We say that a commitment scheme is correct if for any honestly generated cpk, the algorithm Com is deterministic and always terminates except with negligible probability. In addition, we require that it is binding and hiding:

Definition 2.6 (Binding)**.** A commitment scheme $(\mathsf{Setup}, \mathsf{Com})$ is binding if there exists a negligible

function negl, such that for any PPT algorithm $\mathcal{A}$

$$\Pr\left[\begin{array}{c} c_1 = c_2, \\ c_1 \neq \bot, \mu_1 \neq \mu_2 \end{array} \middle| \begin{array}{c} \mathtt{cpk} \leftarrow \mathsf{Setup}(1^\lambda), \\ (\mu_1, \mu_2, r_1, r_2) \leftarrow \mathcal{A}(\mathtt{cpk}), \\ c_1 \leftarrow \mathsf{Com}(\mathtt{cpk}, \mu_1, r_1), \\ c_2 \leftarrow \mathsf{Com}(\mathtt{cpk}, \mu_2, r_2) \end{array}\right] \leq \mathsf{negl}(\lambda).$$

Definition 2.7 (Hiding)**.** A commitment scheme $(\mathsf{Setup}, \mathsf{Com})$ is hiding if there exists a negligible function negl, such that for any PPT algorithm $\mathcal{A}$ we have that

$$\left|\Pr\left[b = \mathcal{A}(c, \mathtt{st}) \middle| \begin{array}{c} \mathtt{cpk} \leftarrow \mathsf{Setup}(1^\lambda), \\ (\mu_0, \mu_1, \mathtt{st}) \leftarrow \mathcal{A}(\mathtt{cpk}), \\ b \xleftarrow{\$} \{0,1\}, r \xleftarrow{\$} \{0,1\}^\lambda, \\ c \leftarrow \mathsf{Com}(\mathtt{cpk}, \mu_b, r)) \end{array}\right] - \frac{1}{2}\right|$$

is bounded by $\mathsf{negl}(\lambda)$.

2.4 Hash Functions

To implement the commitment and Hash-and-Sign paradigm we use a cryptographic hash function (which will be modeled as a random oracle). In this work, we use the SHAKE256 extended output function as specified in the SHA-3 NIST specification [23].

SHAKE256 splits the input into blocks of size $r = 136$ bytes each, and pads the input accordingly. Then, the blocks are absorbed consecutively using a sponge construction. The first input block, concatenated with an initially empty vector of length c, is permuted using a round function (permutation) of input/output length $r+c$. We call this $r+c$-byte vector the state. The first r bytes of the function's output are XOR-ed with the next input block and this process is repeated until all input blocks are processed. When the whole input is processed, the output is generated: the first r bytes of the $r+c$-byte state obtained after processing the input blocks denote the first r output bytes. If more output is needed, the permutation is applied again to the whole $r+c$-byte state and the first r bytes of the outcome are considered as next r bytes of the output. The process can then be repeated as required. For SHAKE256, this algorithm is described by the Keccak$[512]$ sponge function using the Keccak-f$[1600,24]$ permutation. The sponge construction itself uses only XOR-operations, i.e. simple $\mathbb{F}_2$ additions.

The Keccak-f$[1600,24]$ permutation, however, is non-linear and processes 1600 bits state using 24 almost identical permutation rounds. Each permutation round consists of three layers: (1) θ which is an invertible linear map, (2) $\rho\&\pi$ which is a permutation, (3) χ which is an invertible degree 2 map. Therefore each round of Keccak-f$[1600,24]$ is an invertible polynomial function of multiplicative degree 2. The inverse map is known to have degree 3 [22].

In Section 7, we will introduce our new proposed RainHash hash function, which is based on the Rain OWF. Unlike SHAKE256, it is primarily designed to be efficiently evaluated in MPC/VOLEitH-based ZK proof systems.

3 MAYO and one-more preimage resistance

In this section, we will quickly recap how the MAYO signature works. We will also put forward a strengthening of one of the security assumptions underlying MAYO, which will allow us to build a more efficient Blind signature scheme.

A sequence of m multivariate quadratic polynomials in n variables $\mathcal{T} = (p_1, \ldots, p_m)$ is called a multivariate quadratic map, and we identify it with the function

$$\begin{aligned} \mathcal{T} : \mathbb{F}_q^n &\to \mathbb{F}_q^m \\ \mathbf{x} &\mapsto p_1(\mathbf{x}), \ldots, p_m(\mathbf{x}) \end{aligned}$$

given by the evaluation of the component polynomials. For any multivariate quadratic polynomial $p(\mathbf{x})$ we define the corresponding polar form as $p'(\mathbf{x},\mathbf{y}) := p(\mathbf{x}+\mathbf{y}) - p(\mathbf{x}) - p(\mathbf{y}) + p(\mathbf{0})$, which is a symmetric bilinear form. We denote by $\mathcal{T}'(\mathbf{x},\mathbf{y}) : \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q^m$ the function of evaluating the polar forms associated to $\mathcal{T}$, which satisfies

$$\mathcal{T}'(\mathbf{x},\mathbf{y}) = \mathcal{T}(\mathbf{x}+\mathbf{y}) - \mathcal{T}(\mathbf{x}) - \mathcal{T}(\mathbf{y}) + \mathcal{T}(\mathbf{0}).$$

3.1 Oil and Vinegar trapdoors

The Oil and Vinegar digital signature scheme relies on a simple but powerful procedure for sampling preimages for a multivariate quadratic map $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^m$, given knowledge of a linear subspace $O \subset \mathbb{F}_q^n$ of dimension at least m on which $\mathcal{T}$ evaluates to zero. Given a target $\mathbf{t} \in \mathbb{F}_q^m$, the procedure first samples a vector $\mathbf{v} \in \mathbb{F}_q^n$ uniformly at random, and then solves for $\mathbf{o} \in O$ such that $\mathcal{T}(\mathbf{v}+\mathbf{o}) = \mathbf{t}$. The latter can be done efficiently because it reduces to solving a system of linear equations in $\mathbf{o}$ by using the polar form:

$$\mathcal{T}(\mathbf{v}+\mathbf{o}) = \underbrace{\mathcal{T}'(\mathbf{o},\mathbf{v})}_{\text{linear in } \mathbf{o}} + \underbrace{\mathcal{T}(\mathbf{o})}_{=\mathbf{0}} + \underbrace{\mathcal{T}(\mathbf{v}) - \mathcal{T}(\mathbf{0})}_{\text{fixed by choice of } \mathbf{v}} = \mathbf{t}. \quad (1)$$

For properly chosen parameters (m sufficiently large, and n sufficiently larger than $2m$), it is assumed hard to sample preimages for $\mathcal{T}$ without knowledge of O, so this yields a cryptographic trapdoor function.

3.2 The MAYO trapdoor

The Oil and Vinegar signature scheme uses an almost uniformly random subspace O [4], and a map $\mathcal{T}$ which is chosen uniformly at random among the maps vanishing on O. Representing such maps requires $O(m^3)$ coefficients, which unfortunately means that the Oil and Vinegar signature scheme

[4] In typical implementations, the subspace O is the row span of a matrix $(\mathbf{1}_m \; \mathbf{O})$, where the submatrix $\mathbf{O} \in \mathbb{F}_q^{m \times (n-m)}$ is chosen uniformly at random.

has large public key sizes. To solve this problem, the MAYO trapdoor was introduced [11], which is still a multivariate quadratic map which vanishes on a large hidden subspace, but which require fewer than $O(m^3)$ coefficients to represent, reducing the public key size.

The MAYO trapdoor, starts from a "base" multivariate quadratic map $\mathcal{T} : \mathbb{F}_q^n \to \mathbb{F}_q^m$ that vanishes on a subspace O. The only difference with an Oil and Vinegar public key is that in the Oil and Vinegar signature scheme we have $\dim(O) = m$, whereas in the case of MAYO the dimension of O is smaller. This means that the procedure from Section 3.1 no longer works, since the linear system (1) has more equations than variables, and is thus unlikely to have solutions. To increase the size of the subspace, MAYO "whips up" the base map $\mathcal{T}$ into a larger map $\mathcal{T}^* : \mathbb{F}_q^{kn} \to \mathbb{F}_q^m$ that has a factor k more variables, where k is a parameter of the scheme. This is done by letting the polynomials of $\mathcal{T}^*(\mathbf{x}_1, \ldots, \mathbf{x}_k)$ be linear combinations of $\mathcal{T}(\mathbf{x}_i)$ and $\mathcal{T}'(\mathbf{x}_i, \mathbf{x}_j)$ for $i, j \in \{1, \ldots, k\}$, where each $\mathbf{x}_i$ is a sequence of n variables. MAYO defines

$$\mathcal{T}^*(\mathbf{x}_1, \ldots, \mathbf{x}_k) := \sum_{i=1}^{k} \mathbf{E}_{i,i} \mathcal{T}(\mathbf{x}_i) + \sum_{i=1}^{k} \sum_{j=i+1}^{k} \mathbf{E}_{i,j} \mathcal{T}'(\mathbf{x}_i, \mathbf{x}_j),$$

where $\mathbf{E}_{i,i}$ and $\mathbf{E}_{i,j}$ are fixed m-by-m matrices whose rows determine what linear combination of $\mathcal{T}(\mathbf{x}_i)$ and $\mathcal{T}'(\mathbf{x}_i, \mathbf{x}_j)$ each polynomial of $\mathcal{T}^*$ consists of.

Since $\mathcal{T}$ vanishes on the subspace O, it follows that $\mathcal{T}^*$ vanishes on the space $O^k = \{(\mathbf{o}_1, \ldots, \mathbf{o}_k) \,|\, \mathbf{o}_1, \ldots, \mathbf{o}_k \in O\}$, which is a subspace of $\mathbb{F}_q^{nk}$ of dimension $k \dim(O)$. The parameters are chosen such that $k \dim(O) \geq m$, so that the Oil and Vinegar procedure from Section 3.1 can be used to efficiently sample preimages for $\mathcal{T}^*$.

The matrices $\mathbf{E}_{i,j}$ are chosen in such a way that all non-trivial linear combinations of them have full rank [5], since linear combinations with low rank lead to efficient forgery attacks [11].

3.3 The MAYO algorithms

Given the outline of the MAYO signature scheme above, we can summarize its public parameters $\mathbf{P}$ as follows: (i) $\mathbb{F}_q$, the field over which the equations are defined (ii) n, the number of variables (iii) m, the number of polynomials in the public key (iv) o, the dimension of the oil space O (v) k, a whipping parameter (vi) $\{\mathbf{E}_{i,j}\}_{i \leq j \in [k]^2}$, a set of public matrices in $\mathbb{F}_q^{m \times m}$.

We define 3 algorithms $\mathsf{MAYO} = (\mathsf{TrapGen}, \mathsf{Eval}, \mathsf{SPre})$, which all obtain $\mathbf{P}$ as implicit input, as follows:

- $\mathsf{TrapGen}() \to (\mathbf{O}, \mathcal{T})$:
  1. Sample $\mathbf{O} \in \mathbb{F}_q^{(n-o) \times o}$ uniformly at random.
  2. Sample m homogeneous quadratic polynomials $\mathcal{T} = (p_1, \ldots, p_m)$ in n variables, uniformly at random subject to the constraint that that they evaluate to zero on the row-space of $\begin{pmatrix} \mathbf{O} & \mathbf{1}_o \end{pmatrix}$.
  3. Output $(\mathtt{sk} = \mathbf{O}, \mathtt{pk} = \mathcal{T})$
- $\mathsf{Eval}(\mathcal{T}, \mathbf{s} = (\mathbf{s}_1, \ldots, \mathbf{s}_k)) \to \mathbb{F}_q^m$: Output

$$\mathcal{T}^*(\mathbf{s}) := \sum_{i=1}^{k} \mathbf{E}_{i,i} \mathcal{T}(\mathbf{s}_i) + \sum_{i=1}^{k} \sum_{j=i+1}^{k} \mathbf{E}_{i,j} \mathcal{T}'(\mathbf{s}_i, \mathbf{s}_j).$$

- $\mathsf{SPre}(\mathbf{O}, \mathbf{t} \in \mathbb{F}_q^m) \to \mathbf{s}$:
  0. Let $O = \mathsf{rowspan}(\begin{pmatrix} \mathbf{O} & \mathbf{1}_o \end{pmatrix})$. Set $\mathsf{ctr} := 0$.
  1. Sample $\mathbf{v} = (\mathbf{v}_1, \ldots, \mathbf{v}_k) \in \mathbb{F}_q^{n \times k}$ uniformly at random, deterministically based on $(\mathbf{O}, \mathsf{ctr})$.
  2. If the linear map $A_{\mathbf{v}} : O^k \to \mathbb{F}_q^m : \mathbf{o} \mapsto \mathcal{T}^*(\mathbf{v} + \mathbf{o})$ does not have rank m, increase ctr by 1 and go to step 1. (Note that $A_{\mathbf{v}}$ is indeed linear because $\mathcal{T}$ vanishes on O.)
  3. Sample a solution $\mathbf{o}$ to the linear system $\mathcal{T}^*(\mathbf{v} + \mathbf{o}) = \mathbf{t}$ uniformly at random, deterministically based on $(\mathbf{O}, \mathsf{ctr})$. Output $\mathbf{s} = \mathbf{v} + \mathbf{o}$.

Based on a cryptographic hash function, the MAYO signature scheme is constructed from the aforementioned 3 algorithms using the hash-and-sign paradigm. "One-more" assumptions are commonly used to construct Blind signatures, and we now define the one-more preimage resistance hardness assumption that we use for our Blind signature schemes (and that also underlies MAYO). In it, the adversary is given n random target elements $\mathbf{t}_i$ from $\mathbb{F}_q^m$ and can query the SPre algorithm up to q times. It is then asked to produce $q+1$ preimages for $q+1$ out of the n targets $\mathbf{t}_i$.

Definition 3.1 ((n,q)-one-more preimage resistance)**.** We define the advantage $\mathsf{Adv}^{n,q,\mathsf{T}}_{\mathsf{ompr}}$ of an oracle algorithm $\mathcal{A}$ against the (n,q)-one-more preimage resistance property of a trapdoor function $\mathsf{T} = (\mathsf{TrapGen}, \mathsf{SPre})$ as

$$\Pr\left[\begin{array}{c|c} \begin{array}{c} \forall i \neq j \in [q+1]^2 : \\ (I_i \neq I_j), \\ \forall i \in [q+1] : \\ (\mathsf{Eval}(\mathtt{pk}, \mathbf{s}_i) = \mathbf{t}_{I_i}), \\ \mathsf{SPre} \text{ is called} \\ \text{at most } q \text{ times} \end{array} & \begin{array}{c} (\mathtt{sk}, \mathtt{pk}) \leftarrow \mathsf{TrapGen}(), \\ \forall i \in [n] : \\ (\mathbf{t}_i \xleftarrow{\$} \mathsf{T.Codomain}), \\ \{\mathbf{s}_i, I_i\}_{i \in [q+1]} \leftarrow \\ \mathcal{A}^{\mathsf{SPre}(\cdot, \mathsf{td})}(\mathtt{pk}, \{\mathbf{t}_i\}_{i \in [n]}) \end{array} \end{array}\right]$$

Definition 3.1 is a generalization of the Whipped MQ Problem of MAYO [11, Definition 4], which is equivalent to $(1,0)$-one-more preimage resistance. We discuss it further in Appendix A.

[5] To ensure that non-trivial linear combinations have full rank these matrices are chosen to represent multiplication by $\mathbb{F}_q$-linearly independent elements of an extension field $\mathbb{F}_{q^m}$.

4 Non-Interactive Zero-Knowledge proofs

In this section, we describe the model of NIZKs that we use in this work. As we rely on the VOLEitH proof system in our implementation, we expose some of the properties of those proof systems to allow a more efficient construction. We describe our constructions in terms of these general definitions, as the same properties also hold for other NIZK proof systems such as those based on MPC-in-the-head. See Appendix B to see how VOLEitH proofs work and why our model is suitable.

Computational model. We model computation by applying a circuit C to an input $\mathbf{x} \in \mathbb{F}_q^{2\ell}$. We say that the input, or witness, is valid if $C(\mathbf{x}) = 0$ and invalid otherwise. C has 2ℓ input wires and a single output wire. The output may either be an element from $\mathbb{F}_{q^s}$ for any $s \geq 1$ or $\perp$.

C consists of wires and gates, where each gate has at least one input wire and exactly one output wire. The input wires of each gate are either the input wires of C, or output wires of other gates. The output of a gate may be input to multiple other gates, and the wires and gates form an acyclic directed graph. There are 4 types of gates in a circuit:

Lift gates take as input a wire value $x \in \mathbb{F}_{q^r}$ and output a wire value $\bar{x} \in \mathbb{F}_{q^s}$ where $r, s \in \mathbb{N}$, $r|s$ and $\bar{x}$ is the output of the canonical embedding of $\mathbb{F}_{q^r}$ into $\mathbb{F}_{q^s}$.

Linear gates apply public linear functions $\mathtt{lin}(\alpha, \beta, \mathbf{x}) = \langle \alpha, \mathbf{x} \rangle + \beta$ on the input wires $\mathbf{x} \in \mathbb{F}_{q^r}^h$. $\alpha \in \mathbb{F}_{q^r}^h, \beta \in \mathbb{F}_{q^r}$ are public inputs to the gate, while the output is a single element in $\mathbb{F}_{q^r}$. Note that h may be arbitrarily large.

Multiplication gates take as input a vector $\mathbf{x} \in \mathbb{F}_{q^r}^h$, compute the product of all input wires $\mathtt{mul}(\mathbf{x}) = \prod_{i \in [h]} x_i$ and output it as an element in $\mathbb{F}_{q^r}$. h can be larger than 2.

Assert gates take as input a wire value $x \in \mathbb{F}_{q^r}$ and output $\top$ if $x = 0$ and $\perp$ otherwise. Note that the output wire of an assert gate may therefore never be input to another gate.

We define the evaluation of C on input $\mathbf{x}$ as follows: $\mathbf{x}$ is initially assigned to the input wires of the circuit, and then consecutively all gates whose input wires are fully assigned are evaluated and the output of the gate function is assigned to the output wire(s). This process is repeated until a) the single output wire of the circuit has a value assigned to it; and b) all assert gates have been evaluated. If all assert gates have output $\top$ then $C(\mathbf{x})$ is equivalent to the value assigned to the output wire of C, otherwise $C(\mathbf{x}) = \perp$.

We denote with $\mathcal{C}$ a set of circuits, all with the input length 2ℓ, output element in $\mathbb{F}_{q^s} \cup \{\perp\}$ and defined over the field $\mathbb{F}_q$. We assume that $\mathcal{C}$ has a description size that is polynomial in the security parameter λ, even if $|\mathcal{C}|$ is exponential in λ.

Our NIZK model. We now define NIZK proof systems in the random oracle (RO) model. A NIZK proof system Π for the set of circuits $\mathcal{C}$ is a tuple $\Pi = (\mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ of PPT algorithms, consisting of two prover algorithms $\mathcal{P}_1, \mathcal{P}_2$ and the verifier $\mathcal{V}$ that have access to the random oracle H:

- $\mathcal{P}_1^{\mathsf{H}}(\mathcal{C}) \to (\pi_1, \mathbf{w}_\mathtt{r}, \mathtt{st})$: Takes the circuit set $\mathcal{C}$ and outputs a proof π_1, partial witness $\mathbf{w}_\mathtt{r} \in \mathbb{F}_q^\ell$ and state $\mathtt{st}$.
- $\mathcal{P}_2^{\mathsf{H}}(C, \mathbf{w}, \mathtt{st}) \to \pi_2$: Takes a circuit $C \in \mathcal{C}$, a witness $\mathbf{w}$ and a state $\mathtt{st}$, and outputs a proof π_2.
- $\mathcal{V}^{\mathsf{H}}(C, \pi_1, \pi_2) \to \{0,1\}$: Takes a circuit $C \in \mathcal{C}$, and two proofs π_1 and π_2, and outputs a bit to indicate whether the proofs are deemed valid (1) or invalid (0).

In the following, we drop the superscript H from $\mathcal{P}_1, \mathcal{P}_2, \mathcal{V}$ to simplify notation.

Definition 4.1 (Correctness)**.** A proof system $(\mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ for $\mathcal{C}$ in the RO model has correctness error $\varepsilon_{\mathsf{cor}}$ if for $C \in \mathcal{C}$ and for all $\mathbf{w} \in \mathbb{F}_q^\ell$,

$$\Pr\left[\mathcal{V}(C, \pi_1, \pi_2) = 1 \;\middle|\; \begin{array}{c} (\pi_1, \mathbf{w}_\mathtt{r}, \mathtt{st}) \leftarrow \mathcal{P}_1(\mathcal{C}), \\ \pi_2 \leftarrow \mathcal{P}_2(C, \mathbf{w}, \mathtt{st}), \\ C(\mathbf{w}_\mathtt{r}, \mathbf{w}) = 0 \end{array}\right] \geq 1 - \varepsilon_{\mathsf{cor}}(\lambda)$$

$(\mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ is correct if $\varepsilon_{\mathsf{cor}} = \mathsf{negl}$.

Note that we make the additional constraint that $C(\mathbf{w}_\mathtt{r}, \mathbf{w}) = 0$ during correctness. This is because $\mathbf{w}_\mathtt{r}$ is generated independently of C as it only depends on $\mathcal{C}$, so we allow to adaptively pick $\mathbf{w}$ based on $\mathbf{w}_\mathtt{r}, C$.

Definition 4.2 (Non-interactive Zero-Knowledge)**.** A simulator $\mathcal{S} = (\mathcal{S}_1, \mathcal{S}_2)$ for a proof system $\Pi = (\mathcal{P}_1, \mathcal{P}_2, \mathcal{V})$ for $\mathcal{C}$ is a pair of PPT algorithms with implicit input 1^λ. $\mathcal{S}_1$ on input $\mathcal{C}$ outputs $\pi_1, \mathtt{st}$ while $\mathcal{S}_2$ on input $C, \mathtt{st}$ outputs π_2. Let $\mathcal{A}$ be a stateful algorithm and let

$$\mathsf{Real}_{\mathcal{A}}(\lambda) = \Pr\left[b = 1 \,|\, b \leftarrow \mathcal{A}^{\mathsf{H}, O_{\mathcal{P}_1}, O_{\mathcal{P}_2}}(1^\lambda)\right]$$
$$\mathsf{Ideal}_{\mathcal{A}}(\lambda) = \Pr\left[b = 1 \,|\, b \leftarrow \mathcal{A}^{\mathsf{H}, O_{\mathcal{S}_1}, O_{\mathcal{S}_2}}(1^\lambda)\right]$$

The adversary has black-box access to H and to a pair of oracles $O_{\mathcal{P}_1}, O_{\mathcal{P}_2}$ or $O_{\mathcal{S}_1}, O_{\mathcal{S}_2}$. We define these oracles as follows:

- $O_{\mathcal{P}_1}(\mathcal{C})$ computes $(\pi_1, \mathbf{w}_\mathtt{r}, \mathtt{st}) \leftarrow \mathcal{P}_1^{\mathsf{H}}(\mathcal{C})$ and outputs $\pi_1, \mathbf{w}_\mathtt{r}$.
- $O_{\mathcal{P}_2}(C, \mathbf{w}, \pi_1)$ checks if π_1 was generated by $O_{\mathcal{P}_1}(\cdot)$ together with $\mathtt{st}, \mathbf{w}_\mathtt{r}$ and if $C(\mathbf{w}_\mathtt{r}, \mathbf{w}) = 0$. If so then output $\pi_2 \leftarrow \mathcal{P}_2^{\mathsf{H}}(C, \mathbf{w}, \mathtt{st})$, otherwise output $\perp$.
- $O_{\mathcal{S}_1}(\mathcal{C})$ computes $(\pi_1, \mathtt{st}) \leftarrow \mathcal{S}_1^{\mathsf{H}}(\mathcal{C})$, samples $\mathbf{w}_\mathtt{r}$ uniformly at random and outputs $\pi_1, \mathbf{w}_\mathtt{r}$.
- $O_{\mathcal{S}_2}(C, \mathbf{w}, \pi_1)$ checks if π_1 was generated by $O_{\mathcal{S}_1}(\cdot)$ together with $\mathtt{st}$, if $\mathbf{w}_\mathtt{r}$ was generated for π_1 by $O_{\mathcal{S}_1}$ and if $C(\mathbf{w}_\mathtt{r}, \mathbf{w}) = 0$. If so, then output $\pi_2 \leftarrow \mathcal{S}_2^{\mathsf{H}}(C, \mathtt{st})$, otherwise output $\perp$.

The simulator $\mathcal{S}$ is allowed to program the random oracle H on any fresh input to H, i.e. if $\mathsf{H}(m)$ has not been queried by any party before, then $\mathcal{A}$ resp. $\mathcal{S}$ is free to choose $\mathsf{H}(m)$, otherwise, programming fails.

We define the advantage of $\mathcal{A}$ by $|\mathsf{Real}_{\mathcal{A}}(\lambda) - \mathsf{Ideal}_{\mathcal{A}}(\lambda)|$. We call $\mathcal{S}$ a Non-interactive Zero-Knowledge Black-box simulator for Π if for any PPT $\mathcal{A}$ the advantage is negligible.

The knowledge soundness for black-box extraction in the RO model for NIZKs is then defined as follows:

Definition 4.3. Let Π be a non-interactive proof system and $\mathsf{E}_1, \mathsf{E}_2$ be a pair of PPT algorithms. Define the knowledge soundness experiment as follows:

1. The experiment obtains $\mathcal{C}$ as input.
2. Run $\mathcal{A}^{\mathsf{H}}(1^\lambda, \mathcal{C})$ until it outputs $\pi_1, \mathtt{st}$. Let Q_1 be all RO queries made by $\mathcal{A}$.
3. Run $\mathcal{A}^{\mathsf{H}}(\pi_1, \mathtt{st})$ until it outputs C, π_2. Let Q_2 be all RO queries made by $\mathcal{A}$.
4. Return 1 if $C \in \mathcal{C}$, $\mathsf{Ver}(C, \pi_1, \pi_2) = 1$ and for the additional constraint $C(\mathsf{E}_1(\mathcal{C}, \pi_1, Q_1), \mathsf{E}_2(C, \pi_1, \pi_2, Q_2)) \neq 0$, else return 0.

The advantage of $\mathcal{A}$ is defined as the probability that the experiment returns 1. If there exist $\mathsf{E}_1, \mathsf{E}_2$ such that for every PPT $\mathcal{A}$ the advantage is negligible, then Π is knowledge sound.

Remark. Our definition also supports the standard notion of NIZKs, which considers regular circuits C which do not obtain a random witness part $\mathbf{w}_\mathbf{r}$ as input and which only consists of a single prover algorithm $\mathcal{P}$. The standard notion is supported by our definitions by extending C into a circuit $\widehat{C}$ that ignores $\mathbf{w}_\mathbf{r}$ and only operates on $\mathbf{w}$, and then setting $\mathcal{C} = \{\widehat{C}\}$. In that case, one obtains $\mathcal{P}$ by running $\mathcal{P}_1, \mathcal{P}_2$ sequentially and letting $\pi = (\pi_1, \pi_2)$. Our notions of correctness, knowledge-soundness and zero-knowledge collapse to the standard notions of a NIZK.

5 Blind Signatures from Plain MAYO

In this section we construct a Blind signature scheme following Fischlin [24]. Towards this, we combine the MAYO signature scheme $\mathsf{MAYO} = (\mathsf{KG}, \mathsf{Sig}, \mathsf{Ver})$ with the commitment scheme $\mathsf{COM} = (\mathsf{Setup}, \mathsf{Com})$ and the single prover-algorithm NIZK $\mathsf{NIZK} = (\mathcal{P}, \mathcal{V})$ introduced in Section 4.

Towards constructing the 5 algorithms of the Blind signature scheme, note that the Blind signature is a NIZK proof of knowledge of a commitment com, a valid commitment opening μ, r of com as well as a valid MAYO signature sig on com. For the MAYO public key pk, commitment public key cpk, randomness $r \in \{0,1\}^\lambda$ and $\mu \in \{0,1\}^*$, we let $C_{\mathtt{pk},\mathtt{cpk},\mu}(r, \mathsf{sig})$ be a circuit that outputs 0 iff

$$\mathsf{MAYO.Ver}(\mathtt{pk}, \mathsf{Com}(\mathtt{cpk}, \mu, \mathbf{r}), \mathsf{sig}) = 1.$$

We fully describe the scheme in Algorithm 1.

Algorithm 1 Conservative BS

We assume that $\mathtt{cpk} \leftarrow \mathsf{COM.Setup}(1^\lambda)$ is run in advance. cpk is implicit input to all other algorithms.

$\mathsf{KG}(1^\lambda)$:
  Output $(\mathtt{sk}, \mathtt{pk}) \leftarrow \mathsf{MAYO.KG}(1^\lambda)$

$\mathsf{Sig}_1(\mathtt{pk}, \mu)$:
  $\mathbf{r} \leftarrow \{0,1\}^\lambda$
  $\mathsf{com} \leftarrow \mathsf{COM}(\mathtt{cpk}, \mu, \mathbf{r})$ ▷ Committing to message.
  Output $\rho_1 = (\mathsf{com})$ and $S_{\mathcal{U}} = (\mathtt{pk}, \mu, \mathbf{r})$

$\mathsf{Sig}_2(\mathtt{sk}, \rho_1 = (\mathsf{com}))$:
  $\sigma \leftarrow \mathsf{MAYO.Sig}(\mathtt{sk}, \mathsf{com})$ ▷ Sign com.
  Output $\rho_2 = (\sigma)$

$\mathsf{Sig}_3(\rho_2 = (\sigma), S_{\mathcal{U}} = (\mathtt{pk}, \mu, \mathbf{r}))$:
  If $\mathsf{MAYO.Ver}(\mathtt{pk}, \mathsf{com}, \sigma) \neq 1$ then output $\perp$.
  Compute $\pi \leftarrow \mathcal{P}(C_{\mathtt{pk},\mathtt{cpk},\mu}, (\mathbf{r}, \sigma))$
  Output $\mathsf{sig} = (\pi)$.

$\mathsf{Ver}(\mathtt{pk}, \mu, \mathsf{sig} = (\pi))$:
  Output 1 iff $\mathcal{V}(C_{\mathtt{pk},\mathtt{cpk},\mu}, \pi) = 1$, otherwise output 0.

The security proof of our algorithms follows directly from previous works, but we include a sketch for completeness.

Theorem 5.1. *Let* $\mathsf{KG}, (\mathsf{Sig}_i)_{i\in[3]}, \mathsf{Ver}$ *be as defined in Algorithm 1. Then they are a Blind signature scheme that has correctness, blindness and one-more unforgeability according to Definitions 2.3 to 2.5 respectively.*

Proof. Correctness follows directly from the definitions. The commitment scheme is perfectly correct, and both the MAYO signatures and the NIZK are correct except with negligible probability. Therefore, Algorithm 1 is also correct.

Blindness. Consider the standard notion of blindness as defined in Definition 2.4. As a first step, we adapt the security game in such a way, that the challenger uses a simulator for the NIZK when running Sig_3. The difference between the standard game and the modified version is negligible as the simulator is a black-box simulator for the NIZK. Now that the computation of Sig_3 uses a simulator, no knowledge of the actual witness is required. We remove the access to the Sig_3 oracle from the adversary. The oracle only uses public information, as we have utilized the NIZK previously. Hence, the removed signatures sig^0 and sig^1 can just be simulated by the adversary. The adversary only sees commitments for two messages now and being able to distinguish which commitment belongs to which message breaks the hiding of the commitment scheme. As the commitment scheme is hiding, $\mathcal{B}$ only

has a negligble advantage and Algorithm 1 has blindness.

One-more unforgeability. Given any adversary $\mathcal{A}$ against the one-more unforgeability, we construct an adversary $\mathcal{B}$ against the EUF-CMA security of MAYO. $\mathcal{B}$ generates a commitment public key $\mathtt{cpk}$ and sends it to $\mathcal{A}$ together with the MAYO public key. When $\mathcal{A}$ makes a signing query for a c_i, $\mathcal{B}$ asks for a signature on it from its challenger and returns the result to $\mathcal{A}$. It can make at most Q queries. Eventually $\mathcal{A}$ outputs a forgery list of forgeries $(\mu_i^*, \mathsf{sig}_i^*)_{i\in[Q+1]}$. By definition of the soundness property of the NIZK, there are efficient extractors E_1 and E_2 such that they can extract r_i and s_i that are accepted by the circuit captured by the NIZK. Each extractor can fail with a ε probability incurring an additional loss of $(Q+1)\cdot\varepsilon$. This means that s_i is a valid signature for the commitment $\mathsf{COM}(\mathtt{cpk}, \mu_i^*, \mathbf{r}_i) = c_i$. Collisions between the commitments of c_i break the binding property, so each c_i is associated with a distinct μ_i. Therefore, there exists a c_i for $i \in [Q+1]$ that was not queried as input to $\mathsf{Sig}_2(\mathtt{sk}, \cdot)$, and thus, (c_i, s_i) is a valid forgery for the MAYO signature scheme breaking the EUF-CMA security. If $\mathcal{A}$ succeeds in the one-more unforgeability game, then all signatures are valid and assuming that the extractions succeeds, this produces a valid signature for MAYO breaking EUF-CMA. □

6 Blind signatures from One-More-MAYO

For $\mathbf{h} \in \mathbb{F}_q^m$ and MAYO public key $\mathtt{pk}$ let $C_{\mathbf{h},\mathtt{pk}}$ be a circuit that on input $(\mathbf{r}, \mathbf{s})$ outputs 0 iff $\mathcal{T}^*(\mathbf{s}) = \mathbf{h} + \mathbf{r}$. We define $\mathcal{C} = \{C_{\mathbf{h},\mathtt{pk}}\}$ to be the set of all possible such circuits for a fixed $\mathtt{pk}$ but free choice of $\mathbf{h} \in \mathbb{F}_q^m$. Let $\mathsf{H} : \{0,1\}^* \to \mathbb{F}_q^m$ be a hash function modeled as a random oracle.

Theorem 6.1. *Let $\mathsf{KG}, (\mathsf{Sig}_i)_{i\in[3]}, \mathsf{Ver}$ be as defined in Algorithm 2. Then they are a Blind signature scheme that has correctness, blindness and one-more unforgeability according to Definitions 2.3 to 2.5 assuming the hardness of Definition 3.1.*

Proof. The correctness follows directly from the definitions. Assuming the correctness of the NIZK except with negligible probability and assuming that the preimage sampling algorithm works except with negligible probability, honestly generated signatures are accepted except with negligible probability given the definition of the circuit $C_{\mathbf{h},\mathtt{pk}}$.

Blindness. Consider the notion of blindness as defined in Definition 2.4. The procedure for proving the blindness is similar to the proof that we do in Theorem 5.1. First, replace and then remove the Sig_3 algorithm using the NIZK simulator. By definition of the NIZK and its simulator, the partial witness w_{rand}, here r, must be uniform and not recoverable from π_1. By this, both outputs of Sig_1 look uniform, and an adversary can't have any advantage in the blindness game.

Algorithm 2 Optimized BS

$\mathsf{KG}(1^\lambda)$:
  Output $(\mathtt{sk}, \mathtt{pk}) \leftarrow \mathsf{MAYO.KG}(1^\lambda)$

$\mathsf{Sig}_1(\mathtt{pk}, \mu)$:
  $(\pi_1, \mathbf{r}, \mathtt{st}) \leftarrow \mathcal{P}_1(\mathcal{C})$ ▷ $\mathbf{r} \in \mathbb{F}_q^m$
  $\mathbf{h} \leftarrow \mathsf{H}(\mu, \pi_1)$ ▷ $\mathbf{h} \in \mathbb{F}_q^m$
  $\mathbf{t} \leftarrow \mathbf{h} + \mathbf{r}$
  Output $\rho_1 = (\mathbf{t})$ and $S_{\mathcal{U}} = (\mathtt{pk}, \pi_1, \mu, \mathbf{r}, \mathbf{t}, \mathtt{st})$

$\mathsf{Sig}_2(\mathtt{sk}, \rho_1 = (\mathbf{t}))$:
  $\mathbf{s} \leftarrow \mathsf{MAYO.SPre}(\mathtt{sk}, \mathtt{pk}, \mathbf{t})$ ▷ $\mathbf{s} \in \mathbb{F}_q^{kn} : \mathcal{T}^*(\mathbf{s}) = \mathbf{t}$
  Output $\rho_2 = (\mathbf{s})$.

$\mathsf{Sig}_3(\rho_2 = (\mathbf{s}), S_{\mathcal{U}} = (\mathtt{pk}, \pi_1, \mu, \mathbf{r}, \mathbf{t}, \mathtt{st}))$:
  If $\mathcal{T}^*(\mathbf{s}) \neq \mathbf{t}$ output $\perp$
  Compute $C_{\mathbf{h},\mathtt{pk}}$
  $\pi_2 \leftarrow \mathcal{P}_2(C_{\mathbf{h},\mathtt{pk}}, \mathbf{s}, \mathtt{st})$
  Output $\mathsf{sig} = (\pi_1, \pi_2)$

$\mathsf{Ver}(\mathtt{pk}, \mu, \mathsf{sig} = (\pi_1, \pi_2))$:
  $\mathbf{h} \leftarrow \mathsf{H}(\mu, \pi_1)$
  Compute $C_{\mathbf{h},\mathtt{pk}}$
  Output 1 iff $\mathcal{V}(C_{\mathbf{h},\mathtt{pk}}, \pi_1, \pi_2) = 1$, otherwise output 0.

One-more unforgeability. Let $\mathcal{A}$ be an adversary against the one-more unforgeability of Algorithm 2. Now, we construct from this an adversary $\mathcal{B}$ against the one-more unforgeability of MAYO defined in Definition 3.1. $\mathcal{A}$ makes at most Q_{H} hash queries to H, which will be the number of targets Definition 3.1 will be instantiated with. $\mathcal{B}$ receives these initial targets in the beginning. When $\mathcal{A}$ queries H for query $i \in [Q_{\mathsf{H}}]$ with (μ, π_i), $\mathcal{B}$ returns $\mathbf{h}_i \leftarrow \mathbf{t}_i - \mathbf{r}_i$ where it extracts $\mathbf{r}_i$ from π_i and the list of random oracle queries using the extractor E_1 defined by the NIZK. As $\mathbf{t}_i$ is uniform, $\mathbf{h}_i$ is also uniform. When $\mathcal{A}$ queries the Sig_2 oracle for $\mathbf{t}_i$ where $i \in [Q]$, $\mathcal{B}$ asks for a preimage of $\mathbf{s}_i$ from its challenger. It asks for at most Q preimages for the preimage sampling function from MAYO. Eventually $\mathcal{A}$ outputs $Q+1$ forgeries $(\mu_i, (\pi_{i,1}, \pi_{i,2}))$. All μ_i are distinct, which means that all $\mathsf{H}(\mu_i, \pi_{i,1})$ are distinct except with negligible probability. If $\mathcal{A}$ never queried for $(\mu_i, \pi_{i,1})$, the probability of the signature being correct is negligible, as the hash determines the target uniquely, and a proof can only be valid for exactly one target. The hash function follows the uniform distribution over the target space, which is superpolynomial in size. Now that all $\mathsf{H}(\mu_i, \pi_{i,1}) = \mathbf{h}_i$ were queried and are all distinct, we can use the NIZK extractor and extract $\mathbf{s}_i$ and $\mathbf{r}_i$ such that $\mathcal{T}^*(\mathbf{s}_i) = \mathbf{h}_i + \mathbf{r}_i = \mathbf{t}_i$. By definition, these $\mathbf{t}_i$ are all distinct and given by the challenger, and $\mathcal{B}$ can simply return $(\mathbf{s}_i, I_i)_{i\in[Q+1]}$ where I_i identify the correct $\mathbf{t}_i$ to which $\mathbf{s}_i$ is the preimage. □

7 Rain Hash Function

This section describes a zk-friendly, in particular MPCitH and VOLEitH-friendly hash construction, RainHash. It is a larger variant of the Rain permutation proposed by Dobraunig et al. [21] for constructing OWFs used in MPCitH- and VOLEitH-based digital signatures [6, 21] and adjusting the number of internal rounds to accommodate for the different security expectations. Our most conservative Blind signature proves the correct execution of SHAKE256 permutation inside the VOLEitH zk-proof. We show that a Blind signature when instantiated using VOLEitH-friendly RainHash enjoys a much smaller signature size and faster runtime than SHAKE256 due to RainHash's design choice with a low number of non-linear operations over a minimal state size required in the particular use-case with 128-bit of security level. In the past, such special hash designs have been explored for different use-cases like hardware-friendly [20], software-friendly [4], zkSNARKs-friendly [27, 28], among other use-cases [25, 26]. As MPCitH and VOLEitH applications mature, like being used to construct post-quantum signatures, general ZK-proof systems, and Blind signatures (with this work), there is more motivation to explore VOLEitH-friendly hash designs providing smaller communication and faster circuit prove/verify runtime.

The Rain Permutation. The Rain permutation was inspired by the AES S-Box and Nyberg's S-Box [40]. This gives evidence for cryptanalytic resistance of the Rain permutation given the large body of work investigating AES, while increasing the S-Box size and using a more unstructured affine mapping, a random matrix multiplication $M \in \mathbb{F}_2^{n \times n}$ for improved diffusion compared to more structured designs. In MPCitH- and VOLEitH-based signatures, the linear layer does not have a high impact of the performance and the non-linear layers, the S-Box over $\mathbb{F}_{2^n}$, enjoys efficient inverse checks as shown in Helium [32] or FAEST [8].

The Rain permutation (without a secret key) $f_k(x) : \mathbb{F}_2^\lambda \to \mathbb{F}_2^\lambda$ 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

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

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 and $\mathbf{M}_i \in \mathbb{F}_2^{\lambda \times \lambda}$ an invertible matrix.

RainHash Construction. RainHash uses the same round function as the Rain. We refer to Fig. 1 for the actual construction. It performs 7 rounds of the Rain permutation without any special treatment of the last round.When constructing OWFs with Rain, the last linear layer was omitted, as an attacker could simply revert the output of the matrix multiplication with its inverse. In the RainHash construction, however, keeping it improves the diffusion as the output is not trivially revertible as the input message is XOR-ed to the state after the last linear layer. As in Rain, matrices and round constants are public. RainHash uses the same ideas as other short-input hashes [34] and can be traced back to Davies–Meyer [39] replacing the keyed with a "fixed key" permutation.

Security Discussions For the original goal of a OWF, the authors of Rain concluded that the number of permutation rounds could conservatively be chosen to be 4, with 3 rounds as a more aggressive setting leading to better implementation characteristics and less security margin. In the last years, several attack strategies [36, 44] were proposed, violating security expectations for the OWF use-case up to two rounds.

For the choice of a conservative number of rounds for RainHash we start by re-using the existing security analysis from [21] which states that all known attack vectors are expected to stop working after 3 rounds, for all block sizes. Their setting is however quite restricted for an attacker. We assume that due to our distinct requirements (collision and preimage resistance rather than mere one-wayness), attackers can employ so called "inside-out" approaches that could in the limit double the number of rounds that can be attacked. Classic examples of this are rebound attacks on AES-like hash functions [35]. The original analysis includes various algebraic and statistical approaches which are also very relevant for collision attacks. Hence we deem a choice of 7 rounds (doubling the number of rounds for the OWF property which includes a security margin) to be a conservative choice. Future cryptanalysis may suggest a smaller number of rounds to be sufficient, but in the context of this work this demonstrates the advantages of using a custom hash function for VOLEitH.

RainHash Instances. As an example, we propose the instance RainHash512 with a state size of 512 bits for L1 security. Generating larger instances for higher security levels is straight-forward. With the implementation we provide the RainHash round constants and round invertible matrices, and a *sage* script to generate the parameters.

MPC and ZK-Friendliness RainHash, due to its low number of rounds and smaller state size than many other hash functions, enjoys a very competitively small non-linear complexity in terms bytes communicated as part of the proof. One hash function call has only 448 bytes of non-linear communication complexity, compared to un-optimized evaluation of SHAKE256 with 4800 bytes and 800 bytes for our VOLEitH-optimized SHAKE256 instance. Refer to Table 2 for proof size comparison between RainHash and SHAKE256.

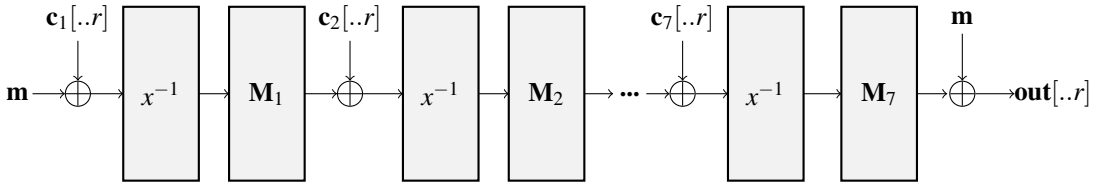

Figure 1: RainHash construction with with 7 rounds of Rain permutation. $\mathbf{m}$ is the input to the hash functions. $\mathbf{c}_i$ are the round constants of the respective i^{th} round XOR-ed with the i^{th} round state. $\mathbf{M}_i$ denotes the multiplication with an unstructured invertible matrix over $\mathbb{F}_2$ in the i-th round. **out** is the output of RainHash after XOR-ing with the input message $\mathbf{m}$.

Scheme	Proof Size (KB)
RainHash512$_{s/f}$	7.6/10.5
SHAKE256-deg2$_{s/f}$	55.5/80.1
SHAKE256-deg16$_{s/f}$	14.9/21.1

Table 2: VOLEitH proof size comparison between RainHash and SHAKE256.

8 Implementation

Blind Signature. Our implementations use the NIST PQ DSA Round 2 submissions of MAYO and FAEST. The MAYO codebase provides functionalities for preimage sampling and only minor modifications were done in the C codebase. We used the C++ FAEST framework, which implements a VOLEitH NIZK proof system over $\mathbb{F}_{2^8}$, to implement two VOLEitH circuit descriptions, namely for SHAKE256 and the MAYO verification. All field operations, wherever possible, were implemented in AVX. From the implementations in C and C++ we compile shared libraries and with the Rust crate bindgen[6] we generate foreign function interfaces making it possible to use the implementations in Rust. The functions in the actual Blind signature algorithm, namely, Sig_1, Sig_2, Sig_3 and Ver are implemented in Rust and make calls to the underlying implementations. This mixture of C++ & Rust is due to our use of a legacy codebase. In the interest of memory-safety and to simplify the codebase, a Rust-based re-implementation of VOLEitH would be interesting future work. SHAKE256 is implemented in two VOLEitH circuits, deg-2 and deg-16, providing a trade-off between signature size and prover runtime.

RainHash. We also provide the RainHash implementation as a circuit for VOLEitH proofs as defined in Section 4. The RainHash plain implementation is AVX optimized. Due to the large state size of $\mathbb{F}_{2^{512}}$, all multiplications are done with Karatsuba's algorithm and for reduction we use Itoh-Tusuji algorithm. Depending on the use case, we provide two variants of computing the multiplication and reduction over $\mathbb{F}_{2^{512}}$ bits representation. The standard multiplication calls `clmul` Karatsuba over 512 bits directly and reduces with irreducible polynomial over $x^{512}+x^8+x^5+x^2+1$. The second variant is crafted for the VOLEitH circuit proof/verify use-case since the FAEST verifier performs operations over $\mathbb{F}_{2^\lambda}$ to achieve security levels $\lambda = 128, 192, 256$ respectively and since the circuit for FAEST must be defined over a subfield of $\mathbb{F}_{2^\lambda}$, checking the RainHash inverse S-Box of 512-bit-size required some optimization instead of naively increasing the verifier bit level to 512-bits (which would make computation significantly more expensive and the proof substantially larger). Namely, we view the 512-bits input/output state of the S-Box as $\mathbb{F}^4_{2^{128}}$ state represented as a degree-3 polynomial with coefficients from $\mathbb{F}_{2^{128}}$. Any inverse check multiplies the input polynomial $f_3X^3+\cdots+f_0$ and output polynomial $g_3X^3+\cdots+g_0$ representing the input and output of the S-Box, yielding a degree-6 polynomial $h_6X^6+\cdots+h_0$. which is then reduced to the degree-3 polynomial $h'_3X^3+\cdots+h'_0$ using an irreducible polynomial in $\mathbb{F}_{2^{128}}[X]$ of degree-4 which defines a field extension $\mathbb{F}_{2^{128}} \subset \mathbb{F}_{2^{512}}$. All these operations can be efficiently done in the ZK proof system. The final check in the ZK proof system is to test if $h_3, h_2, h_1 = 0$ and $h_0 = 1$, which tests that the output of the S-Box is indeed the inverse of the input.

This leads to the following design of a circuit that evaluates RainHash: first, the prover commits to the input as well as the output of x^{-1} in each round. Here, each element is committed over $\mathbb{F}_2$. Next, the prover applies the remaining linear functions for each round i forward until it reaches a bit-wise encoding of the input of the S-Box. Note that using the Lift gates, we can lift all inputs to elements in $\mathbb{F}_{2^\lambda}$. Next, in each round i the $\mathbb{F}_2$-committed state is propagated backwards through the linear layer until it reaches the output of the S-Box. Similarly, the outputs are now lifted to elements in $\mathbb{F}_{2^\lambda}$. Finally, all S-Boxes are checked as mentioned above and the output of RainHash is revealed to the verifier.

Evaluation of MAYO. The components $p_1(\mathbf{s}), \ldots, p_m(\mathbf{s})$ of the MAYO trapdoor $\mathcal{T}$ are quadratic polynomials over $\mathbb{F}_{16}$, it is straightforward to add these quadratic constraints to the VOLEitH proof to prove $\mathcal{T}(\mathbf{s}) = \mathbf{t}$. However, to reduce the running time of the prover and verifier, we make use of the optimization described in [6], where given a $\mathbb{F}_{16}$-basis $B = \{b_1, \ldots, b_{\lambda/4}\}$ for $\mathbb{F}_{2^\lambda}$, the idea is to prove the linear com-

[6] https://github.com/rust-lang/rust-bindgen

Scheme	Runtime in ms				Size in KB	
	Sig_1	Sig_2	Sig_3	Ver	Communication	Blind Signature
SHAKE256+MAYO-128_s	0.021	5.651	292.555	296.672	0.47	92.66
SHAKE256+MAYO-128_f	0.022	5.928	52.760	50.466	0.47	163.15
SHAKE256+MAYO-192_s	0.023	15.428	844.999	827.654	0.71	149.06
SHAKE256+MAYO-192_f	0.024	15.488	580.431	551.821	0.71	252.29
SHAKE256+MAYO-256_s	0.026	37.914	1706.023	1699.451	1.00	220.66
SHAKE256+MAYO-256_f	0.025	37.116	1411.481	1402.816	1.00	348.23
SHAKE256-deg16+MAYO-128_s	0.023	6.119	178.067	143.088	0.47	24.32
SHAKE256-deg16+MAYO-128_f	0.019	5.314	75.591	41.794	0.47	41.65
SHAKE256-deg16+MAYO-192_s	0.023	16.159	761.779	689.241	0.71	44.28
SHAKE256-deg16+MAYO-192_f	0.023	16.082	630.773	555.048	0.71	72.67
SHAKE256-deg16+MAYO-256_s	0.025	39.541	1686.270	1546.398	1.00	73.16
SHAKE256-deg16+MAYO-256_f	0.024	39.308	1477.271	1385.398	1.00	112.23
RainHash+MAYO-128_s	0.305	6.174	114.399	125.268	0.47	16.51
RainHash+MAYO-128_f	0.265	5.782	49.783	44.862	0.47	27.78
One-More-MAYO-128_s	47.191	6.053	42.004	91.305	0.45	6.73
One-More-MAYO-128_f	1.004	5.900	39.972	40.145	0.45	10.39
One-More-MAYO-192_s	73.454	16.109	541.852	592.591	0.68	15.49
One-More-MAYO-192_f	2.442	15.872	561.136	545.397	0.68	23.31
One-More-MAYO-256_s	11.318	34.313	1434.543	1369.497	0.97	41.00
One-More-MAYO-256_f	8.614	39.354	1414.202	1467.207	0.97	43.20

Table 3: Blind signature signing times (ms), verification times (ms), and signature sizes (KB) of plain MAYO and One-More-MAYO based schemes. Slow and fast versions of the VOLEitH proof are denoted with s and f respectively. The verification keys consist of a commitment public key with 16/24/32B, the MAYO public key with 1.420/2.986/5.554KB, and the secret key is MAYO's secret key of size 24/32/40B for NIST level 1/3/5.

bination $\sum_{i=1}^{\lambda/4} b_i p_i(\mathbf{s})$ instead of proving the p_i individually. This allows to combine $\lambda/4$ equations over $\mathbb{F}_{16}$ into one equation over $\mathbb{F}_{2^\lambda}$, which is more efficient. Since B is a basis, the linear combination is satisfied if and only if all p_i are satisfied. However, this still leaves $\lceil 4m/\lambda \rceil$ equations to be proven. We take the idea one step further and combine all the equations over $\mathbb{F}_{16}$ into a single equation over $\mathbb{F}_{2^\lambda}$. Instead of using a basis B (of length $\lambda/4$), we sample a list L of length m of uniformly random elements of $\mathbb{F}_{2^\lambda}$ and consider a single linear combination $\sum_{i=1}^{m} \ell_i p_i(\mathbf{s})$. It is crucial that the list be chosen only after the prover has committed to the witness. It is easy to prove that if at least one of the constraints is not satisfied, then the linear combination will not be satisfied with probability $2^{-\lambda}$, which means that the proof system remains sound with this optimization. In practice, this optimization reduces the proving and verification time of the MAYO trapdoor by roughly a factor of 2 for our chosen parameter sets.

Results. We evaluated our Blind signature implementations and report on the runtime and the associated communication and signature sizes in Table 3. On 128 bit security level, each of our constructions outperforms current lattice-based proposals (see Table 1) with competitive or improved size.

We report the performance of the Blind signature with and without the optimized degree-16 Keccak circuit technique, marked SHAKE256-deg16+MAYO and SHAKE256+MAYO respectively, as well as the optimized construction that avoids proving a statement about symmetric primitives One-More-MAYO. We don't expect the SHAKE256+MAYO implementation to be useful, as it is larger and less efficient and otherwise has no advantages over SHAKE256-deg16+MAYO. It is included in Table 3 purely to showcase the effect of the degree-16 optimization. As a contribution of independent interest, we also present a new VOLEitH-friendly hash function, RainHash, complimenting our conservative Blind signature construction, RainHash+MAYO, with even smaller signature size and faster runtime than when using Keccak.

9 Acknowledgments

We would like to thank the anonymous reviewers and the shepherd for their valuable comments and suggestions. Carsten Baum was supported by research grant VIL53029 from VILLUM FONDEN. Marvin Beckmann acknowledges support from the Danish Ministry of Defense Acquisition and Logistics Organization (FMI).

Ethical Considerations

In this work, we have demonstrated that certain cryptographic primitives are viable with security against quantum computers. Our work primarily benefits the privacy of users that employ digital signature schemes in use-cases such as e-cash or as a building block to construct anonymous credential schemes. While fully anonymized payments have known societal drawbacks (money laundering, avoidance of sanctions, financing of terrorism), one can add traceability mechanisms to them to counter these. Such traceability mechanisms are particularly viable using ZK-based approaches such as ours. Our work does not impact any currently running system, service or users in any - particularly not harmful - way. To the best of our understanding, no harm is done due to our work that upgrades a popular cryptographic construction with stronger security guarantees.

Open Science

The artifacts of this work are available here: https://github.com/shibammukherjee/pq_blind_signatures, permanent Zenodo DOI: https://doi.org/10.5281/zenodo.17979641. The artifacts contain a README, codebase, benchmarks and a test and benchmark script `bench.sh`.

Our testing platform. Benchmarks are performed on an intel Ultra 9 Processor 185H, running Ubuntu 24.04. The codebase is compiled with gcc version 13.3.0 with $-$`O2` and $-$`fomit` $-$ `frame` $-$ `pointer` flags.

How to replicate experiments. We provide a README alongside our artifacts. The README gives a project overview explaining the projects structure, dependencies and instructions explaining how to replicate the benchmarks. Once all required dependencies are installed it is sufficient to just run `bench.sh` script, automatically compiling and running the tests and benches.

References

[1] Amit Agarwal, Carsten Baum, Lennart Braun, and Peter Scholl. Low-bandwidth mixed arithmetic in VOLE-based ZK from low-degree PRGs. pages 396–426, 2025.

[2] Shweta Agrawal, Elena Kirshanova, Damien Stehlé, and Anshu Yadav. Practical, round-optimal lattice-based blind signatures. pages 39–53, 2022.

[3] Sven Argo, Tim Güneysu, Corentin Jeudy, Georg Land, Adeline Roux-Langlois, and Olivier Sanders. Practical post-quantum signatures for privacy. pages 1523–1537, 2024.

[4] Jean-Philippe Aumasson, Samuel Neves, Zooko Wilcox-O'Hearn, and Christian Winnerlein. BLAKE2: simpler, smaller, fast as MD5. In Michael J. Jacobson Jr., Michael E. Locasto, Payman Mohassel, and Reihaneh Safavi-Naini, editors, Applied Cryptography and Network Security - 11th International Conference, ACNS 2013, Banff, AB, Canada, June 25-28, 2013. Proceedings, volume 7954 of Lecture Notes in Computer Science, pages 119–135. Springer, 2013.

[5] Foteini Baldimtsi, Jiaqi Cheng, Rishab Goyal, and Aayush Yadav. Non-interactive blind signatures: Post-quantum and stronger security. pages 70–104, 2024.

[6] Carsten Baum, Ward Beullens, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. One tree to rule them all: Optimizing GGM trees and OWFs for post-quantum signatures. pages 463–493, 2024.

[7] Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Sebastian Ramacher, Christian Rechberger, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Faest: Algorithm specifications (version 1.1), 2025.

[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. pages 581–615, 2023.

[9] Carsten Baum, Samuel Dittmer, Peter Scholl, and Xiao Wang. Sok: vector OLE-based zero-knowledge protocols. 91(11):3527–3561, 2023.

[10] Fabrice Benhamouda, Tancrède Lepoint, Julian Loss, Michele Orrù, and Mariana Raykova. On the (in)security of ROS. pages 33–53, 2021.

[11] Ward Beullens. MAYO: Practical post-quantum signatures from oil-and-vinegar maps. pages 355–376, 2022.

[12] Ward Beullens. Multivariate blind signatures revisited. pages 230–236, 2024.

[13] Ward Beullens, Fabio Campos, Sofía Celi, Basil Hess, and Mattias J. Kannwischer. Mayo 2: Algorithm specifications, 2025.

[14] Ward Beullens, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Gregor Seiler. Lattice-based blind signatures: Short, efficient, and round-optimal. pages 16–29, 2023.

[15] Jonathan Bootle, Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Alessandro Sorniotti. A framework for

practical anonymous credentials from lattices. pages 384–417, 2023.

[16] Charles Bouillaguet, Thibauld Feneuil, Jules Maire, Matthieu Rivain, Julia Sauvage, and Damien Vergnaud. Blinding post-quantum hash-and-sign signatures. Cryptology ePrint Archive, Paper 2025/895, 2025.

[17] Charles Bouillaguet, Thibauld Feneuil, Jules Maire, Matthieu Rivain, Julia Sauvage, and Damien Vergnaud. Multivariate commitments and signatures with efficient protocols. Cryptology ePrint Archive, Paper 2025/2035, 2025.

[18] David Chaum. Blind signatures for untraceable payments. pages 199–203, 1982.

[19] Rafaël del Pino and Shuichi Katsumata. A new framework for more efficient round-optimal lattice-based (partially) blind signature via trapdoor sampling. pages 306–336, 2022.

[20] Christoph Dobraunig, Maria Eichlseder, Florian Mendel, and Martin Schläffer. Ascon v1.2: Lightweight authenticated encryption and hashing. J. Cryptol., 34(3):33, 2021.

[21] Christoph Dobraunig, Daniel Kales, Christian Rechberger, Markus Schofnegger, and Greg Zaverucha. Shorter signatures based on tailor-made minimalist symmetric-key crypto. pages 843–857, 2022.

[22] Ming Duan and Xuejia Lai. Improved zero-sum distinguisher for full round keccak-f permutation. IACR Cryptol. ePrint Arch., page 23, 2011.

[23] Morris J Dworkin et al. Sha-3 standard: Permutation-based hash and extendable-output functions. 2015.

[24] Marc Fischlin. Round-optimal composable blind signatures in the common reference string model. pages 60–77, 2006.

[25] Lorenzo Grassi, Dmitry Khovratovich, Katharina Koschatko, Christian Rechberger, Markus Schofnegger, and Verena Schröppel. Poseidon2b: A binary field version of poseidon2. IACR Cryptol. ePrint Arch., page 1893, 2025.

[26] Lorenzo Grassi, Dmitry Khovratovich, Reinhard Lüftenegger, Christian Rechberger, Markus Schofnegger, and Roman Walch. Monolith: Circuit-friendly hash functions with new nonlinear layers for fast and constant-time implementations. IACR Trans. Symmetric Cryptol., 2024(3):44–83, 2024.

[27] Lorenzo Grassi, Dmitry Khovratovich, Christian Rechberger, Arnab Roy, and Markus Schofnegger. Poseidon: A new hash function for zero-knowledge proof systems. In Michael D. Bailey and Rachel Greenstadt, editors, 30th USENIX Security Symposium, USENIX Security 2021, August 11-13, 2021, pages 519–535. USENIX Association, 2021.

[28] Lorenzo Grassi, Dmitry Khovratovich, and Markus Schofnegger. Poseidon2: A faster version of the poseidon hash function. In Nadia El Mrabet, Luca De Feo, and Sylvain Duquesne, editors, Progress in Cryptology - AFRICACRYPT 2023 - 14th International Conference on Cryptology in Africa, Sousse, Tunisia, July 19-21, 2023, Proceedings, volume 14064 of Lecture Notes in Computer Science, pages 177–203. Springer, 2023.

[29] Eduard Hauck, Eike Kiltz, Julian Loss, and Ngoc Khanh Nguyen. Lattice-based blind signatures, revisited. pages 500–529, 2020.

[30] Corentin Jeudy, Adeline Roux-Langlois, and Olivier Sanders. Lattice signature with efficient protocols, application to anonymous credentials. pages 351–383, 2023.

[31] Corentin Jeudy and Olivier Sanders. Improved lattice blind signatures from recycled entropy. Cryptology ePrint Archive, Report 2024/1289, 2024.

[32] Daniel Kales and Greg Zaverucha. Efficient lifting for shorter zero-knowledge proofs and post-quantum signatures. Cryptology ePrint Archive, Report 2022/588, 2022.

[33] Shuichi Katsumata, Yi-Fu Lai, and Michael Reichle. Breaking parallel ROS: Implication for isogeny and lattice-based blind signatures. pages 319–351, 2024.

[34] Stefan Kölbl, Martin M. Lauridsen, Florian Mendel, and Christian Rechberger. Haraka v2 - efficient short-input hashing for post-quantum applications. IACR Trans. Symmetric Cryptol., 2016(2):1–29, 2016.

[35] Mario Lamberger, Florian Mendel, Christian Rechberger, Vincent Rijmen, and Martin Schläffer. Rebound distinguishers: Results on the full Whirlpool compression function. pages 126–143, 2009.

[36] Fukang Liu, Mohammad Mahzoun, Morten Øygarden, and Willi Meier. Algebraic attacks on RAIN and AIM using equivalent representations. IACR Trans. Symmetric Cryptol., 2023(4):166–186, 2023.

[37] Vadim Lyubashevsky, Ngoc Khanh Nguyen, and Maxime Plançon. Efficient lattice-based blind signatures via gaussian one-time signatures. pages 498–527, 2022.

[38] Vadim Lyubashevsky, Gregor Seiler, and Patrick Steuer. The LaZer library: Lattice-based zero knowledge and succinct proofs for quantum-safe privacy. pages 3125–3137, 2024.

[39] Stephen M Matyas. Generating strong one-way functions with cryptographic algorithm. IBM Technical Disclosure Bulletin, 27:5658–5659, 1985.

[40] Kaisa Nyberg and Lars Ramkilde Knudsen. Provable security against differential cryptanalysis. In Annual international cryptology conference, pages 566–574. Springer, 1992.

[41] Albrecht Petzoldt, Alan Szepieniec, and Mohamed Saied Emam Mohamed. A practical multivariate blind signature scheme. pages 437–454, 2017.

[42] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. 13(3):361–396, June 2000.

[43] Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. pages 2986–3001, 2021.

[44] Kaiyi Zhang, Qingju Wang, Yu Yu, Chun Guo, and Hongrui Cui. Algebraic attacks on round-reduced rain and full AIM-III. pages 285–310, 2023.

A On the One-More-MAYO assumption

The difference between OMM and EUF-CMA security of MAYO is that the OMM adversary can obtain preimages for arbitrarily chosen outputs, instead of preimages for the output of a hash function. This extra freedom does not appear to help an attacker forge extra signatures, but we will briefly discuss two natural (but fruitless) attempts to make use of the preimage oracle. Firstly, since the MAYO secret key is a linear space of vectors that all evaluate to 0 under the public key, one might think that asking for preimages of 0 could reveal a vector in the secret key space, which would be catastrophic. However, preimages for 0 sampled using the secret key are statistically close to uniformly random preimages, and therefore lie in the secret space with extremely small probability, showing that this is not a problem. Secondly, one can use the inversion oracle to obtain slightly larger multicollisions for the MAYO trapdoor by first computing a multicollision of size k, i.e. compute $\mathbf{s}_1,\dots,\mathbf{s}_k$ such that $\mathbf{t} = \mathsf{Eval}(\mathtt{pk},\mathbf{s}_1) = \cdots = \mathsf{Eval}(\mathtt{pk},\mathbf{s}_k)$, and then asking the oracle for a $k+1$-th preimage for $\mathbf{t}$. Concretely, for MAYO_1 it is possible to compute multicollisions of size only up to 11 in polynomial time without using the preimage oracle, so the adversary can obtain multi-collisions of size up to 12 using the oracle. While multi-collisions of size 12 do not seem to threaten security, this separation shows that a new hardness assumption is indeed necessary to make the security proof work. As with any new assumption, cryptanalysis is needed to build confidence. Note that the recent work on the Cryptology ePrint archive by Bouillaguet et al. independently used the same assumption to construct "signatures with efficient protocols" [17].

B VOLE-in-the-head-based NIZKs

We now present a (short) introduction into VOLE-in-the-Head (VOLEitH)-based NIZKs of [8]. We explain their high-level structure and how they evaluate circuits based on commitments. This description of circuit evaluation uses our terminology from Section 4, which is compatible with the optimizations and notation from [1,6]. We also argue that these algorithms are compatible with our NIZK notion presented in Section 4, meaning we do not have to change these algorithms to fit our generalized definitions.

B.1 How VOLEitH-based NIZKs work

Let $g,\lambda \in \mathbb{N}^+$ such that $g|\lambda$ and define $q = 2^g$. A (subfield) 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 $\Delta \in \mathbb{F}_{2^\lambda}$ as well as random vectors $\mathbf{u} \in \mathbb{F}_q^m$, a random VOLE tag $\mathbf{v} \in \mathbb{F}_{2^\lambda}^m$ and VOLE keys $\mathbf{x} \in \mathbb{F}_{2^\lambda}^m$ such that[7] $\mathbf{x} = \mathbf{u}\cdot\Delta - \mathbf{v}$. $\mathcal{P}$ obtains $\mathbf{u},\mathbf{v}$ while $\mathcal{V}$ obtains $\Delta,\mathbf{x}$. The correlation commits $\mathcal{P}$ to the u_i values (i.e. the coefficients in $\mathbf{u}$) as linearly homomorphic commitments, allowing to construct efficient interactive ZK proof systems (see more on this below). One of the main drawbacks of such VOLE-based ZK schemes (see [9] for a recent survey) is that they are inherently designated verifier proof systems since the verifier $\mathcal{V}$ needs to know its part of the VOLE correlation to verify the proof. At the same time, $\Delta,\mathbf{x}$ has to remain secret from the prover for the proof to be sound. These proof systems can be made non-interactive (after setting up the VOLE correlation through interaction) using the Fiat-Shamir transform.

In [8], Baum et al. realized a delayed VOLE functionality that allows the prover to generate vectors $\mathbf{u},\mathbf{v}$ of VOLE correlations independently of $\Delta,\mathbf{x}$. Later on, anyone (including the prover) can then generate $\mathbf{x}$ based on a choice of Δ. This delayed VOLE functionality is sufficient to instantiate previous VOLE-based ZK proof systems if Δ is chosen using the Fiat-Shamir transform, and can be realized from all-but-one vector commitments (VCs) non-interactively. These VCs allow the sender to efficiently reveal all but one of the indices of a vector committed by the prover, where the verifier can choose the unopened index. The value at the unopened index remains hidden.

In Figure 2 we give a quick overview of the proof system steps of [8]. First, the prover generates τ VCs with random values which in turn define τ random, independent VOLE correlations over a small field $\mathbb{F}_{q^k}$ where $gk\tau = \lambda$. Each of

[7] The product holds over $\mathbb{F}_{2^\lambda}$ where $\mathbf{u}$ is identified as the canonical representative of the $\mathbb{F}_q$-element in the extension field $\mathbb{F}_{2^\lambda}$.

these VOLE correlations commits to a different (random) secret $\mathbf{u}^{(j)}$. By letting the first such VOLE instance be fixed, which also fixes its secret $\mathbf{u} = \mathbf{u}^{(0)}$, the prover can compute correction values $\delta^{(j)} = \mathbf{u} - \mathbf{u}^{(j)}$ for all other instances. The prover then sends these τ VCs, together with $\tau - 1$ correction values $\delta^{(1)}, \ldots, \delta^{(\tau-1)}$ to the verifier, which can later apply the correction values to the $\tau - 1$ VOLE instances. This yields τ "corrected" VOLE instances that all share the same secret $\mathbf{u}$. Moreover, the τ independent VOLE instances with the same secret $\mathbf{u}$ can then be considered as a single VOLE instance with secret $\mathbf{u}$ but where the correlation now holds over $\mathbb{F}_{q^{k\tau}} = \mathbb{F}_{2^\lambda}$. After obtaining a random challenge, the prover will use a consistency check to demonstrate that all correction values δ_i are consistent (i.e. they indeed correct to a single secret $\mathbf{u}$). Finally, prover and verifier run a VOLE-based proof system such as QuickSilver [43], finished by a challenge Δ that defines the $\mathbf{x}^{(j)}$ to be revealed to the verifier based on the VCs. To save rounds, the prover commits to its witness as VOLE commitments after the first challenge, i.e. together with the message containing the VOLE consistency proof.

B.2 Computation on commitments in VOLEitH

As described above, VOLE correlations are lists of tuples (u_i, v_i, x_i) such that the VOLE relation $x_i = u_i\Delta - v_i$ 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 x_i) without knowledge of Δ, and can thus be considered as a designated-verifier commitment to u_i. Binding and hiding of the commitment are explained in more detail e.g. in [9].

To simplify the exposition of circuit evaluation on commitments, we define a commitment $[\![a]\!]^{(d,r)}$ to mean that a value $a \in \mathbb{F}_{q^r}, r \geq 1$ held by the prover, together with a correlation defined as follows:

- $\mathcal{P}$ holds coefficients $(a_0, \ldots, a_{d-1}, a) \in \mathbb{F}_{q^{k\tau}}^d \times \mathbb{F}_{q^r}$, representing the polynomial $\rho_a(t) = a_0 + a_1 t + \cdots + a_d t^d$, where a_d equals a lifted into $\mathbb{F}_{q^{k\tau}}$.
- $\mathcal{V}$ holds $\gamma_a = \rho_a(\Delta) \in \mathbb{F}_{2^\lambda}$.

Note that a VOLE correlation u_i, v_i, x_i forms such a commitment $[\![u_i]\!]^{(1,1)}$ where $\rho_{u_i}(t) = u_i \cdot t - v_i$ and $\gamma_{u_i} = x_i = \rho_{u_i}(\Delta)$.

To open the commitment $[\![a]\!]^{(d,r)}$, the prover simply reveals the degree-d polynomial $\rho_a(t)$ to the verifier which checks that $\rho_a(\Delta) = \gamma_a$ and then outputs the leading coefficient of ρ_a. We refer to [9] for more details.

Note that 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. We now describe how linear as well as lift gates work:

- **Add:** $[\![a+b]\!]^{(d,r)} = [\![a]\!]^{(d_1,r)} + [\![b]\!]^{(d_2,r)}$ with $d = \max\{d_1, d_2\}$ and $d_1 \geq d_2$:
  - $\mathcal{P}$ outputs $\rho_{a+b}(t) = \rho_a(t) + t^{d_1-d_2}\rho_b(t)$
  - $\mathcal{V}$ outputs $\gamma_c = \gamma_a + \Delta^{d_1-d_2}\gamma_b$
- **Add constant:** $[\![a+b]\!]^{(d,r)} = [\![a]\!]^{(d,r)} + b$ with $b \in \mathbb{F}_{q^r}$:
  - $\mathcal{P}$ outputs $\rho_{a+b}(t) = \rho_a(t) + t^d b$
  - $\mathcal{V}$ outputs $\gamma_{a+b} = \gamma_a + \Delta^d b$
- **Multiply by constant:** $[\![c \cdot a]\!]^{(d,r)} = c \cdot [\![a]\!]$ with $c \in \mathbb{F}_{q^r}$. Each party locally multiplies their value by c.
- **Lift:** of $[\![a]\!]^{(d,r)}$ to $[\![a]\!]^{(d,s)}$ where $r|s$: $\mathcal{P}$ lifts $a \in \mathbb{F}_{q^r}$ to its representative in $\mathbb{F}_{q^s}$ via the canonical embedding[8].

Before describing multiplications, we introduce how assert gates work. They allow $\mathcal{V}$ to check that a commitment $[\![a]\!]^{(d,r)}$ has $a = 0$ as leading coefficient. While this can be done by simply sending ρ_a to the verifier (who in turn checks that $a = 0$) this simple approach might leak information. We therefore now describe the algorithm Assert($[\![a]\!]^{(d,r)}$):

1. Let $\rho_a(t) = a_0 + \cdots + a_{d-1}t^{d-1}$, for $a_i \in \mathbb{F}_{q^k}$ (recall that the degree-d coefficient should be zero). $\mathcal{V}$ holds $\gamma_a = \rho_a(\Delta)$.
2. Sample $d-1$ additional random VOLEs, to obtain masks $[\![s_0]\!]^{(1,1)}, \ldots, [\![s_{d-2}]\!]^{(1,1)}$, each represented by a polynomial $\rho_{s_i}(t) = s_i t - r_i$, where r_i, s_i are both uniform in $\mathbb{F}_{2^\lambda}$.
3. $\mathcal{P}$ computes the degree-$(d-1)$ mask polynomial $\rho_{s'}(t) = \sum_{i=0}^{d-2} t^i \cdot \rho_{s_i}(t)$
4. $\mathcal{P}$ sends $\rho(t) = \rho_a(t) + \rho_{s'}(t)$ to the verifier
5. $\mathcal{V}$ computes the corresponding MAC $\gamma' = \gamma_a + \sum_{i=0}^{d-2} \Delta^i \cdot \gamma_{s_i}$ and checks that $\rho(\Delta) = \gamma'$ and $deg(\rho) \leq d-1$

This algorithm perfectly randomizes all coefficients except for the leading coefficient of $\rho_a(t)$. By the fundamental theorem of algebra, Assert($[\![a]\!]^{(d,r)}$) fails for non-zero a except with probability $d/2^\lambda$. It requires the use of $d-1$ additional VOLE correlations and that d $\mathbb{F}_{2^\lambda}$-elements are sent from $\mathcal{P}$ to $\mathcal{V}$.

Amortizing Assert gates. In Section 4 we outline that **Assert** gates do not directly compute outputs that can be inputs to other gate types. However, they can make a circuit evaluation result computed inside the NIZK invalid. As such, all **Assert** gates clearly can simultaneously be evaluated and their evaluation be separated from all other gates of a circuit C. Since

[8] Note that since we define the VOLE correlation already with a secret from $\mathbb{F}_{2^\lambda}$, this is only a local operation for the prover

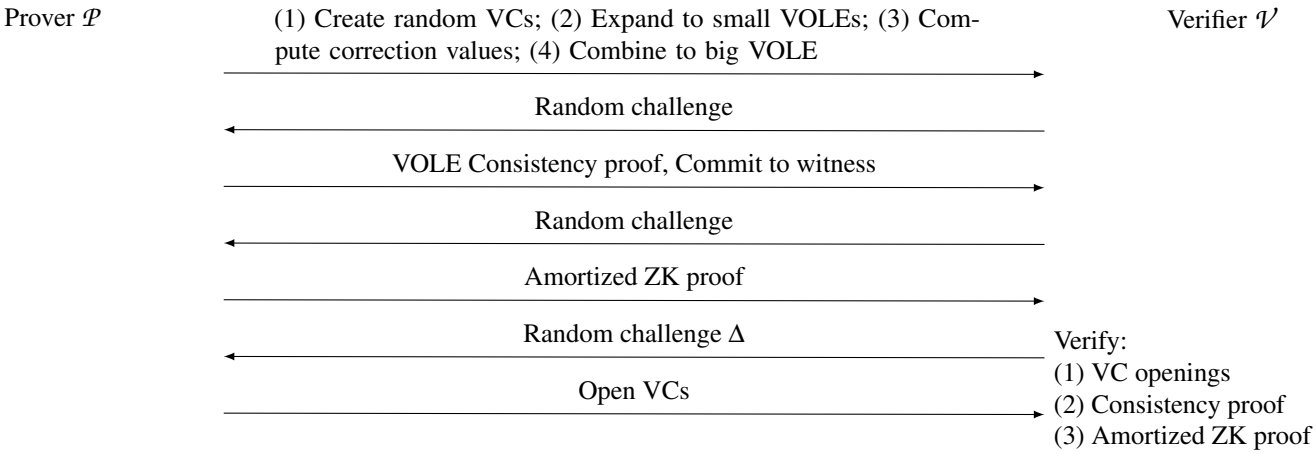

Figure 2: Main steps of the VOLEitH-based ZK proof

Assert creates communication between $\mathcal{P}$ and $\mathcal{V}$, VOLEitH amortizes all evaluations by only checking a random linear combination of all commitments, i.e. merging all **Assert** calls into one. This random linear combination is defined by the second challenge in Figure 2.

To achieve this for n commitments $[\![a_1]\!]^{(d_1,r)},\dots,[\![a_n]\!]^{(d_n,r)}$, $\mathcal{V}$ sends[9] n uniformly random values $\chi_1,\dots,\chi_n \in \mathbb{F}_{2^\lambda}$. $\mathcal{P}$ runs a version of Assert on $\chi_1 a_1 + \dots + \chi_n a_n$ (which must be a 0 for an honest prover). More concretely, the algorithm $\mathsf{Assert}([\![a_1]\!]^{(d_1,r)},\dots,[\![a_n]\!]^{(d_n,r)})$ works as follows:

1. For each $j \in [n]$ let $\rho_j(t)$ be the polynomial $\mathcal{P}$ holds of $[\![a_j]\!]^{(d_j,r)}$. $\mathcal{V}$ holds $\gamma_j = \rho_j(\Delta)$.
2. Set $d = \max_{j\in[n]}\{d_j\}$.
3. Sample $d-1$ additional random VOLEs, to obtain masks $[\![s_0]\!]^{(1,1)},\dots,[\![s_{d-2}]\!]^{(1,1)}$, each represented by a polynomial $\rho_{s_i}(t)$
4. $\mathcal{P}$ computes the degree-$(d-1)$ mask polynomial $\rho_{s'}(t) = \sum_{i=0}^{d-2} t^i \cdot \rho_{s_i}(t)$
5. $\mathcal{V}$ sends the n uniformly random challenges $\chi_1,\dots,\chi_n \in \mathbb{F}_{2^\lambda}$
6. $\mathcal{P}$ sends $\rho(t) = \sum_{j\in[n]} t^{d-d_j}\chi_j\rho_j(t) + \rho_{s'}(t)$ to the verifier
7. $\mathcal{V}$ computes the corresponding MAC $\gamma' = \sum_{j\in[n]} \Delta^{d-d_j}\chi_j\gamma_j + \sum_{i=0}^{d-2}\Delta^i \cdot \gamma_{s_i}$ and checks that $\rho(\Delta) = \gamma'$ and $deg(\rho) \le d-1$

The protocol requires the use of $d-1$ random VOLEs and sends d $\mathbb{F}_{q^k}$-elements to the verifier. The soundness error (i.e. the probability that the check passes if a commitment was not a commitment to 0) is $(d+1)/2^\lambda$. Thus, the efficiency of the extended Assert is dominated by the maximal degree of the input commitments.

Computing and checking Multiplications. While our circuit format in Section 4 only specifies a single type of multiplication gate, we will introduce two different types which we call cmul (for cheap multiplication) and drmul (for degree-reducing multiplication). Both achieve the same outcome (provably creating a commitment to the product of the input commitments) but have differing performance metrics. When implementing circuit evaluation, it is therefore up to the implementer to find the best combination.

Assume that there exist 2 commitments $[\![a]\!]^{(d_1,s)}, [\![b]\!]^{(d_2,s)}$ where the prover wants to compute a commitment $[\![c]\!]$ such that $a \cdot b = c$. The prover holds the polynomials $\rho_a(t), \rho_b(t)$ while the verifier has the evaluations γ_a, γ_b. Since evaluation of a polynomial is a ring homomorphism (and in particular, evaluating $\rho(t)$ in Δ is the evaluation homomorphism) we have that $\mathcal{P}$ can locally compute the degree-d_1+d_2-polynomial $\rho_c(t) = \rho_a(t) \cdot \rho_b(t)$ of which $\mathcal{V}$ implicitly holds the evaluation $\gamma_c = \gamma_a \cdot \gamma_b$. We thus define:

cmul: $[\![c]\!]^{(d,r)} = [\![a]\!]^{(d_1,r)}[\![b]\!]^{(d_2,r)}$, where $d = d_1 + d_2$

- $\mathcal{P}$: Output the coefficients of $\rho_c(t) = \rho_a(t)\rho_b(t)$
- $\mathcal{V}$: Output $\gamma_c = \gamma_a \cdot \gamma_b$

It is straightforward to see that these operations preserve that invariant that the message is stored in the degree-d coefficient and the verifier holds $\gamma_c = \rho_c(\Delta)$. Also note that the degree of the polynomial $\rho(t)$ (at most) doubles during each multiplication. This is important in practice: for each opened commitment of degree d, d coefficients must be sent to the verifier. Moreover, during the evaluation of gates on commitments the prover's computational overhead increases with the degree of polynomials: any subsequent linear or

[9] In the non-interactive proof, these values are generated using the Fiat-Shamir transform as output of a Random Oracle.

multiplication gates must be evaluated on higher-degree polynomials.

To remedy this, when evaluating a circuit, one combines Multiplication with Assert gates to reduce the degree. We call this the drmul gate. To understand how it works, consider a prover who first runs cmul as before. For two commitments $[\![a]\!]^{(d_1,r)}, [\![b]\!]^{(d_2,r)}$, let the prover also commit to the product $c = a \cdot b$ as $[\![c]\!]^{(1,r)}$ such that the commitment only has degree 1, i.e. is a standard VOLE relation. To check consistency between $[\![c]\!]^{(1,r)}$ and $[\![a]\!]^{(d_1,r)} \cdot [\![b]\!]^{(d_2,r)}$, note that the polynomial $\rho_a(t) \cdot \rho_b(t) - t^{d_1+d_2-1}\rho_c(t)$ is zero in its leading coefficient if and only if $c = a \cdot b$. At the same time, $\mathcal{V}$ can compute the evaluation $\gamma_a \cdot \gamma_b - \Delta^{d_1+d_2-1}\gamma_c$ of the polynomial. Both parties can run Assert on the resulting commitment, with the aforementioned communication overhead and soundness loss inherent to Assert.

Circuit evaluation on commitments. We now describe how the input to the proof system 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 VOLEitH [8]. We modify the original approach to suit the circuit evaluation required for our extended definition of a NIZK. This is based on the gate evaluation techniques outlined above.

Let $\mathcal{C}$ denote a set of arithmetic circuits over $\mathbb{F}_q$, containing at most t drmul gates, for which the proof system provides $\mathbf{w}_\mathtt{r} \in \mathbb{F}_q^\ell$ and for which the prover generates an input (i.e. the witness) $\mathbf{w} \in \mathbb{F}_q^\ell$ of length ℓ as well as a circuit $C \in \mathcal{C}$ such that $C(\mathbf{w}_\mathtt{r}, \mathbf{w}) = 0$. To prove its knowledge of the witness, the prover will interact with the verifier to evaluate C. For completeness, let $\overline{d}$ be the largest degree of any $[\![\cdot]\!]$ that is input to any Assert evaluated in the following.

1. The prover requests $2\ell + t + \overline{d} - 1$ VOLE correlations from the VOLE protocol. This provides the prover with $(\rho_{u_i}(t))_{i\in[2\ell+t+\overline{d}-1]}$. The verifier will later compute $(\gamma_{u_i})_{i\in[2\ell+t+\overline{d}-1]}$ and Δ when performing this step as outlined above.
2. For every element in $\mathbf{w}_\mathtt{r}$ we let the first ℓ VOLE correlations with secrets $u_i, i \in [\ell]$ determine the random witness as $\mathbf{w}_\mathtt{r}[i] = u_i$.
3. For every input element w_i of $\mathbf{w}$, for $i \in [\ell]$, the prover computes $d_i := w_i - u_{i+\ell}$ and sends $(d_i)_{i\in[n]}$ to the verifier. This allows the verifier later to update the commitments $[\![u_{i+\ell}]\!]^{(1,1)}$ to $[\![w_i]\!]^{(1,1)}$ locally using the linearity of the commitment scheme described above.
4. 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.
   - cmul gate: the prover and the verifier use the multiplicative property of the commitment scheme to compute their shares of the output commitment locally. Again, this does not require any communication.
   - **Lift** gate: evaluated by the prover locally.
   - i-th drmul gate, for $i \in [t]$: the prover computes $w_\gamma := w_\alpha \cdot w_\beta$ and sends $d_{\ell+i} := w_\gamma - u_{2\ell+i} \in \mathbb{F}_q$ to the verifier. The verifier can later, when holding his shares of the VOLE correlation as well as $d_{\ell+i}$, update the commitment $[\![u_{2\ell+i}]\!]^{(1,1)}$ to $[\![w_\gamma]\!]^{(1,1)}$.
5. The prover runs Assert for all instances of drmul as well as on the commitment $[\![a]\!]^{(d,s)}$ to the output wire of the circuit, thus showing that $a = 0$.

The evaluation of C requires $2\ell + t + \overline{d} - 1$ random VOLE correlations and communicates $\ell + t$ $\mathbb{F}_q$ as well as $\overline{d}$ $\mathbb{F}_{2^\lambda}$-elements from $\mathcal{P}$ to $\mathcal{V}$.

B.3 VOLEitH and our NIZK notion

The protocol outlined above is identical to the VOLEitH protocol of [6–8], except that an additional ℓ VOLE correlations are generated initially and repurposed to become the random witness string $\mathbf{w}_\mathtt{r}$. We will add another very small change to the original protocol. This change is required for the security proof to work, and is as follows: before the first message is sent, the prover will in addition sample a string $\mathsf{rand} \in \{0,1\}^{2\lambda}$ uniformly at random, compute $\mathsf{com}_{\mathsf{rand}} := \mathsf{H}(\mathsf{rand})$ where $\mathsf{com}_{\mathsf{rand}} \in \{0,1\}^{2\lambda}$ as well and add $\mathsf{com}_{\mathsf{rand}}$ to the first message. In the second message sent by the prover, it will include rand as part of the message. During verification, the verifier will check that $\mathsf{com}_{\mathsf{rand}} = \mathsf{H}(\mathsf{rand})$ holds and abort if it does not. This adds 4λ bits of proof size, and one additional hash function call done by both the prover and the verifier.

The first message in Figure 2 (plus $\mathsf{com}_{\mathsf{rand}}$), which is sent by the prover to the verifier, as well as the VOLE Consistency proof from the second message, form π_1. All subsequent messages, namely rand, the ZK proof and the opening of $\mathcal{V}$'s shares of the VOLE correlations, as well as second part of the second message which is the commitment to the witness, become π_2. This is consistent with our definition: the total number of necessary VOLEs to evaluate a circuit $C \in \mathcal{C}$ can be determined in advance, and $\mathcal{P}_1$ will generate as many VOLE correlations necessary for the largest circuit in $\mathcal{C}$.

These VOLE correlations are then, during circuit evaluation, used to commit to the input $\mathbf{w}$ and to evaluate drmul and assert gates as outlined above. The protocol also ensures that the first ℓ VOLE correlations cannot be derandomized by $\mathcal{P}$.

Given that the original proof system is correct, it also follows that our modification fulfills correctness according to Definition 4.1 as can be seen above. The Zero-Knowledge property of Definition 4.2 also reduces to the original Zero-Knowledge property: our protocol is identical to the original protocol, except in two places: we use the additional randomizer and ℓ additional VOLE correlations are sampled to generate and commit to $\mathbf{w}_\mathtt{r}$. One can split the original simulator of the proof of [7, Theorem 2] into two simulators $\mathcal{S}_1, \mathcal{S}_2$, which then fulfill Definition 4.2: $\mathcal{S}_1$ simulates the VOLE correlation generation the VOLE consistency check as well as generation of $\mathtt{com}_{\mathsf{rand}}$, which are the messages contained in π_1. $\mathcal{S}_2$ will simulate rand, the commitment to the witness $\mathbf{d}$, the ZK proof and the opening. The simulator described in [7, Theorem 2] works as follows: it first samples the random challenges output by the Fiat-Shamir transform in the beginning of the simulation and programs the Random Oracle to output these, making sure that the Random Oracle was not queried before by the attacker on the programmed points. It then first rerandomizes all small VOLEs and correction values, then the VOLE Consistency proof, the Amortized ZK proof and finally the commitment to the witness. It appears that the first steps that randomize the Fiat Shamir challenges can directly be emulated in our protocol: $\mathcal{S}_1$ chooses the random challenges and saves them as well as any other choices that it makes as st. Then, $\mathcal{S}_2$ continues the programming. However, this does not work in our setting: for the second challenge, the proof of [7, Theorem 2] relies on the secrecy of the first challenge to the adversary before the whole proof is released to it. We cannot use this, as $\mathcal{S}_1$ outputs the whole first message to the adversary before obtaining $\mathbf{w}$ from it. The second message (chosen by $\mathcal{S}_2$) also includes $\mathbf{d} = \mathbf{w} - \mathbf{u}$ and should therefore be sufficiently unpredictable. However, this cannot be used to program the random oracle to output a random second challenge either: at this point in the proof we cannot argue that $\mathbf{u}$ is uniformly random as this relies on [7, Lemma 5]. However, this Lemma can only be applied once the last challenge in the proof has been randomized (i.e. we finished programming the Random Oracle) as it crucially relies on the challenge being uniformly random. Thus, it appears that we are stuck.

We solve this problem using rand. Note that it is part of st and not revealed by the adversary by $\mathcal{S}_1$ and only later by $\mathcal{S}_2$ once the proof is finished. Hence, using the programmability of H we can choose a uniform rand before reprogramming the second challenge to a uniform value as before. Thereafter, we can apply the same proof technique as in [7] as the adversary cannot predict rand except with a negligibly small probability. Note that having $\mathcal{C}$ instead of a fixed circuit C as input to $\mathcal{S}_1$ does not change the simulation strategy which only relies on the number of required VOLE correlations. Arguing that the generated $\mathbf{w}_\mathtt{r}$ string is uniformly random follows from the proof that $\mathbf{d}$ generated in the VOLEitH proof are uniformly random.

It remains to show that the algorithms fulfill our notion of soundness. Note that Definition 4.3 requires the existence of a pair of extractors $\mathsf{E}_1, \mathsf{E}_2$ which output $\mathbf{w}_\mathtt{r}, \mathbf{w}$ separately. In [7, Theorem 1] the authors show that their proof system is a straight-line extractable knowledge-sound NIZK. This is because the VCs which generate the VOLE-correlation are instantiated in the Random Oracle model and straight-line extractable by observing the Random Oracle calls. For each obtained VC, the extractor can predict for which indices extraction will fail, and assuming preimage resistance of the Random Oracle the number of such indices for each VC can be bounded. For the majority of VCs all indices must potentially be extractable (as soundness would otherwise degrade), and their committed VOLE instances can be fully extracted after the VOLE consistency check. Note that after fixing π_1 (and thereby committing to the VOLE correlation shares of the prover), one can therefore straight-line extract the secrets of all random commitments and in particular can extract $\mathbf{w}_\mathtt{r}$, which in our construction is identical to the secrets of the first ℓ generated correlations/commitments. Hence, the extractor E_1 works straight-line and identically to the first steps of the extractor of [7, Theorem 1]. Given the remaining shares of the VOLE correlations and all observed values d_i (from π_2) as well as the soundness of assert, which remains unchanged as our circuit evaluation is basically unchanged compared to [6–8], one can thereby also construct E_2 which outputs $\mathbf{w}$. However, we must also consider the impact of the added commitment $\mathtt{com}_{\mathsf{rand}}$. Here, as in the existing proof, we can abort if the adversary either never made a query to H to generate $\mathtt{com}_{\mathsf{rand}}$ or found a collision of inputs that generates the same $\mathtt{com}_{\mathsf{rand}}$. Given that $\mathtt{com}_{\mathsf{rand}}$ has length 2λ bits, this probability is negligible. As the extraction strategy does not change, the soundness error essentially remains unchanged. The added soundness loss to ensure the collision resistance for $\mathtt{com}_{\mathsf{rand}}$ is minimal, as the knowledge extractor of [7, Theorem 1] must already handle collisions of random oracle outputs for strings of equal length in other places.