Modelings for generic PoK and Applications: Shorter SD and PKP based Signatures

Slim Bettaieb[1], Loïc Bidoux[1], Philippe Gaborit[2], and Mukul Kulkarni[1]

[1] Cryptography Research Center, Technology Innovation Institute, UAE
{slim.bettaieb,loic.bidoux,mukul.kulkarni}@tii.ae
[2] XLIM, University of Limoges
philippe.gaborit@unilim.fr

Abstract. The Multi-Party Computation in the Head (MPCitH) paradigm has proven to be a versatile tool to design proofs of knowledge (PoK) based on variety of computationally hard problems. For instance, many post-quantum signatures have been designed from MPC based proofs combined with the Fiat-Shamir transformation. Over the years, MPCitH has evolved significantly with developments based on techniques such as threshold computing and other optimizations. Recently, Vector Oblivious Linear Evaluation (VOLE) and the VOLE in the Head framework has spurred further advances. In this work, we introduce three VOLE-friendly modelings for generic and communication efficient PoK to prove the knowledge of secret witness in the form of elementary vectors, vectors of Hamming weight at most ω, and permutation matrices. Remarkably, these modelings scale *logarithmically* with respect to the original witness sizes. Specifically, our modeling for elementary vectors of size n transforms the witness size to $\mathcal{O}(\log_2(n))$, in case of vectors of size n and Hamming weight at most ω the reduced witness is of size $\mathcal{O}\left(\omega \log_2(n)\right)$ while our modeling for permutation matrix of size $n \times n$ results in an (equivalent) witness of size $\mathcal{O}(n \log_2(n))$, which leads to small proofs in practice. To achieve this, we consider modelings with higher multiplicative depth $d > 2$. Even if this increases the proof size, we show that our approach compares favorably with existing proofs. Such design choices were mostly overlooked in previous comparable works, maybe because prior to the VOLEitH framework, multiplications were often emulated with Beaver's triples causing the proof size to scale poorly with d. We also provide several applications for our modelings namely i) post-quantum signature schemes based on the SD (Syndrome Decoding) problem and PKP (Permuted Kernel Problem), ii) PoK of secrets key for code-based key encapsulation mechanism (KEM), and iii) ring signatures from SD and PKP. Our signatures based on SD over $\mathbb{F}_2$ and PKP feature sizes of 3.9 kB and 3.6 kB for NIST-I security level which is respectively 26% and 38% shorter than state-of-the-art alternatives. Our PoK of secret key of BIKE and HQC are twice shorter than similar PoK for Kyber. In addition, we obtain the smallest ring signature based on SD and the first ring signature based on PKP.

1 Introduction

Introduced in 1985 [GMR85,GMR89], Zero-Knowledge Proofs of Knowledge (ZK PoK) have been extensively used in cryptography through applications such as authentication systems, digital signatures [FS87,FFS88], CCA-secure encryption schemes [NY90], variety of privacy-preserving schemes (group signatures [Cv91], ring signatures [RST01] etc.), verifiable computing, decentralized storage, voting systems or blockchains. One of the most popular applications of ZK PoK is digital signatures. Digital signatures can be constructed by Fiat-Shamir transformation [FS87] of an interactive public-coin honest-verifier zero-knowledge PoK system. Informally, a ZK PoK for a NP relation R is an interactive protocol that allows a prover to demonstrate knowledge of a witness w for some public statement x such that $(x, w) \in R$, without revealing any additional information about the witness.

The MPC-in-the-Head (MPCitH) paradigm introduced in [IKOS07,IKOS09] is a generic method to design ZK proofs for NP relations using techniques from secure Multi-Party Computation (MPC). Informally, the MPC protocol is used to compute the verification of an NP relation and its privacy guarantee is used to achieve the zero-knowledge property. In order to do so, the prover splits its witness into N parties using a secret sharing scheme. Then, it simulates locally ("in-its-head") all the parties of the MPC protocol and commits to their views in the MPC protocol. The verifier chooses a random subset of $N' < N$ parties and asks to reveal their corresponding views. This allows the verifier to check that the views of the revealed parties are consistent with each other and consistent with an honest execution of the MPC protocol yielding the expected output.

VOLE-in-the-Head (VOLEitH) introduced in [BBD$^+$23b] is another recent paradigm to design efficient ZK PoK systems. It uses preprocessed random correlations (Vector Oblivious Linear Evaluations – VOLE [BCGI18]) in commit-and-prove framework since the VOLEs can be seen as information-theoretically secure commitments. The linear homomorphic property of VOLE commitments enables constructions of efficient protocols. The main idea is to commit to the witness using VOLE and then compute the linear relation using the homomorphic properties of the commitments. Quicksilver [YSWW21] showed an efficient way to check relations represented as low degree polynomials using VOLE correlations generated by an ideal functionality. Later, Softspoken [Roy22] described how to share VOLE correlations using GGM tree ($N - 1$ out of N OTs). However, this approach was somewhat limited as it operated in the designated verifier setting, which meant that signatures could not be built as the Fiat-Shamir transformation could not be applied (because it needs public-coin PoK and designated verifier implies that the verifier holds secret state). This barrier was removed in [BBD$^+$23b] which built upon the techniques introduced in Quicksilver and Softspoken without relying on the designated verifier setting thus giving public-coin efficient ZK PoK from VOLE.

In response to the threat posed by quantum computing, the NIST has launched a Post-Quantum Cryptography standardization effort in 2016. Several signatures designed using the MPCitH, TCitH (Threshold Computation in the Head, an improvement over standard MPCitH relying on threshold secret sharing) paradigms have been proposed in this context. These signatures relies on various security assumptions such as the Multivariate Quadratic (MQ) problem [FR23a], the Syndrome Decoding (SD) problem [AFG$^+$23], the Permuted Kernel Problem (PKP) [ABB$^+$23a], the MinRank problem [ABB$^+$23c, ARV$^+$23, BFG$^+$24], the Rank Syndrome Decoding (RSD) problem [ABB$^+$23b] or other assumptions [ZCD$^+$20, BKPV23, KCC$^+$23]. Notably, the approach presented by [BBD$^+$23b] was used to design the FAEST signature scheme [BBd$^+$23a] which is the only candidate scheme using VOLEitH framework.

Contributions. In this work, we consider the following question: What kind of VOLE-friendly modelings will be useful in constructions of generic and communication efficient PoK? Such proofs can constitute versatile building blocks which support the design of privacy preserving protocols or short post-quantum signatures. To answer this question we provide: i) a modeling to prove the knowledge of a secret permutation and, ii) a modeling to prove the knowledge of a secret vector of Hamming weight at most ω. Our new modelings are based on a simpler modeling to prove the knowledge of a secret elementary vector (binary vector of Hamming weight 1). We present several applications of our modelings, namely shorter signatures based on the SD and PKP problem, short PoK of secret keys of code-based Key Encapsulation Mechanisms (KEMs) and short ring signatures; which demonstrate their versatility and utility in designing cryptographic protocols.

- *Modelings and resulting PoK.* Our main motivation is to find modelings which can lead to efficient TCitH or VOLEitH signatures from SD or PKP since this has been a challenging task for the post-quantum cryptography researchers over the years. In order to achieve it, we design three VOLE-friendly modelings to prove the knowledge of secret i) elementary vectors, ii) vectors of Hamming weight at most ω, and iii) permutations. Interestingly, our modelings feature a *logarithmic* scaling with respect to their witness size. Therefore, our modelings lead to compressed representation of the secret witnesses which translates in shorter proof sizes. Specifically, our modeling for elementary vectors of size n uses an (equivalent) representative witness of size $\mathcal{O}(\log_2(n))$, our modeling for vectors of size n and Hamming weight at most ω translates to an (equivalent) representative of size $\mathcal{O}(\omega \log_2(n))$ while our modeling for permutation matrix of size $n \times n$ results in an (equivalent) witness of size $\mathcal{O}(n \log_2(n))$, which leads to small proofs in practice. To achieve this, we consider modelings with multiplicative depth $d > 2$. With the notable exception of [BBM$^+$24], almost all existing modelings used in the design of post-quantum cryptographic schemes use multiplicative depth $d = 2$. This might be explained by the fact that, prior to the introduction of the VOLEitH framework, multiplications were often emulated with Beaver's triples [Bea92] causing the proof size to scale poorly with d. As a consequence, our modelings are instantiated using degree-d VOLE correlations. Intuitively this seems counter-productive since increasing the degree of the considered VOLE correlations increases the size of the proof, however we explore an interesting trade-off between multiplicative depth and witness sizes, as illustrated by our applications that outperform existing constructions. In addition, we emphasize that our modelings are generic enough to find many applications beyond our initial goal of constructing shorter post-quantum signatures. Indeed, elementary vectors and permutations are used extensively in privacy preserving protocols (for example to achieve anonymity via secret shuffling), and verifiable computation (working with one out of many values secretly).

- *Shorter* SD *and* PKP *based signatures.* We design PoK of a solution of the SD, and PKP instances based on our modelings for vector of Hamming weight at most ω, and for permutation matrices, respectively. We obtain short signatures based on the SD and PKP problems, by instantiating our modelings in the VOLEitH framework and applying the Fiat-Shamir transformation. As shown in Table 1, our signatures improve the SDitH and PERK schemes currently considered in the NIST Post Quantum Standardization project. Indeed, our signatures based on SD are 26% shorter than the best signature from SD [FJR22, FR23b] while our signature based on PKP are 31% to 38% shorter than PERK depending on the desired security level and do not require the relaxed version of PKP assumption r-IPKP considered in [ABB+23a].

Security	**Assumption**	**Scheme**	sk	pk	**Signature**
NIST-I	SD	SDitH [AFG+23]	0.4 kB	0.1 kB	**8.5 kB**
		[FJR22, FR23b]	16 B	0.1 kB	**5.3 kB**
		This work	16 B	0.1 kB	**3.9 kB**
	r-IPKP	PERK [ABB+23a]	16 B	0.2 kB	**5.8 kB**
	PKP	This work	16 B	0.1 kB	**3.6 kB**
NIST-V	SD	SDitH [AFG+23]	0.8 kB	0.2 kB	**33.9 kB**
		[FJR22, FR23b]	32 B	0.2 kB	**21.9 kB**
		This work	32 B	0.2 kB	**16.2 kB**
	r-IPKP	PERK [ABB+23a]	32 B	0.5 kB	**23.0 kB**
	PKP	This work	32 B	0.2 kB	**15.9 kB**

Table 1: Comparison of our signatures with respect to SDitH and PERK

- *Short PoK of secret keys of code-based KEMs.* Our PoK for SD can also be used to prove the knowledge of the secret key associated with the code-based KEMs BIKE [ABB+22] and HQC [AAB+22]. As both BIKE and

HQC rely on codes whose length is a magnitude of order bigger than the code length used to design signatures, this setting highlights the advantage of our modeling namely its logarithmic scaling with respect to the witness size. Taking HQC as an example, our PoK is 73% to 82% shorter than proofs that can be constructed from the best modeling available in the literature depending on the considered security level (see Table 8). In particular, we obtain proof-size of 8.5 kB for HQC with NIST-I security level while similar proof for Kyber [SAB+22] requires 17.8 kB [GHL+22]. This is significant as it suggests that advanced protocols using PoK of secret keys of KEM (KEMTLS [SSW20], verifiable encryption) may have a smaller footprint when instantiated with code-based KEMs instead of lattice-based KEMs.

- *Shorter ring signature.* We propose two ring signatures by extending our signatures based on the SD and PKP problems. In order to achieve this, we modify these signatures so that they use a secret keypair $(\mathsf{pk}_{i^*}, \mathsf{sk}_{i^*})$ where i^* denotes the index of the signer within the ring of $\bar{n}$ users whose public key is defined as $\mathsf{pk}_{\mathcal{R}} = (\mathsf{pk}_0, \ldots, \mathsf{pk}_{\bar{n}-1})$. The public key pk_{i^*} can be computed from the ring key $\mathsf{pk}_{\mathcal{R}}$ along with the secret position i^* using our modeling for elementary vectors. We obtain the shortest ring signatures based on the SD problem (see Table 9) as our signatures are 29% to 41% shorter than existing ones depending on the considered ring size. In addition, to the best of our knowledge, we provide the first ring signature based on PKP.

Concurrent work. Hereafter, we briefly discuss a concurrent work studying code-based PoK from VOLEitH [OTX24]. In this work, a modeling to prove the correctness of a regular encoding process is described and used to improve the commitment scheme, accumulator scheme, ring signature and group signature constructions from [NTWZ19] and the fully dynamic attribute-based signature from [LNP+24]. In addition, a PoK to prove the knowledge of a plaintext of a variant of the McEliece scheme [McE78] and a signature based on the Regular SD problem are provided. Although their modeling for regular encoding is based on boolean functions and our modeling for elementary vectors is based on a binary tree construction, both produce proofs of similar size which is not entirely surprising given that regular words and elementary vectors describe the same objects (vectors with only one non zero coordinate equal to one). Except for this similarity, the two works differ in terms of scope and goals. They use their modeling to improve many code-based constructions based on the Regular SD problem (a variant of the SD problem with additional structure). Even if the SD problem has been studied extensively over the years, the Regular SD variant has not received the same amount of scrutiny yet and best known attacks have been improved recently [BØ23, ES24]. While elementary vectors can be used by themselves in many applications, our modeling for elementary vectors is mainly used as a building block to design other modelings. As a consequence, we consider a different set of applications or whenever there is an overlap between considered primitives, we build them from different security assumptions. Interestingly, our PoK for the SD problem is versatile enough to handle both the (unstructured) SD problem and the (structured) Regular SD problem (we exploit this property in the design of our signature from SD, see Section 4 for additional details). Thus, applications targeted in [OTX24] can also be constructed from our work while the opposite is not true.

Paper organization. We provide background on several hard problems used in post-quantum cryptography and the VOLEitH paradigm in Section 2. Next, we present our modellings and their associated PoK in Section 3. We describe our signatures based on the SD and PKP problems in Section 4. The application of our modellings to PoK of secret keys of code-based KEMs is discussed in Section 5 while their application to ring signatures is detailed in Section 6.

2 Preliminaries

Notations and conventions

Let a positive integer λ be the security parameter. For integers i, j let $[i, j]$ denote the set of integers k such that $i \leq k \leq j$ and let $[n]$ be a shorthand for $[1, n]$. For a vector $\boldsymbol{x}$ its Hamming weight is defined as the number of non-zero coordinates, and we denote the Hamming weight of $\boldsymbol{x}$ by $w_H\big(\boldsymbol{x}\big)$. Let $\mathcal{S}_n$ denote the group of permutations of the set $[n]$ and let $\mathcal{S}^n_\omega(\mathbb{F}_2)$ denote the set of vector of $\mathbb{F}_2^n$ of Hamming weight at most ω. Let $\mathbb{F}_q$ denote the finite field of q elements where q is the power of a prime. If X is a finite set, let $x \xleftarrow{\$} X$ denote that x is chosen uniformly at random from X. Similarly, let $x \xleftarrow{\$,\theta} X$ denote that x is sampled pseudo-randomly from the set X based on the seed θ. Vectors are denoted by bold lower-case letters and matrices by bold capital letters (e.g., $\boldsymbol{v} = (v_i)_{i\in[n]} \in \mathbb{F}_q^n$ and $\boldsymbol{A} = (a_{ij})_{i\in[m],j\in[n]} \in \mathbb{F}_q^{m\times n}$). Let PRG denote a pseudo-random generator.

2.1 Hard problems for cryptography

We give the definitions of the computationally hard problems underlying the security of our proposed signature schemes namely the Permuted Kernel Problem (PKP) and the Syndrome Decoding (SD) problem. We also define the Regular SD problems which is a variant of the SD problem.

Definition 2.1 (Permuted Kernel Problem (PKP)). *Let (q, m, n) be positive integers such that $m < n$, $\boldsymbol{H} \in \mathbb{F}_q^{m\times n}$, $\boldsymbol{x} \in \mathbb{F}_q^n$ and $\pi \in \mathcal{S}_n$ be a permutation such that $\boldsymbol{H}\big(\pi[\boldsymbol{x}]\big) = \boldsymbol{0}$. Given $(\boldsymbol{H}, \boldsymbol{x})$, the Permuted Kernel Problem $\mathsf{PKP}(q, m, n)$ asks to find $\tilde{\pi} \in \mathcal{S}_n$ such that $\boldsymbol{H}\big(\tilde{\pi}[\boldsymbol{x}]\big) = \boldsymbol{0}$.*

Hereafter, we interprete the PKP problem in matrix form namely the secret permutation π is seen as a permutation matrix $\boldsymbol{P} \in \mathbb{F}_2^{n\times n}$ such that $\boldsymbol{HPx} = 0$

Definition 2.2 (Syndrome Decoding (SD)). *Let* $(q,\ n,\ k,\ \omega)$ *be positive integers such that* $k < n$, $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\boldsymbol{x} \in \mathbb{F}_q^n$ *such that* $w_H(\boldsymbol{x}) \leq \omega$ *and* $\mathbf{y} \in \mathbb{F}_q^{(n-k)}$ *such that* $\boldsymbol{y} = \boldsymbol{Hx}$*. Given* $(\boldsymbol{H}, \boldsymbol{y})$*, the Syndrome Decoding problem* $\mathsf{SD}(q, n, k, \omega)$ *asks to find* $\tilde{\mathbf{x}} \in \mathbb{F}_q^n$ *such that* $w_H(\tilde{\boldsymbol{x}}) \leq \omega$ *and* $\boldsymbol{H}\tilde{\boldsymbol{x}} = \boldsymbol{y}$*.*

Definition 2.3 (Regular SD). *Let* $(q,\ n,\ k,\ \omega)$ *be positive integers such that* $k < n$ *and* $\omega \,|\, n$, $\boldsymbol{H} \in \mathbb{F}_q^{(n-k)\times n}$, $\boldsymbol{x} = (\boldsymbol{x}_1, \cdots, \boldsymbol{x}_\omega) \in \mathbb{F}_q^n$ *such that* $\forall i \in [\omega]$, $\boldsymbol{x}_i \in \mathbb{F}_q^{n/\omega}$, $w_H(\boldsymbol{x}_i) = 1$ *and* $\mathbf{y} \in \mathbb{F}_q^{(n-k)}$ *such that* $\boldsymbol{y} = \boldsymbol{Hx}$*. Given* $(\boldsymbol{H}, \boldsymbol{y})$*, the Regular* $\mathsf{SD}(q, n, k, \omega)$ *asks to find* $\tilde{\mathbf{x}} = (\tilde{\boldsymbol{x}_1}, \cdots, \tilde{\boldsymbol{x}_\omega}) \in \mathbb{F}_q^n$ *such that* $\forall i \in [d]$, $\tilde{\boldsymbol{x}_i} \in \mathbb{F}_q^{n/\omega}$, $w_H(\tilde{\boldsymbol{x}_i}) = 1$ *and* $\boldsymbol{H}\tilde{\boldsymbol{x}} = \boldsymbol{y}$*.*

2.2 Overview of the VOLEitH framework

VOLE correlations. A VOLE correlation over a finite field $\mathbb{F}_{2^\kappa}$ are random values $u \in \mathbb{F}_2$, and $v, \Delta, q \in \mathbb{F}_{2^\kappa}$, such that $q = u\Delta + v$. This can be generalized to obtain VOLE correlations of arbitrary length ℓ where, $\boldsymbol{u} \in \mathbb{F}_2^\ell$, $\boldsymbol{v}, \boldsymbol{q} \in \mathbb{F}_{2^\kappa}^\ell$, and $\Delta \in \mathbb{F}_{2^\kappa}$ such that $q_i = u_i + \Delta v_i$ for $i \in [\ell]$. Sometimes we write this in vector notation as $\boldsymbol{q} = \boldsymbol{u}\Delta + \boldsymbol{v}$. The values $\boldsymbol{u}, \boldsymbol{v}$ are given to the prover, and the verifier is given $\Delta, \boldsymbol{q}$. As the values are chosen randomly and split between the two parties, the VOLE correlations can be used as linearly homomorphic commitments. Informally, the relation $\boldsymbol{q} = \boldsymbol{u}\Delta + \boldsymbol{v}$ serves as commitment by prover to value $\boldsymbol{u}$ with a random mask $\boldsymbol{v}$. Since the verifier does not know $\boldsymbol{v}$, the commitment is hiding. In order to provide an opening $\boldsymbol{v}'$ for some $\boldsymbol{u}' \neq \boldsymbol{u}$ such that $\boldsymbol{q} = \boldsymbol{u}'\Delta + \boldsymbol{v}'$, the prover will have to guess Δ which provides binding. The linear homomorphic property follows from the fact that the VOLE correlation is linear. In the following, we denote the prover by P and the verifier as V.

In an ideal scenario, the VOLE correlation $\boldsymbol{q} = \boldsymbol{u}\Delta + \boldsymbol{v}$, with $\boldsymbol{u} \in \mathbb{F}_2^\ell$, $\boldsymbol{v}, \boldsymbol{q} \in \mathbb{F}_{2^\kappa}^\ell$, and $\Delta \in \mathbb{F}_{2^\kappa}$ would have been generated by a trusted party (or an ideal functionality $\mathcal{F}_{\mathsf{VOLE}}$) which would then send $\boldsymbol{u}, \boldsymbol{v}$ to the prover and $\boldsymbol{q}, \Delta$ to the verifier privately. Looking ahead, all our protocols and (interactive) proof of knowledge systems are secure in the so-called $\mathcal{F}_{\mathsf{VOLE}}$-hybrid model, which assumes existence of such ideal functionality.

In practice, the VOLE correlations are generated by one of the parties (say the prover) with the help of puncturable pseudorandom functions (puncturable PRF), which are instantiated from pseudorandom generators (PRG) using GGM trees [GGM84]. The GGM tree allows the prover to commit to N PRG seeds arranged as leaves of a binary tree, by simply committing to the root of the tree by computing its hash using a collision-resistant hash function. More importantly, after committing to these N seeds, the prover can open $N - 1$ of those seeds (all except one) by sending only $O(\log n)$ bits (essentially siblings of nodes on path from root to the hidden leaf). This allows the verifier to reconstruct the tree (except the hidden leaf node) and after obtaining the auxiliary information (commitment to the hidden node) from the prover, the verifier can recompute the commitment to the root and check if it matches, thus verifying that the opened $N - 1$ seeds are consistent with those committed by the prover. The

VOLE correlations can be obtained from the seeds. Let $\boldsymbol{r}_i = \mathsf{PRG}(\mathsf{seed}_i)$ be ℓ-bit pseudorandom strings for $i \in [0, N-1]$. The prover computes $\boldsymbol{u}, \boldsymbol{v}$ as

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

The verifier chooses $\Delta \in [0, N-1]$ uniformly at random and receives all the seeds seed_i for $i \in [0, N-1] \setminus \Delta$ from the prover. The verifier can then compute

$$\boldsymbol{q} = \sum_{i=0}^{N-1} (\Delta - i) \cdot \boldsymbol{r}_i.$$

Note that the prover should be given Δ (to receive all the remaining seeds) only after the prover has committed to all the seeds and main steps which rely on the binding of VOLE commitments are performed since knowing Δ trivially allows him to cheat. We denote the above process by GenAuthVOLE to indicate that prover and verifier have received their respective authentic shares for VOLE correlation. And from now onward, we assume that for protocols and schemes used, the prover and verifier always start with authentic VOLE shares as inputs.

Operations on VOLE correlations. Algorithm LinearCombination (see Appendix A) shows how VOLE correlations for linear functions of secret values $u_1, \ldots, u_n \in \mathbb{F}_2$ can be computed as given in [BBd+23a]. Standard VOLE correlation corresponds to degree-1 commitment to u given by $f_u(X) = uX + v$, with the evaluation $q = f_u(\Delta)$ given to the verifier. We denote degree-1 commitment to s by $[\![s]\!]$. Following the notation of [BBM+24], we write $[\![s]\!]^{(d)}$ to denote a degree-d commitment to a secret value $s \in \mathbb{F}_2$ where the prover holds $f_s(X) = \sum_{i=0}^{d} a_i X^i$ with coefficients $a_i \in \mathbb{F}_{2^\kappa}$ and a_d equal to s lifted to $\mathbb{F}_{2^\kappa}$ while the verifier holds $q_s = f_s(\Delta) \in \mathbb{F}_{2^\kappa}$. It is possible to perform homomorphic operations on the degree-d-commitments locally by the prover and the verifier with the help of the Algorithms Add and Multiply given in Appendix A.

Let $[\![w]\!]^{(d)}$ be a degree-d-commitment to some secret value $w \in \mathbb{F}_2$ with $f_w(X) := a_0 + a_1 X + \cdots + a_d X^d$, where $a_i \in \mathbb{F}_{2^\kappa}$ for $i \in [0, d-1]$ and $a_d = w$ lifted to $\mathbb{F}_{2^\kappa}$. Algorithms P.CheckZero and V.CheckZero can be used to check that the secret value $w = 0$.

Algorithm 2.1: P.CheckZero$\left([\![w]\!]^{(d)}, ([\![s_i]\!])_{i \in [0,d-2]}\right)$

Public information and inputs

Public information: Degree of input VOLE correlations d.

Prover's input: Degree-d VOLE correlation $[\![w]\!]^{(d)}$, $(d-1)$ random VOLE correlation represented as $f_{s_i}(X) = r_i + s_i X$ where both $r_i, s_i \in \mathbb{F}_{2^\lambda}$ for $i \in [0, d-2]$.

Output

Prover's output: VOLE correlation $[\![a]\!]$.

```
1 :  $f_{\mathsf{mask}}(X) = \sum_{i=0}^{d-2} f_{s_i}(X) \cdot X^i$
2 :  $[\![a]\!] = f_w(X) + f_{\mathsf{mask}}(X)$
3 :  return $[\![a]\!]$
```

Algorithm 2.2: V.CheckZero$\left([\![a]\!]^{(d)}, q_w, (q_{s_i})_{i\in[0,d-2]}, \Delta\right)$

Public information and inputs

Public information: Degree of input VOLE correlations d.

Verifier's input: VOLE correlation $[\![a]\!]$, $q_a, q_{s_i} = f_{s_i}(\Delta)$ for $i \in [0, d-2]$, Δ.

Output

Verifier's output: Boolean indicating if leading coefficient of f_a is equal to zero or not.

```
1 :  $q = q_w + \sum_{i=0}^{d-2} q_{s_i} \cdot \Delta^i$
2 :  $b \leftarrow (q \overset{?}{=} f(\Delta))$
3 :  return $b$
```

VOLEitH framework. Given a VOLE correlation $q = u\Delta + v$ for a random $u \in \mathbb{F}_2$, it is possible to embed arbitrary value $w \in \mathbb{F}_2$. To do so, the prover computes $t = w \oplus u$ and sends t to the verifier. Since u is uniform random and unknown to the verifier, t does not leak any information about w. The verifier then computes $q_w = q + t\Delta$. Note that the prover and verifier now possess their respective parts for the VOLE correlation $q_w = w\Delta + v$. In the rest of the paper, we will assume that the prover and verifier receive the VOLE shares corresponding to values w of prover's choice, since they can receive the VOLE correlations for random u from the ideal functionality (realized using vector commitments based on GGM tree) and then embed any value of prover's choice.

After embedding the witness w, one can use the VOLE correlations to establish PoK of the witness for relations which can be modeled as linear functions of the witness. These proofs provide soundness error of $|\mathbb{F}_{2^\kappa}|^{-1}$. And therefore, soundness amplification is needed to achieve the negligible soundness error by repeating the protocol (say τ times). This however increases communication cost since we need to send the VOLE correlations for τ repetitions. An optimization introduced in [BBd+23a] allows to reduce this communication by sending one VOLE correlation input (say $\boldsymbol{u}_0$) and only corrections for remaining $\tau - 1$ VOLE inputs with respect to the $\boldsymbol{u}_0$. However, since the prover can cheat while sending corrections, the verifier needs to check for consistency of all the received VOLE inputs $\boldsymbol{q}_0, \ldots, \boldsymbol{q}_{\tau-1}$ by checking random linear combination of the inputs. For more details, we refer the interested readers to Section 2.2 of [BBd+23a]. We denote the above process by VOLEConsistencyCheck to indicate that prover and verifier have carried out the consistency check by taking random linear combinations, successfully, on their authentic VOLE inputs.

VOLE correlations provide a powerful tool to construct efficient PoK and signatures as depicted in Algorithm 2.3 which is an abstraction of the design of the FAEST signature scheme [BBd+23a]. The framework can be split into parts that are problem independent thus can be designed and used generically, which allow all the proof systems based on the framework to design a unified interfaces. Also, the schemes can benefit from any optimization or technical improvements in this parts in problem-agnostic way (see for instance the GGM related optimization from [BBM+24]). This allows PoK designers to focus on the modeling associated to the problem they are addressing. In the next section, we focus on the problem specific design part and then instantiate our modeling with the VOLEitH framework in Sections 4, 5 and 6.

```
Algorithm 2.3: VOLE Based PoK Framework

[Problem independent] - Computing commitments
- Generate VOLE correlations using GenAuthVOLE to achieve targeted soundness.
- Commit to the VOLE correlations and send corrections with respect to $\boldsymbol{u}_0$.
- Run VOLEConsistencyCheck to ensure all VOLE correlations are consistent with $\boldsymbol{u}_0$.

[Problem dependent] - Computing Proof
- Prover P embeds the witness $w$ in $\boldsymbol{u}_0$ then compute linear relations on VOLE inputs to
  obtain its proof (Verifier V will perform analogous steps during verification).

[Problem independent] Opening commitments
- Send openings to commitments, auxiliary information for verification of consistency
  checks, and proof to the verifier.
```

3 Modelings for compact PoK of elementary vectors, vectors of Hamming weight at most ω and permutations

We present modelings that can be used to prove the knowledge of an elementary vector (a vector whose coordinates are all equal to "0" except one coordinate that is equal to "1"), the knowledge of a vector over $\mathbb{F}_2$ of Hamming weight at most ω and the knowledge of a permutation matrix. Our modelings leverage compressed representation of the secret witnesses which translates in shorter proof sizes. Specifically, our modeling for elementary vectors of size n generates an (equivalent) representative witness of size $\mathcal{O}(\log_2(n))$, our modeling for vectors of size n and Hamming weight at most ω translates to an (equivalent) representative of size $\mathcal{O}(\omega \log_2(n))$ while our modeling for permutation matrix of size $n \times n$ results in an (equivalent) witness of size $\mathcal{O}(n \log_2(n))$.

In addition, we show how to model the linear relations satisfied by the witness of our modelings in VOLE-friendly objects. Generally, this boils down to starting from VOLE correlations (which can be seen as degree-1 or linear polynomials) corresponding to the witness and computing low-degree polynomials from these

VOLE correlations, which will correspond to intermediate steps computing the relation that is being checked, leading to the final VOLE correlations, which can help verify the validity of the witness. We describe how the modeling can be used by the prover and the verifier to locally update their VOLE correlation inputs and obtain the VOLE correlations corresponding to the final result.

3.1 Modeling for PoK of elementary vectors

Hereafter, we present a modeling that can be used to design a ZK PoK that a vector is elementary namely $\boldsymbol{v}$ has one coordinate equal to 1 and all its other coordinates equal to 0. This modeling only requires inputs of size $\log(n)$.

Notations. Let n be a positive integer and let $d = \log_2(n)$. Let $\mathcal{B} : [0, n-1] \to \mathbb{F}_2^d$ be the function that given a value $x \in [0, n-1]$ returns the vector in $\mathbb{F}_2^d$ corresponding to its binary representation which we call $\mathcal{B}^x$. Let $\overline{x} : \mathbb{F}_2^d \to \mathbb{F}_2^d$ be the function that given a vector $\boldsymbol{x} \in \mathbb{F}_2^d$ returns its complement namely $\overline{\boldsymbol{x}} = (1 \oplus x_0, \cdots, 1 \oplus x_{d-1})$.

Modeling. Let $\mathsf{pos} \in [0, n-1]$ be the position of the 1 in the secret vector. Given as input $\boldsymbol{w} = \mathcal{B}^{\mathsf{pos}} \in \mathbb{F}_2^d$, compute $\boldsymbol{z} \in \mathbb{F}_2^n$ as:

$$\forall j \in [0, n-1],\ z_j = \prod_{k=0}^{d-1} \left(\overline{\mathcal{B}_k^j} \oplus w_k\right).$$

We now explain why such a characterization leads to the expected result. Let's start by building a binary tree with n leaves (assuming that n is a power of two for simplicity) and depth $d = \log(n)$. In such a graph, each node (except the leaves) spans two children nodes. We label the corresponding edges with 0 and 1 respectively. In addition, we associate to each leaf the vector in $\mathbb{F}_2^d$ whose coordinates are the label values of the path from the root node to the considered leaf. Doing so, one can see in Figure 1 that whenever leaves are labelled (from left to right) from 0 to $n-1$, each leaf i is associated with a unique vector corresponding to $\mathcal{B}^i$ the binary representation of i.

As we are working over $\mathbb{F}_2$, it is readily seen that adding $\mathcal{B}^i$ to its complement $\overline{\mathcal{B}^i}$ leads to the vector $\mathbf{1}_d = (1, \cdots, 1) \in \mathbb{F}_2^d$ while adding any other vector to $\overline{\mathcal{B}^i}$ results in a vector having at least one zero coordinate. As such, given a vector $\boldsymbol{w} \in \mathbb{F}_2^d$, the product of the coordinates of $(\overline{\mathcal{B}^i} \oplus \boldsymbol{w})$ is equal to 1 when $\boldsymbol{w} = \mathcal{B}^i$ and 0 otherwise. Hence, given as input $\mathsf{pos} \in [0, n-1]$ and $\boldsymbol{w} = \mathcal{B}^{\mathsf{pos}} \in \mathbb{F}_2^d$, the vector $\boldsymbol{z} \in \mathbb{F}_2^n$ constructed as $\forall j \in [0, n-1],\ \boldsymbol{z}_j = \prod_{k=0}^{d-1} \left(\overline{\mathcal{B}_k^j} \oplus w_k\right)$ contains a 1 at position pos and 0's in all its remaining coordinates.

We can use this idea to create degree-d VOLE correlations $[\![\boldsymbol{z}]\!]^{(d)}$ of an elementary vector given a secret position pos. Looking ahead, these VOLE correlations will be used in several protocols leveraging ZK PoK to verify the knowledge of some secret held by a prover P. Algorithm P.VOLE-ElementaryVector

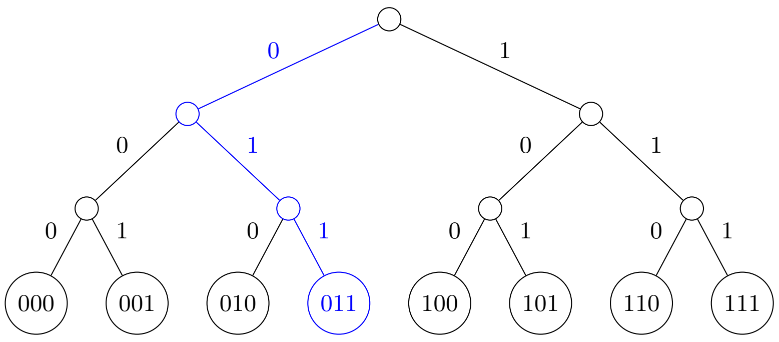

Fig. 1: Binary tree of depth $d = 3$ with $n = 8$ leaves. Each leaf contains the vector associated to its path which is also the binary representation of its number whenever leaves are labelled from left to right from 0 to $n-1$. In blue, the 4^{th} leaf (leaf number 3 starting from 0) is associated with $\mathcal{B}^3 = (011)$.

shows how to construct the VOLE correlation $[\![\boldsymbol{z}]\!]^{(d)}$ with help of d random VOLE correlations $([\![\boldsymbol{u}_k]\!])_{k\in[0,d-1]}$. Its output contains the masked secret $\boldsymbol{t}$ derived from the position pos. The verifier can use V.VOLE-ElementaryVector to retrieve the VOLE correlations $q_{\boldsymbol{z}}$ associated to $[\![\boldsymbol{z}]\!]^{(d)}$ using the VOLE correlations $(q_{\boldsymbol{u}_k})_{k\in[0,d-1]}$ associated $([\![\boldsymbol{u}_k]\!])_{k\in[0,d-1]}$ after receiving the masked secret $\boldsymbol{t}$ from the prover.

Algorithm 3.1: P.VOLE-ElementaryVector$\big(\mathsf{pos}, ([\![u_k]\!])_{k\in[0,d-1]}\big)$

Public information and inputs

Public information: Length of the vector n, $d = \lceil \log n \rceil$.

Prover's input: Secret position $\mathsf{pos} \in [0, n-1]$, d VOLE correlations $[\![u_k]\!]$ for random $(u_k, v_k) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$ represented as polynomials $f_{u_k}(X) = u_k X + v_k$ for $k \in [0, d-1]$.

Output

Prover's output: Masked secret $\boldsymbol{t} \in \mathbb{F}_2^d$, degree-$d$ VOLE correlations $[\![\boldsymbol{z}]\!]^{(d)}$.

```
     Compute masked secret
1 :  $\boldsymbol{w} = \mathcal{B}^{\mathsf{pos}}$
2 :  $\boldsymbol{t} = \boldsymbol{w} \oplus \boldsymbol{u}$

     Compute elementary vector
3 :  for $j \in [0, n-1]$
4 :    for $k \in [0, d-1]$
5 :      $\beta_k = \overline{\mathcal{B}_k^j} \oplus w_k$
6 :      $f_{\beta_k}(X) = \beta_k X + v_k$
7 :    endfor
8 :    $[\![z_j]\!]^{(d)} = \prod_{k=0}^{d-1} f_{\beta_k}(X)$
9 :  endfor

10 : $[\![\boldsymbol{z}]\!]^{(d)} = ([\![z_0]\!]^{(d)}, \ldots, [\![z_{n-1}]\!]^{(d)})$
11 : return $(\boldsymbol{t}, [\![\boldsymbol{z}]\!]^{(d)})$
```

Algorithm 3.2: V.VOLE-ElementaryVector$(\boldsymbol{t}, (q_{u_k})_{k\in[0,d-1]}, \Delta)$

Public information and inputs

Public information: Length of the vector n, $d = \lceil \log n \rceil$.

Verifier's input: Masked secret $\boldsymbol{t} \in \mathbb{F}_2^d$, $q_{u_k} = f_{u_k}(\Delta)$ for $k \in [0, d-1]$, Δ.

Output

Verifier's output: VOLE correlations $q_{\boldsymbol{z}}$.

```
     Compute VOLE correlations for $\boldsymbol{z}$
1 :  for $j \in [0, n-1]$
2 :    for $k \in [0, d-1]$
3 :      $\gamma_k = \overline{\mathcal{B}_k^j} \oplus t_k$
4 :      $q_{\beta_k} = q_{u_k} + \gamma_k \cdot \Delta$
5 :    endfor
6 :    $q_{z_j} = \prod_{k=0}^{d-1} q_{\beta_k}$
7 :  endfor

8 :  $q_{\boldsymbol{z}} = (q_{z_0}, \ldots, q_{z_{n-1}})$
9 :  return $q_{\boldsymbol{z}}$
```

3.2 Modeling for PoK of vectors of Hamming weight at most ω

In this section, we present a modeling for proving that a vector $\boldsymbol{z}$ over $\mathbb{F}_2$ has Hamming weight at most ω namely $\boldsymbol{z} \in \mathcal{S}_\omega^n(\mathbb{F}_2)$. In order to do so, we build upon the technique from Section 3.1 by constructing ω elementary vectors (one for each non-zero coordinate in $\boldsymbol{z}$) and combining them together. One should note that due to the triangle inequality, adding ω vectors of Hamming weight 1 produces a vector of Hamming weight at most ω thus giving the expected result. This modelling uses $\omega \log(n)$ bits as inputs.

Modeling. Let $\boldsymbol{s} = (\mathsf{pos}_0, \ldots, \mathsf{pos}_{\omega-1}) \in [0, n-1]^\omega$ be the secret support of $\boldsymbol{z}$ represented as positions. Given $\boldsymbol{w} = (\mathcal{B}^{\mathsf{pos}_0}, \ldots, \mathcal{B}^{\mathsf{pos}_{\omega-1}}) \in (\mathbb{F}_2^d)^\omega$ as input, compute $\boldsymbol{z}_i \in \mathcal{S}_1^n(\mathbb{F}_2)$ and $\boldsymbol{z} \in \mathcal{S}_\omega^n(\mathbb{F}_2)$ as:

$$\forall (i,j) \in [0, \omega-1] \times [0, n-1],\ z_{i,j} = \prod_{k=0}^{d-1} \left(\overline{\mathcal{B}_k^j} \oplus w_{i,k}\right) \text{ and } z_j = \sum_{i=0}^{\omega-1} z_{i,j}.$$

Algorithms P.VOLE-HammingWeight and V.VOLE-HammingWeight explain how to build the degree-d VOLE correlations $[\![\boldsymbol{z}]\!]^{(d)}$ associated to this modeling using $\omega \cdot d$ random VOLE correlations by describing the prover P and verifier V sides respectively.

Algorithm 3.3: P.VOLE-HammingWeight$\left(\boldsymbol{z}, ([\![u_{i,k}]\!])_{i\in[0,\omega-1],k\in[0,d-1]}\right)$

Public information and inputs

Public information: Upper bound on vector's weight ω, length of the vector n, $d = \lceil \log n \rceil$.
Prover's input: Secret vector $\boldsymbol{z}$ represented as ω positions $(\mathsf{pos}_0, \ldots, \mathsf{pos}_{\omega-1})$, $\omega \cdot d$ VOLE correlations $[\![u_{i,k}]\!]$ for random $(u_{i,k}, v_{i,k}) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$ represented as polynomials $f_{u_{i,k}}(X) = u_{i,k}X + v_{i,k}$ for $(i,k) \in [0, \omega-1] \times [0, d-1]$.

Output

Prover's output: Masked secret $\boldsymbol{t} \in (\mathbb{F}_2^d)^\omega$, degree-$d$ VOLE correlations $[\![\boldsymbol{z}]\!]^{(d)}$.

```
   Compute $\omega$ elementary vectors
1: for $i \in [0, \omega-1]$
2:   $(\boldsymbol{t}_i, [\![\boldsymbol{z}_i]\!]^{(d)}) \leftarrow$ P.VOLE-ElementaryVector$(\mathsf{pos}_i, ([\![u_{i,k}]\!])_{k\in[0,d-1]})$
3: endfor

   Compute $\boldsymbol{z}$ such that $w_H(\boldsymbol{z}) \leq \omega$
4: for $j \in [0, n-1]$
5:   $[\![z_j]\!]^{(d)} = \sum_{i=0}^{\omega-1} [\![z_{i,j}]\!]^{(d)}$
6: endfor

7: $\boldsymbol{t} = (\boldsymbol{t}_0, \cdots, \boldsymbol{t}_{\omega-1})$
8: $[\![\boldsymbol{z}]\!]^{(d)} = ([\![z_0]\!]^{(d)}, \ldots, [\![z_{n-1}]\!]^{(d)})$
9: return $(\boldsymbol{t}, [\![\boldsymbol{z}]\!]^{(d)})$
```

Algorithm 3.4: V.VOLE-HammingWeight$(\boldsymbol{t}, (q_{u_{i,k}})_{i\in[0,\omega-1],k\in[0,d-1]}, \Delta)$

Public information and inputs

Public information: Upper bound on vector's weight ω, length of the vector n, $d = \lceil \log n \rceil$.

Verifier's input: Masked secret $\boldsymbol{t} \in (\mathbb{F}_2^d)^\omega$, $q_{u_{i,k}} = f_{u_{i,k}}(\Delta)$ for $(i,k) \in [0,\omega-1]\times[0,d-1]$, Δ.

Output

Verifier's output: VOLE correlations $q_{\boldsymbol{z}}$.

```
        Compute VOLE correlations for elementary vectors
 1 :    for $i \in [0, \omega - 1]$
 2 :      $q_{\boldsymbol{z}_i} \leftarrow$ V.VOLE-ElementaryVector$(\boldsymbol{t}_i, (q_{u_{i,k}})_{k\in[0,d-1]}, \Delta)$
 3 :    endfor

 4 :    Compute VOLE correlations for $\boldsymbol{z}$
 5 :    for $j \in [0, n - 1]$
 6 :      $q_{z_j} = \sum_{i=0}^{\omega-1} q_{z_{i,j}}$
 7 :    endfor
 8 :
 9 :    $q_{\boldsymbol{z}} = (q_{z_0}, \ldots, q_{z_{n-1}})$
10 :    return $q_{\boldsymbol{z}}$
```

3.3 Modeling for PoK of permutation matrices

A permutation matrix can be seen as a $n \times n$ matrix containing an elementary vector on each row with the additional constraints that all its column-wise sums of coefficients are equals to 1. As a consequence, by using n times the characterization from Section 3.1, one gets a modeling that can be used to prove the knowledge of a permutation matrix using input size $n\log(n)$.

Modeling. Let $\boldsymbol{P} = (\mathsf{pos}_0, \ldots, \mathsf{pos}_{n-1}) \in [0, n-1]^n$ be the secret permutation matrix represented by the positions of its 1 on each row. Given as input the vector $\boldsymbol{w} = (\mathcal{B}^{\mathsf{pos}_0}, \ldots, \mathcal{B}^{\mathsf{pos}_{n-1}}) \in (\mathbb{F}_2^d)^n$, compute $\boldsymbol{z} \in \mathbb{F}_2^{n\times n}$ as:

$$\forall (i,j) \in [0,n-1]\times[0,n-1],\ \boldsymbol{z}_{i,j} = \prod_{k=0}^{d-1} \left(\overline{\mathcal{B}_k^j} \oplus w_{i,k}\right).$$

One can verify that $\boldsymbol{P}$ is a permutation matrix by checking that:

$$\forall i \in [0,n-1], \sum_{j=0}^{n-1} \boldsymbol{z}_{i,j} = 1.$$

This modeling can be instantiated to create VOLE correlations $[\![\boldsymbol{z}]\!]^{(d)}$ using $n \cdot d$ random VOLE correlations as described in Algorithm 3.5 and Algorithm 3.6.

Algorithm 3.5: P.VOLE-Permutation$\big(\boldsymbol{P}, (\llbracket u_{i,k} \rrbracket)_{i\in[0,n-1],k\in[0,d-1]}\big)$

Public information and inputs

Public information: Matrix dimension n of the secret permutation matrix, $d = \lceil \log n \rceil$.

Prover's input: Secret permutation matrix $\boldsymbol{P}$ represented as n positions $(\mathsf{pos}_0, \ldots, \mathsf{pos}_{n-1})$, $n \cdot d$ VOLE correlations $\llbracket u_{i,k} \rrbracket$ for random $(u_{i,k}, v_{i,k}) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$ represented as polynomials $f_{u_{i,k}}(X) = u_{i,k}X + v_{i,k}$ for $(i,k) \in [0, n-1] \times [0, d-1]$.

Output

Prover's output: Masked secret $\boldsymbol{t} \in (\mathbb{F}_2^d)^n$, VOLE correlations $(\llbracket \boldsymbol{z} \rrbracket^{(d)}, \llbracket \mathsf{ColCheck} \rrbracket^{(d)})$.

```
      Compute $\boldsymbol{P}$ row-wise as $n$ elementary vectors
 1 :  for $i \in [0, n-1]$ :
 2 :    $(\boldsymbol{t}_i, \llbracket \boldsymbol{z}_i \rrbracket^{(d)}) \leftarrow$ P.VOLE-ElementaryVector$\big(\mathsf{pos}_i, (\llbracket u_{i,k} \rrbracket)_{k\in[0,d-1]}\big)$
 3 :  endfor

      Compute $\boldsymbol{P}$ columns check
 4 :  for $j \in [0, n-1]$ :
 5 :    $\llbracket \mathsf{ColSum}_j \rrbracket^{(d)} = \sum_{i=0}^{n-1} \llbracket z_{i,j} \rrbracket^{(d)}$
 6 :    $\llbracket \mathsf{ColCheck}_j \rrbracket^{(d)} = \llbracket \mathsf{ColSum}_j \rrbracket^{(d)} - 1 \cdot X^d$
 7 :  endfor

 8 :  $\boldsymbol{t} = (\boldsymbol{t}_0, \cdots, \boldsymbol{t}_{n-1})$
 9 :  $\llbracket \boldsymbol{z} \rrbracket^{(d)} = (\llbracket \boldsymbol{z}_0 \rrbracket^{(d)}, \ldots, \llbracket \boldsymbol{z}_{n-1} \rrbracket^{(d)})$
10 :  $\llbracket \mathsf{ColCheck} \rrbracket^{(d)} = (\llbracket \mathsf{ColCheck}_0 \rrbracket^{(d)}, \ldots, \llbracket \mathsf{ColCheck}_{n-1} \rrbracket^{(d)})$
11 :  return $(\boldsymbol{t}, \llbracket \boldsymbol{z} \rrbracket^{(d)}, \llbracket \mathsf{ColCheck} \rrbracket^{(d)})$
```

Algorithm 3.6: V.VOLE-Permutation$\big(\boldsymbol{t}, (q_{u_{i,k}})_{i\in[0,n-1],k\in[0,d-1]}, \Delta\big)$

Public information and inputs

Public information: Matrix dimension n of the secret permutation matrix, $d = \lceil \log n \rceil$,

Verifier's input: Masked secret $\boldsymbol{t} \in (\mathbb{F}_2^d)^n$, $q_{u_{i,k}} = f_{u_{i,k}}(\Delta)$ for $(i,k) \in [0, n-1] \times [0, d-1]$, Δ.

Output

Verifier's output: VOLE correlations $q_{\boldsymbol{z}}$ and q_{ColCheck}.

```
    Compute VOLE correlations for $\boldsymbol{P}$'s coefficients
1 : for $i \in [0, n-1]$
2 :     $q_{\boldsymbol{z}_i} \leftarrow$ V.VOLE-ElementaryVector$(\boldsymbol{t}_i, (q_{u_{i,k}})_{k \in [0,d-1]}, \Delta)$
3 : endfor

    Compute VOLE correlations for $\boldsymbol{P}$'s columns check
4 : for $j \in [0, n-1]$
5 :     $q_{\mathsf{ColSum}_j} = \sum_{i=0}^{n-1} q_{i,j}$
6 :     $q_{\mathsf{ColCheck}_j} = q_{\mathsf{ColSum}_j} - \Delta^d$
7 : endif

8 : $q_{\boldsymbol{z}} = (q_{\boldsymbol{z}_0}, \dots, q_{\boldsymbol{z}_{n-1}})$
9 : $q_{\mathsf{ColCheck}} = (q_{\mathsf{ColCheck}_0}, \dots, q_{\mathsf{ColCheck}_{n-1}})$
10 : return $(q_{\boldsymbol{z}}, q_{\mathsf{ColCheck}})$
```

3.4 Resulting PoK from our modelings

Our modelings can be embedded into any proof systems (for instance MPCitH, TCitH or VOLEitH) in order to produce their corresponding PoK. Given that our modelings have multiplicative depth logarithmic with respect to their input size, the VOLEitH framework is well suited for them. One can build VOLEitH based PoK from our modelings by instantiating Algorithm 2.3. This is straightforward for our modelings for elementary vectors (using Algorithms P.VOLE-ElementaryVector and V.VOLE-ElementaryVector) and vectors of Hamming weight at most ω (using Algorithms P.VOLE-HammingWeight and V.VOLE-HammingWeight). The case of permutation is slightly more involved as Algorithms P.Check-Permutation and V.Check-Permutation have to be combined with the CheckZero protocol (see Appendix B for a detailed description of the PoK). These proposed PoK are knowledge sound and zero-knowledge in $\mathcal{F}_{\mathsf{VOLE}}$-hybrid model. We omit the proofs for showing these properties since they are identical to the proofs of protocols in [YSWW21, Roy22, BBD+23b, BBd+23a].

4 Application to SD and PKP based signatures

In this section, we describe signatures for the SD and PKP problems. In order to do so, we instantiate the VOLEitH framework presented in Section 2 with proofs built upon the modelings detailed in Sections 3.2 and 3.3 respectively. Our proofs use VOLE correlations with coefficients $u \in \mathbb{F}_2$ and $v \in \mathbb{F}_{2^\kappa}$ and τ repetitions are considered in order to obtain correlations over $\mathbb{F}_{2^{\kappa\tau}}$. Using as input a witness of size $|w|$, our proofs produce degree-d VOLE correlations $[\![a]\!]$ starting from a GGM tree with N leaves. The padding parameter B affects the security of the VOLE consistency check and is fixed to $B = 16$ similarly to what is done in [BBd+23a]. In addition, parameters (w', T_{open}) refers to the generic optimizations presented in [BBM+24]. In particular, this allows to reduce the cost associated to the GGM tree from $\tau \cdot (\lambda \log(N) + 2\lambda)$ to $\lambda \cdot T_{\mathsf{open}} + \tau \cdot 2\lambda$. This optimization is independent of the problem considered and can be used by all of the schemes. We defer the interested reader to [BBd+23a, BBM+24] for additional details. All things considered, our signatures have size:

$$\textsc{Size} = 4\lambda + \underbrace{(\tau - 1) \cdot \Big[|w| + (d-1) \cdot \kappa\tau + (\kappa\tau + B)\Big]}_{\text{VOLE correction strings}} + \underbrace{(\kappa\tau + B)}_{\text{VOLE consistency check}}$$
$$+ \underbrace{|w|}_{\text{Masked witness } \boldsymbol{t}} + \underbrace{d \cdot \kappa\tau}_{\text{Proof correlation } [\![a]\!]} + \underbrace{\lambda \cdot T_{\mathsf{open}} + \tau \cdot 2\lambda}_{\text{GGM tree}}.$$

4.1 Signature based on the SD problem

Since the seminal work of Stern [Ste94, Ste96], signatures built from PoK for the SD problem have been studied extensively [Vér97, GG07, AGS11, CVE11, BBBG21, BGKS22, GPS21, BGKM23, FJR22, BG23, FJR23, AFG+23, AGH+23, CCJ23, FR23b, CLY+24, BCC+24, OTX24]. Hereafter, we focus our comparison on signatures based on the SD over $\mathbb{F}_2$ which we consider to be the most conservative setting. For the sake of conciseness, we only consider the shortest signature from the literature which, to date, is obtained using the modeling from [FJR22, FR23b]. We also report numbers for SDitH [AFG+23] as this scheme is a candidate in the ongoing NIST's PQC standardization of additional signature schemes project. We propose a new signature scheme based on the SD problem by instantiating the VOLEitH framework with our modeling for vectors of Hamming weight at most ω using Algorithms P.Check-SD and V.Check-SD.

On the versatility of our modeling. While we are mainly considering the modeling from Section 3.2 in the context of SD over $\mathbb{F}_2$, one should note that the modeling is quite versatile and can also be used to design signatures based on SD over $\mathbb{F}_q$ or the d-split SD which generalizes the Regular SD problem. Regarding SD over $\mathbb{F}_q$, the proof described in Algorithms P.Check-SD and V.Check-SD can be seen as computing VOLE correlations $[\![\boldsymbol{z}_1]\!]$ associated to the support of the secret vector (output by P.VOLE-HammingWeight and V.VOLE-HammingWeight in line 1). We can also embed the actual values in $\mathbb{F}_q$ associated to each non-zero coordinates in VOLE correlations $[\![\boldsymbol{z}_2]\!]$, then one only needs to modify P.Check-SD to include the multiplication of $[\![\boldsymbol{z}_1]\!]$ and $[\![\boldsymbol{z}_2]\!]$ in order to extend it to vectors over $\mathbb{F}_q$. The d-split SD and Regular SD are variants of SD in which the secret vector is structured into d blocks of similar Hamming weight. The Regular SD (see Definition 2.3) constitutes the most structured case with ω blocks of weight 1. Our modeling can be adapted for this setting by running it d times on vector of size n/d which in the case of Regular SD reduce the witness size from $\omega \cdot \log_2(n)$ to $\omega \cdot \log_2(n/w)$ and the degree of the VOLE correlations from $\log_2(n)$ to $\log_2(n/\omega)$. A full description of this variant is given in Appendix C.

On the parameter optimization for our modeling. We now highlight an observation allowing us to reduce the input size of our proof by exploiting the fact that the witness size of our modeling scales logarithmictly with n. Contrarily to existing modelings that are optimal when considering codes of rate $1/2$, our case benefits from increasing the length n of the code and minimize the error weight ω as our modeling use a witness of size $\omega \cdot \log_2(n)$. As a result, we consider codes of low error rate ω/n. Before continuing with our observation, we have to mention that the Regular SD problem can be reduced to the SD problem easily, see for instance the proof from [FJR22]. Given an SD instance $(\boldsymbol{H}, \boldsymbol{y})$, one can apply a random permutation on the columns of $\boldsymbol{H}$ and submit the resulting instance to a Regular SD oracle. The probability to transform an SD instance into a Regular SD one is equal to the probability that a random word in $\mathcal{S}^n_\omega(\mathbb{F}_2)$ is regular namely $\binom{n/\omega}{1}^\omega / \binom{n}{\omega}$. In most cases, this reduction is not tight enough to be useful as the proportion of structured words is too small with respect to the number of words of Hamming weight ω. Interestingly, as we are considering codes of low error rate ω/n, we are using a large blocksize n/w which improves the tightness of the reduction and make it exploitable efficiently. As a consequence, we can use the Regular SD to benefit from a proof with input size $\omega \cdot \log_2(n/w)$ and degree $\log_2(n/\omega)$ while choosing parameters large enough (compensating the security loss due to the tightness) to benefit from the reduction to the SD problem. While this approach might seem counter-intuitive given that the reduction is not tight, it turns out to be more efficient than using the SD problem directly as reported in Table 3. This is explained by the fact that reducing the multiplicative depth of the modeling compensate for the larger witness size induced by the choice of bigger parameters.

Signature based on SD over $\mathbb{F}_2$. We provide parameters for signatures based on SD and corresponding witness sizes in Tables 2 and 3 respectively. For our work, we provide two parameter sets for NIST-I security level namely one based directly on SD ($\mathsf{Param\ 1}$) and one based on SD using the Regular SD structure along with its reduction to SD ($\mathsf{Param\ 2}$). We stress that the security of both approaches relies on the SD over $\mathbb{F}_2$ problem. In addition, we provide the resulting signature sizes for NIST-I and NIST-V security levels in Table 4. In order to provide a fair comparison, we have added the optimization from [BBM+24] to the [FR23b] scheme. Numbers for the SDitH scheme are given without any modification or optimization as they are intended to serve as a reference.

Signature based on Regular SD over $\mathbb{F}_2$. Regarding Regular SD, our modeling leads to a signature of size 3.2 kB using parameters $(n = 1120, k = 700, \omega = 140, \kappa = 11, \tau = 11, w' = 6, T_{\mathsf{open}} = 100)$ for NIST-I security level. The schemes from [CLY+24] and [OTX24] feature similar signature sizes although one should note that these schemes differs on the optimizations used, the performance/size trade-offs used in their choice of MPC parameters or their targeted security level hence are not fully commensurable. Indeed, the signature from [CLY+24] features a size of 4.0 kB for NIST-I security level and the signature from [OTX24] is 3.0 kB long for a security level of $\lambda = 128$.

Algorithm 4.1:
P.Check-SD$\big(\boldsymbol{x}, \mathsf{pk}, [\![u_{i,k}]\!]_{i\in[0,\omega-1],k\in[0,d-1]}, ([\![s_{k'}]\!])_{k'\in[0,d-2]}, \mathsf{seed}\big)$

Public information and inputs

Public information: Weight of the vector ω, length of the vector n, $d = \lceil \log n \rceil$.
Prover's input: Secret vector $\boldsymbol{x}$ represented as ω positions $(\mathsf{pos}_0, \dots, \mathsf{pos}_{\omega-1})$, public key $\mathsf{pk} = (\boldsymbol{H}, \boldsymbol{y}) \in \mathbb{F}_2^{m\times n} \times \mathbb{F}_2^m$, $\omega \cdot d$ VOLE correlations $[\![u_{i,k}]\!]$ for random $(u_{i,k}, v_{i,k}) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$ represented as polynomials $f_{u_{i,k}}(X) = u_{i,k}X + v_{i,k}$ for $(i,k) \in [0,\omega-1] \times [0,d-1]$, $(d-1)$ VOLE correlations $[\![s_{k'}]\!]$ for random $r_{k'}, s_{k'} \in \mathbb{F}_{2^\lambda}$ represented as $f_{s_{k'}}(X) = s_{k'}X + r_{k'}$ for $k' \in [0, d-2]$, $\mathsf{seed} \in \mathbb{F}_{2^\lambda}$.

Output

Prover's output: Proof $([\![a]\!], \boldsymbol{t}, \mathsf{seed})$ that $\boldsymbol{x}$ is a solution to the SD instance defined by pk.

```
     Compute $\boldsymbol{x}$ from its support
1 :  $(\boldsymbol{t}, [\![\boldsymbol{x}]\!]^{(d)}) \leftarrow$ P.VOLE-HammingWeight$\big(\boldsymbol{x}, ([\![u_{i,k}]\!])_{i\in[0,\omega-1],k\in[0,d-1]}\big)$

     Compute $\boldsymbol{Hx} - \boldsymbol{y}$
2 :  for $i \in [0, m-1]$
3 :    $f_i(X) = \sum_{j=0}^{n-1} h_{i,j} \cdot [\![\boldsymbol{x}_j]\!]^{(d)} - \boldsymbol{y}_i \cdot X^d$
4 :  endfor

     Merge polynomials and run CheckZero
5 :  $\boldsymbol{\alpha} \xleftarrow{\$,\mathsf{seed}} \in \mathbb{F}_{2^\lambda}^m$
6 :  $f(X) = \sum_{i=0}^{m-1} \alpha_i \cdot f_i(X)$
7 :  $[\![a]\!] \leftarrow$ P.CheckZero$\big(f(X), ([\![s_{k'}]\!])_{k'\in[0,d-2]}\big)$

8 :  proof $= ([\![a]\!], \boldsymbol{t}, \mathsf{seed})$
9 :  return proof
```

Algorithm 4.2:
V.Check-SD$\big(\mathsf{proof}, \mathsf{pk}, (q_{u_{i,k}})_{i\in[0,\omega-1],k\in[0,d-1]}, (q_{s_{k'}})_{k'\in[0,d-2]}, \Delta\big)$

Public information and inputs

Public information: Weight of the vector ω, length of the vector n, $d = \lceil \log n \rceil$.
Verifier's input: proof $= ([\![a]\!], \boldsymbol{t}, \mathsf{seed})$, public key $\mathsf{pk} = (\boldsymbol{H}, \boldsymbol{y}) \in \mathbb{F}_2^{m\times n} \times \mathbb{F}_2^m$, $q_{u_{i,k}} = f_{u_{i,k}}(\Delta)$ for $(i,k) \in [0,\omega-1] \times [0,d-1]$, $q_{s_{k'}} = f_{s_{k'}}(\Delta)$ for $k' \in [0,d-2]$, Δ.

Output

```
Verifier's output: Boolean indicating if proof is a valid proof of a SD solution.
- - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - - -
     Compute VOLE correlations for $\boldsymbol{x}$
1 :  $q_{\boldsymbol{x}} \leftarrow$ V.VOLE-HammingWeight $(\boldsymbol{t}, (q_{u_{i,k}})_{i \in [0,\omega-1], k \in [0,d-1]}, \Delta)$

     Compute VOLE correlations for $\boldsymbol{Hx} - \boldsymbol{y}$
2 :  for $i \in [0, m-1]$
3 :    $q_i = \sum_{j=0}^{n-1} h_{i,j} \cdot q_{\boldsymbol{x}_j} - \boldsymbol{y}_i \cdot \Delta^d$
4 :  endfor

     Merge polynomials and run CheckZero
5 :  $\boldsymbol{\alpha} \xleftarrow{\$,\mathsf{seed}} \in \mathbb{F}_{2^\lambda}^m$
6 :  $q = \sum_{i=0}^{m-1} \alpha_i \cdot q_i$
7 :  $b \leftarrow$ V.CheckZero $([\![a]\!], q, (q_{s_{k'}})_{k' \in [0,d-2]}, \Delta)$

8 :  return $b$.
```

Scheme	**Security**	n	k	ω	N	κ	τ	w'	T_{open}
[FJR22, FR23b]	NIST-I	1280	640	132	2048	11	11	5	100
	NIST-V	2400	1200	266	2048	11	23	0	214
This work (Param 1)	NIST-I	4096	3737	46	2048	11	11	6	100
	NIST-V	4096	3280	128	2048	11	23	2	214
This work (Param 2)	NIST-I	6080	5379	95	2048	11	11	6	100
	NIST-V	12160	10755	190	2048	11	23	2	214

Table 2: Parameter sets for SD based signatures

4.2 Signature based on PKP

Since the seminal work of Shamir [Sha90], several signatures based on the PKP problem have been proposed namely PKP-DSS [BFK+19], SUSHYFISH [Beu20] or the unnamed ones from [BG23] and [Fen24]. To date, PERK [BBD+24, ABB+23a] features the smallest signature sizes amongst these schemes although it relies on a slightly weaker assumption denoted r-IPKP for relaxed inhomegeneous PKP. Hereafter, we design a new signature scheme based on PKP by instantiating the VOLEitH framework along with our modeling from Section 3.3 using Algorithms P.Check-PKP and V.Check-PKP.

Modeling	Witness		Mul. depth	
	Formula	**Size**	**Formula**	**Value**
[FJR22, FR23b]	$k + w \cdot \log_2(\mathbb{F}_{2^\kappa})$	**262 B**	-	**2**
This Work (Param 1)	$w \cdot \log_2(n)$	**69 B**	$\log_2(n)$	**12**
This Work (Param 2)	$w \cdot \log_2(n/w)$	**72 B**	$\log_2(n/w)$	**6**

Table 3: Modelings for SD over $\mathbb{F}_2$ and witness sizes (for NIST-I security level)

Scheme	**Assumption**	**Security**	sk	pk	**Signature**
SDitH	SD over $\mathbb{F}_q$	NIST-I	0.4 kB	0.1 kB	**8.5 kB**
		NIST-V	0.8 kB	0.2 kB	**34.0 kB**
[FJR22, FR23b]	SD over $\mathbb{F}_2$	NIST-I	16 B	0.1 kB	**5.3 kB**
		NIST-V	32 B	0.2 kB	**21.9 kB**
This work (Param 1)	SD over $\mathbb{F}_2$	NIST-I	16 B	0.1 kB	**4.9 kB**
		NIST-V	32 B	0.2 kB	**21.7 kB**
This work (Param 2)	SD over $\mathbb{F}_2$	NIST-I	16 B	0.1 kB	**3.9 kB**
		NIST-V	32 B	0.2 kB	**16.2 kB**

Table 4: Comparison of SD based signatures

For the sake of conciseness, we only compare our new signature to PERK as it is the smallest PKP based signature. Our scheme brings three improvements with respect to PERK as (i) it relies on the PKP assumption (with $q = 2^k$) rather than the weaker r-IPKP one, (ii) it is compatible with the recent VOLEitH framework which is not the case of PERK and (iii) it features a smaller witness size. Parameters for our schemes are given in Table 5 while parameters for PERK are from [ABB+23a]. Comparison of modelings and resulting signatures are given in Tables 6 and 7.

Algorithm 4.3:
P.Check-PKP$\left(\boldsymbol{P}, \mathsf{pk}, [\![u_{i,k}]\!]_{i\in[0,n-1],k\in[0,d-1]}, ([\![s_{k'}]\!])_{k'\in[0,d-2]}, \mathsf{seed}\right)$

Public information

Public information: Matrix dimension n of the secret permutation matrix, $d = \lceil \log n \rceil$.
Prover's input: Secret permutation matrix $\boldsymbol{P}$ represented as n positions $(\mathsf{pos}_0, \ldots, \mathsf{pos}_{n-1})$, public key $\mathsf{pk} = (\boldsymbol{H}, \boldsymbol{x})$, $n \cdot d$ VOLE correlations $[\![u_{i,k}]\!]$ for random $(u_{i,k}, v_{i,k}) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$

represented as polynomials $f_{u_{i,k}}(X) = u_{i,k}X + v_{i,k}$ for $(i,k) \in [0, n-1] \times [0, d-1]$, $(d-1)$ VOLE correlations $[\![s_{k'}]\!]$ for random $r_{k'}, s_{k'} \in \mathbb{F}_{2^\lambda}$ represented as $f_{s_{k'}}(X) = s_{k'}X + r_{k'}$ for $k' \in [0, d-2]$, $\mathsf{seed} \in \mathbb{F}_{2^\lambda}$.

Output

Prover's output: Proof $([\![a]\!], \boldsymbol{t}, \mathsf{seed})$ that $\boldsymbol{P}$ is a solution to the PKP instance defined by pk.

```
          Compute $\boldsymbol{P}$ in matrix form
  1 :   $(\boldsymbol{t}, [\![\boldsymbol{z}]\!]^{(d)}, [\![\mathsf{ColCheck}]\!]^{(d)}) \leftarrow \text{P.VOLE-Permutation}\big(\boldsymbol{P}, ([\![u_{i,k}]\!])_{i\in[0,n-1],k\in[0,d-1]}\big)$
  2 :   Parse $[\![\mathsf{ColCheck}]\!]^{(d)}$ as $(f_0(X), f_1(X), \ldots, f_{n-1}(X))$

          Compute $\boldsymbol{x}' = \boldsymbol{P}\boldsymbol{x}$
  3 :   for $i \in [0, n-1]$
  4 :      $[\![\boldsymbol{x}'_i]\!]^{(d)} = \sum_{j=0}^{n-1} [\![z_{i,j}]\!]^{(d)} \cdot \boldsymbol{x}_j$
  5 :   endfor

          Compute $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x}'$
  6 :   for $i \in [0, m-1]$
  7 :      $[\![\boldsymbol{y}_i]\!]^{(d)} = f_{i+n}(X) = \sum_{j=0}^{n-1} h_{i,j} \cdot [\![\boldsymbol{x}'_j]\!]^{(d)}$
  8 :   endfor

          Merge polynomials and run CheckZero
  9 :   $\boldsymbol{\alpha} \xleftarrow{\$,\mathsf{seed}} \in \mathbb{F}_{2^\lambda}^{n+m}$
 10 :   $f(X) = \sum_{j=0}^{n+m-1} \alpha_j \cdot f_j(X)$
 11 :   $[\![a]\!] \leftarrow \text{P.CheckZero}\big(f(X), ([\![s_{k'}]\!])_{k'\in[0,d-2]}\big)$

 12 :   $\mathsf{proof} = ([\![a]\!], \boldsymbol{t}, \mathsf{seed})$
 13 :   return proof
```

Algorithm 4.4:
V.Check-PKP$\big(\mathsf{proof}, \mathsf{pk}, (q_{u_{i,k}})_{i\in[0,n-1],k\in[0,d-1]}, (q_{s_{k'}})_{k'\in[0,d-2]}, \Delta\big)$

Public information and inputs

Public information: Matrix dimension n of the secret permutation matrix, $d = \lceil \log n \rceil$.

Verifier's input: $\mathsf{proof} = ([\![a]\!], \boldsymbol{t}, \mathsf{seed})$, public key $\mathsf{pk} = (\boldsymbol{H}, \boldsymbol{x})$, $q_{u_{i,k}} = f_{u_{i,k}}(\Delta)$ for $(i,k) \in [0, n-1] \times [0, d-1]$, $q_{s_{k'}} = f_{s_{k'}}(\Delta)$ for $k' \in [0, d-2]$, Δ.

Output

Verifier's output: Boolean indicating if proof is a valid proof of a PKP solution.

```
     Compute VOLE correlations for $\boldsymbol{P}$
1 :  $(q_{\boldsymbol{z}}, q_{\mathsf{ColCheck}}) \leftarrow$ V.VOLE-Permutation$(\boldsymbol{t}, (q_{u_{i,k}})_{i\in[0,n-1],k\in[0,d-1]}, \Delta)$

     Compute VOLE correlations for $\boldsymbol{x}' = \boldsymbol{P}\boldsymbol{x}$
2 :  for $i \in [0, n-1]$
3 :    $q_{\boldsymbol{x}'_i} = \sum_{j=0}^{n-1} q_{i,j} \cdot x_j$
4 :  endfor

     Compute VOLE correlations for $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x}'$
5 :  for $i \in [0, m-1]$
6 :    $q_{\boldsymbol{y}_i} = \sum_{j=0}^{n-1} h_{i,j} \cdot q_{\boldsymbol{x}'_j}$
7 :  endfor

     Merge polynomials and run CheckZero
8 :  $\boldsymbol{\alpha} \xleftarrow{\$,\mathsf{seed}} \in \mathbb{F}_{2^\lambda}^{n+m}$
9 :  $q = \sum_{j=0}^{n-1} \alpha_j \cdot q_{\mathsf{ColCheck}_j} + \sum_{j=n}^{n+m-1} \alpha_j \cdot q_{\boldsymbol{y}_{j-n}}$
10 : $b \leftarrow$ V.CheckZero $([\![a]\!], q, (q_{s_{k'}})_{k'\in[0,d-2]}, \Delta)$

11 : return $b$.
```

Security	q	n	m	N	κ	τ	w'	T_{open}
NIST-I	2048	64	27	2048	11	11	6	100
NIST-V	4096	109	49	2048	11	23	2	214

Table 5: Parameter sets for our PKP based signature

Modeling	**Witness**	**Size**
PERK [ABB+23a]	$n\log_2(n) + n\log_2(q)$	**136 B**
This work	$n\log_2(n)$	**48 B**

Table 6: Modelings for PKP and resulting witness sizes (for NIST-I security level)

Scheme	Assumption	Security	sk	pk	Signature
PERK [ABB+23a]	r-IPKP	NIST-I	16 B	0.2 kB	**5.8 kB**
		NIST-V	32 B	0.5 kB	**23.0 kB**
This work	PKP	NIST-I	16 B	0.1 kB	**3.6 kB**
		NIST-V	32 B	0.2 kB	**15.9 kB**

Table 7: Comparison of PKP based signatures

5 Application to PoK of secret keys of KEMs

In the shift toward post-quantum cryptography, [SSW20] introduced the KEMTLS protocol as an alternative to the TLS 1.3 handshake, utilizing key encapsulation mechanisms (KEMs) for authentication in place of digital signatures. However, KEMTLS requires certificates to contain KEM keys, presenting the challenge of proving possessing a KEM key. In [GHL+22], a non-interactive proof of possession of a KEM certificate was proposed for lattice-based KEMs. For the code-based KEMs BIKE and HQC, currently under consideration in the fourth round of NIST's Post-Quantum Standardization project, our approach provides compact short proofs, achieving a size of less than 10 kilobytes for NIST-I security level. Interestingly, a comparable proof for Kyber is 17.8 kB long [GHL+22] which suggests that advanced protocols using PoK of secret keys of KEM may have a smaller footprint when instantiated with code-based KEMs instead of lattice-based KEMs.

Indeed, similarly to what was detailed in Section 4, our new modelling can be used to prove the knowledge of the secret keys associated to BIKE and HQC. Doing so requires to prove the knowledge of a solution of a 2-QCSD over $\mathbb{F}_2$ instance (where QCSD stands for Quasi-Cyclic Syndrome Decoding). This can be achieved using our modelling for the SD twice as the solution is composed of two blocks of similar weight. Interestingly, this setting highlights one of the advantages of our modelling namely that it scales logarithmically with the length of the considered code while existing modelings feature a polynomial scaling. As both BIKE and HQC uses quasi-cyclic codes whose lengths are more than 10 times greater than the lengths typically considered when designing signatures, our modelling leads to significant improvement with respect to existing ones in this setting. This is illustrated in Table 8 using HQC parameters $(n = 35338, k = 17669, \omega = 132)$ and $(n = 115274, k = 57637, \omega = 262)$ for NIST-I and NIST-V security levels respectively along with MPC parameters $(\kappa = 11, \tau = 12)$ and $(\kappa = 11, \tau = 24)$ for NIST-I and NIST-V security levels.

Modeling	Code Length	Security	Witness Size	PoK Size
[FJR22, FR23b]	4 418 B	NIST-I	2.4 kB	**31.7 kB**
	14 410 B	NIST-V	7.6 kB	**193 kB**
This work	4 418 B	NIST-I	248 B	**8.6 kB**
	14 410 B	NIST-V	524 B	**35.5 kB**

Table 8: Comparison of PoK of HQC's secret key

6 Application to Ring Signatures

An interesting application of our modelings given in Section 3 is ring signatures. Informally, a ring signature allows a user to anonymously sign a message on behalf of a group, making it impossible to determine who in the group signed the message. Let some user Alice be a part of a group of $\bar{n}$ users (ring $\mathcal{R}$ for the ring signature), with each user user_i holding its corresponding public and private keys $(\mathsf{pk}_i, \mathsf{sk}_i)$. Let Alice's index be $i^* \in [0, \bar{n}-1]$ and let $\bar{d} = \lceil \log(\bar{n}) \rceil$. Note that Alice should be able to sign a message using sk_{i^*} which can be verified by anyone holding the set of public keys $\mathsf{pk}_{\mathcal{R}} := (\mathsf{pk}_0, \mathsf{pk}_1, \ldots, \mathsf{pk}_{i^*}, \ldots, \mathsf{pk}_{\bar{n}-1})$ without revealing i^*. Following the technique introduced in [BS13] and used in [FR23b], this can be achieved as follows:

1. Alice shares her secret key as $[\![\mathsf{sk}_{i^*}]\!]$;
2. Alice shares the selection vector $\boldsymbol{e}_{i^*} \in \{0,1\}^{\bar{n}}$ (i.e. the vector of size $\bar{n}$ which has all its coordinates equal to zero except for the i^*th one set to 1) as $[\![\boldsymbol{e}_{i^*}]\!]$;
3. Alice uses $[\![\boldsymbol{e}_{i^*}]\!]$ to select $[\![\mathsf{pk}_{i^*}]\!]$ from $\mathsf{pk}_{\mathcal{R}}$ without revealing i^* by running some interactive protocol with the verifier ;
4. The verifier uses $[\![\mathsf{pk}_{i^*}]\!]$ to check the signature generated using $[\![\mathsf{sk}_{i^*}]\!]$.

Note that, the selection vector $\boldsymbol{e}_{i^*} \in \{0,1\}^{\bar{n}}$ can be generated from VOLE correlations with $\mathsf{P.VOLE-ElementaryVector}$ using the secret position i^* as input. Therefore, one can construct VOLEitH based ring signatures as follows:

1. Generate random VOLE correlations then send the prover's inputs to the signer and the verifier's inputs to the verifier of the ring signature scheme ;
2. Compute $[\![\boldsymbol{e}_{i^*}]\!]$ using $\mathsf{P.VOLE-ElementaryVector}$;
3. Compute the verification key as $[\![\mathsf{vk}_{\mathcal{R}}]\!] = \sum_{i=0}^{\bar{n}-1} [\![\boldsymbol{e}_{i^*}]\!] \cdot \mathsf{pk}_i$;
4. Compute the signature using the secret key $[\![\mathsf{sk}_{i^*}]\!]$ corresponding to $[\![\mathsf{vk}_{\mathcal{R}}]\!]$.

Doing so, we can convert the signatures based on SD and PKP from Section 4 into ring signatures. The main modification consists to update the polynomial constraints to take into account the fact that the public key is now a VOLE correlation rather than a public value. Hereafter, we briefly explain how to do it and refer the interested reader to the full description of the ring signatures provided in Appendices D.1 and D.2 respectively.

Ring signature based on SD. Each user user_i in the ring possesses a key pair defined as $\mathsf{pk}_i = (\boldsymbol{H}_i, \boldsymbol{y}_i)$ and $\mathsf{sk}_i = \boldsymbol{x}_i$. The polynomial constraints of the proof are defined as follows:

$$f_i(X) = \sum_{j=0}^{n-1} [\![h_{i,j}]\!]^{(\bar{d})} \cdot [\![\boldsymbol{x}_j]\!]^{(d)} - [\![\boldsymbol{y}_i]\!]^{(\bar{d})} \cdot X^d \quad \forall i \in [0, m-1]$$

where $[\![\boldsymbol{x}_j]\!]$ is computed as in P.Check-SD using the secret key $\boldsymbol{x}_{i^*}$ and ($[\![h_{i,j}]\!]$, $[\![\boldsymbol{y}_i]\!]$) are computed using $\mathsf{pk}_{\mathcal{R}}$ and P.VOLE-ElementaryVector with input i^* as described previously.

Ring signature based on PKP. Each user user_i in the ring possesses a key pair defined as $\mathsf{pk}_i = (\boldsymbol{H}_i, \boldsymbol{x}_i)$ and $\mathsf{sk}_i = \boldsymbol{P}_i$ The polynomial constraints of the proof are defined as follows:

$$[\![\boldsymbol{y}_i]\!]^{(d+2\cdot\bar{d})} = f_{i+n}(X) = \sum_{j=0}^{n-1} [\![h_{i,j}]\!]^{(\bar{d})} \cdot [\![\boldsymbol{x}'_j]\!]^{(d+\bar{d})} \quad \forall i \in [0, m-1].$$

where ($[\![h_{i,j}]\!]$, $[\![\boldsymbol{x}]\!]$) are computed using $\mathsf{pk}_{\mathcal{R}}$ and P.VOLE-ElementaryVector with i^* as input and $[\![\boldsymbol{x}'_j]\!]$ is computed as in P.Check-PKP using $\boldsymbol{P}_{i^*}$ and $[\![\boldsymbol{x}]\!]$.

Several post-quantum ring signatures have been proposed over the years, see for instance [KKW18, GGHAK22, LAZ19, EZS+19, BKP20, BBN+22, BESV22, LN22, FR23b]. To date, the shortest ring signatures are obtained using the MQ problem using the work from [FR23b]. As described in Table 9, our modelings lead to the shortest ring signature based on the SD problem and the first ring signature based on the PKP problem.

Scheme	2^3	2^6	2^8	2^{10}	2^{12}	2^{20}	Assumption
[FR23b]	4.30	4.33	4.37	4.45	4.60	5.62	MQ
	7.37	7.51	7.96	8.24	8.40	10.09	SD
This work	4.34	4.85	5.18	5.52	5.85	7.20	
	4.62	5.62	6.29	6.96	7.63	10.30	PKP

Table 9: Comparison of ring signature sizes (in kB) for NIST-I security level

Acknowledgement. The authors would like to thank Thibauld Feneuil for fruitful discussions concerning the impact of the multiplicative depth of our modelings as well as Andre Esser for fruitful discussions regarding the PKP assumption.

References

AAB$^+$22. Carlos Aguilar-Melchor, Nicolas Aragon, Slim Bettaieb, Loïc Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Philippe Gaborit, Edoardo Persichetti, Gilles Zémor, Jurjen Bos, Arnaud Dion, Jerome Lacan, Jean-Marc Robert, and Pascal Veron. HQC. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

ABB$^+$22. Nicolas Aragon, Paulo Barreto, Slim Bettaieb, Loic Bidoux, Olivier Blazy, Jean-Christophe Deneuville, Phillipe Gaborit, Shay Gueron, Tim Guneysu, Carlos Aguilar-Melchor, Rafael Misoczki, Edoardo Persichetti, Nicolas Sendrier, Jean-Pierre Tillich, Gilles Zémor, Valentin Vasseur, Santosh Ghosh, and Jan Richter-Brokmann. BIKE. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/round-4-submissions.

ABB$^+$23a. Najwa Aaraj, Slim Bettaieb, Loïc Bidoux, Alessandro Budroni, Victor Dyseryn, Andre Esser, Philippe Gaborit, Mukul Kulkarni, Victor Mateu, Marco Palumbi, Lucas Perin, and Jean-Pierre Tillich. PERK. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

ABB$^+$23b. Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Matthieu Rivain, Jean-Pierre Tillich, and Adrien Vinçotte. RYDE. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

ABB$^+$23c. Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, Matthieu Rivain, and Jean-Pierre Tillich. MIRA. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

AFG$^+$23. Carlos Aguilar-Melchor, Thibauld Feneuil, Nicolas Gama, Shay Gueron, James Howe, David Joseph, Antoine Joux, Edoardo Persichetti, Tovohery H. Randrianarisoa, Matthieu Rivain, and Dongze Yue. SDitH — Syndrome Decoding in the Head. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

AGH$^+$23. Carlos Aguilar-Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. The return of the SDitH. In Hazay and Stam [HS23], pages 564–596.

AGS11. Carlos Aguilar, Philippe Gaborit, and Julien Schrek. A new zero-knowledge code based identification scheme with reduced communication. In *IEEE Information Theory Workshop*, 2011.

ARV$^+$23. Gora Adj, Luis Rivera-Zamarripa, Javier Verbel, Emanuele Bellini, Stefano Barbero, Andre Esser, Carlo Sanna, and Floyd Zweydinger. MiRitH — MinRank in the Head. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

BBBG21. Slim Bettaieb, Loïc Bidoux, Olivier Blazy, and Philippe Gaborit. Zero-Knowledge Reparation of the Véron and AGS Code-based Identification Schemes. In *IEEE International Symposium on Information Theory (ISIT)*, 2021.

BBd$^+$23a. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. FAEST. Technical report, National Institute of Standards and Technology, 2023. available at `https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures`.

BBD$^+$23b. Carsten Baum, Lennart Braun, Cyprien Delpech de Saint Guilhem, Michael Klooß, Emmanuela Orsini, Lawrence Roy, and Peter Scholl. Publicly verifiable zero-knowledge and post-quantum signatures from VOLE-in-the-head. In Helena Handschuh and Anna Lysyanskaya, editors, *CRYPTO 2023, Part V*, volume 14085 of *LNCS*, pages 581–615. Springer, Cham, August 2023.

BBD$^+$24. Slim Bettaieb, Loïc Bidoux, Victor Dyseryn, Andre Esser, Philippe Gaborit, Mukul Kulkarni, and Marco Palumbi. PERK: compact signature scheme based on a new variant of the permuted kernel problem. *Designs, Codes and Cryptography*, pages 2131–2157, 2024.

BBM$^+$24. 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. Cryptology ePrint Archive, Report 2024/490, 2024.

BBN$^+$22. Alessandro Barenghi, Jean-Francois Biasse, Tran Ngo, Edoardo Persichetti, and Paolo Santini. Advanced signature functionalities from the code equivalence problem. Cryptology ePrint Archive, Report 2022/710, 2022.

BCC$^+$24. Dung Bui, Eliana Carozza, Geoffroy Couteau, Dahmun Goudarzi, and Antoine Joux. Short signatures from regular syndrome decoding, revisited. Cryptology ePrint Archive, Report 2024/252, 2024.

BCGI18. Elette Boyle, Geoffroy Couteau, Niv Gilboa, and Yuval Ishai. Compressing vector OLE. In Lie et al. [LMBW18], pages 896–912.

Bea92. Donald Beaver. Efficient multiparty protocols using circuit randomization. In Joan Feigenbaum, editor, *CRYPTO'91*, volume 576 of *LNCS*, pages 420–432. Springer, Berlin, Heidelberg, August 1992.

BESV22. Emanuele Bellini, Andre Esser, Carlo Sanna, and Javier A. Verbel. MR-DSS - smaller MinRank-based (ring-)signatures. In Jung Hee Cheon and Thomas Johansson, editors, *Post-Quantum Cryptography - 13th International Workshop, PQCrypto 2022*, pages 144–169. Springer, Cham, September 2022.

Beu20. Ward Beullens. Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part III*, volume 12107 of *LNCS*, pages 183–211. Springer, Cham, May 2020.

BFG$^+$24. Loïc Bidoux, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, and Matthieu Rivain. Dual support decomposition in the head: Shorter signatures from rank SD and MinRank. Cryptology ePrint Archive, Report 2024/541, 2024.

BFK+19. Ward Beullens, Jean-Charles Faugère, Eliane Koussa, Gilles Macario-Rat, Jacques Patarin, and Ludovic Perret. PKP-based signature scheme. In Feng Hao, Sushmita Ruj, and Sourav Sen Gupta, editors, *INDOCRYPT 2019*, volume 11898 of *LNCS*, pages 3–22. Springer, Cham, December 2019.

BG23. Loïc Bidoux and Philippe Gaborit. Compact Post-quantum Signatures from Proofs of Knowledge Leveraging Structure for the PKP, SD and RSD Problems. In *Codes, Cryptology and Information Security (C2SI)*, pages 10–42. Springer, 2023.

BGKM23. Loïc Bidoux, Philippe Gaborit, Mukul Kulkarni, and Víctor Mateu. Code-based signatures from new proofs of knowledge for the syndrome decoding problem. *DCC*, 91(2):497–544, 2023.

BGKS22. Loïc Bidoux, Philippe Gaborit, Mukul Kulkarni, and Nicolas Sendrier. Quasi-Cyclic Stern Proof of Knowledge. In *IEEE International Symposium on Information Theory (ISIT)*, pages 1459–1464, 2022.

BKP20. Ward Beullens, Shuichi Katsumata, and Federico Pintore. Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part II*, volume 12492 of *LNCS*, pages 464–492. Springer, Cham, December 2020.

BKPV23. Luk Bettale, Delaram Kahrobaei, Ludovic Perret, and Javier Verbel. Biscuit. Technical report, National Institute of Standards and Technology, 2023. available at `https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures`.

BØ23. Pierre Briaud and Morten Øygarden. A new algebraic approach to the regular syndrome decoding problem and implications for PCG constructions. In Hazay and Stam [HS23], pages 391–422.

BS13. Slim Bettaieb and Julien Schrek. Improved lattice-based threshold ring signature scheme. In Philippe Gaborit, editor, *Post-Quantum Cryptography - 5th International Workshop, PQCrypto 2013*, pages 34–51. Springer, Berlin, Heidelberg, June 2013.

CCJ23. Eliana Carozza, Geoffroy Couteau, and Antoine Joux. Short signatures from regular syndrome decoding in the head. In Hazay and Stam [HS23], pages 532–563.

CLY+24. Hongrui Cui, Hanlin Liu, Di Yan, Kang Yang, Yu Yu, and Kaiyi Zhang. ReSolveD: Shorter signatures from regular syndrome decoding and VOLE-in-the-head. In Tang and Teague [TT24], pages 229–258.

Cv91. David Chaum and Eugène van Heyst. Group signatures. In Donald W. Davies, editor, *EUROCRYPT'91*, volume 547 of *LNCS*, pages 257–265. Springer, Berlin, Heidelberg, April 1991.

CVE11. Pierre-Louis Cayrel, Pascal Véron, and Sidi Mohamed El Yousfi Alaoui. A zero-knowledge identification scheme based on the q-ary syndrome decoding problem. In Alex Biryukov, Guang Gong, and Douglas R. Stinson, editors, *SAC 2010*, volume 6544 of *LNCS*, pages 171–186. Springer, Berlin, Heidelberg, August 2011.

ES24. Andre Esser and Paolo Santini. Not just regular decoding: Asymptotics and improvements of regular syndrome decoding attacks. In Leonid Reyzin and Douglas Stebila, editors, *CRYPTO 2024, Part VI*, volume 14925 of *LNCS*, pages 183–217. Springer, Cham, August 2024.

EZS+19. Muhammed F. Esgin, Raymond K. Zhao, Ron Steinfeld, Joseph K. Liu, and Dongxi Liu. MatRiCT: Efficient, scalable and post-quantum blockchain confidential transactions protocol. In Lorenzo Cavallaro, Johannes Kinder, XiaoFeng Wang, and Jonathan Katz, editors, *ACM CCS 2019*, pages 567–584. ACM Press, November 2019.

Fen24. Thibauld Feneuil. Building MPCitH-based signatures from MQ, MinRank, and rank SD. In Christina Pöpper and Lejla Batina, editors, *ACNS 24International Conference on Applied Cryptography and Network Security, Part I*, volume 14583 of *LNCS*, pages 403–431. Springer, Cham, March 2024.

FFS88. Uriel Feige, Amos Fiat, and Adi Shamir. Zero-knowledge proofs of identity. *Journal of Cryptology*, 1(2):77–94, June 1988.

FJR22. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In Yevgeniy Dodis and Thomas Shrimpton, editors, *CRYPTO 2022, Part II*, volume 13508 of *LNCS*, pages 541–572. Springer, Cham, August 2022.

FJR23. Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Shared permutation for syndrome decoding: new zero-knowledge protocol and code-based signature. *DCC*, 91(2):563–608, 2023.

FR23a. Thibauld Feneuil and Matthieu Rivain. MQOM — MQ on my Mind. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

FR23b. Thibauld Feneuil and Matthieu Rivain. Threshold computation in the head: Improved framework for post-quantum signatures and zero-knowledge arguments. Cryptology ePrint Archive, Report 2023/1573, 2023.

FS87. Amos Fiat and Adi Shamir. How to prove yourself: Practical solutions to identification and signature problems. In Andrew M. Odlyzko, editor, *CRYPTO'86*, volume 263 of *LNCS*, pages 186–194. Springer, Berlin, Heidelberg, August 1987.

GG07. Philippe Gaborit and Marc Girault. Lightweight code-based identification and signature. In *IEEE International Symposium on Information Theory (ISIT)*, 2007.

GGHAK22. Aarushi Goel, Matthew Green, Mathias Hall-Andersen, and Gabriel Kaptchuk. Efficient set membership proofs using MPC-in-the-head. *PoPETs*, 2022(2):304–324, April 2022.

GGM84. Oded Goldreich, Shafi Goldwasser, and Silvio Micali. How to construct random functions (extended abstract). In *25th FOCS*, pages 464–479. IEEE Computer Society Press, October 1984.

GHL$^+$22. Tim Güneysu, Philip W. Hodges, Georg Land, Mike Ounsworth, Douglas Stebila, and Greg Zaverucha. Proof-of-possession for KEM certificates using verifiable generation. In Heng Yin, Angelos Stavrou, Cas Cremers, and Elaine Shi, editors, *ACM CCS 2022*, pages 1337–1351. ACM Press, November 2022.

GMR85. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof-systems (extended abstract). In *17th ACM STOC*, pages 291–304. ACM Press, May 1985.

GMR89. Shafi Goldwasser, Silvio Micali, and Charles Rackoff. The knowledge complexity of interactive proof systems. *SIAM J. Comput.*, 18(1):186–208, 1989.

GPS21. Shay Gueron, Edoardo Persichetti, and Paolo Santini. Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup. Cryptology ePrint Archive, Report 2021/1020, 2021.

HS23. Carmit Hazay and Martijn Stam, editors. *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*. Springer, Cham, April 2023.

IKOS07. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge from secure multiparty computation. In David S. Johnson and Uriel Feige, editors, *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

IKOS09. Yuval Ishai, Eyal Kushilevitz, Rafail Ostrovsky, and Amit Sahai. Zero-knowledge proofs from secure multiparty computation. *SIAM Journal on Computing*, 39(3):1121–1152, 2009.

KCC$^+$23. Seongkwang Kim, Jihoon Cho, Mingyu Cho, Jincheol Ha, Jihoon Kwon, Byeonghak Lee, Joohee Lee, Jooyoung Lee, Sangyub Lee, Dukjae Moon, Mincheol Son, and Hyojin Yoon. AIMer. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

KKW18. Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In Lie et al. [LMBW18], pages 525–537.

LAZ19. Xingye Lu, Man Ho Au, and Zhenfei Zhang. Raptor: A practical lattice-based (linkable) ring signature. In Robert H. Deng, Valérie Gauthier-Umaña, Martín Ochoa, and Moti Yung, editors, *ACNS 19International Conference on Applied Cryptography and Network Security*, volume 11464 of *LNCS*, pages 110–130. Springer, Cham, June 2019.

LMBW18. David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors. *ACM CCS 2018*. ACM Press, October 2018.

LN22. Vadim Lyubashevsky and Ngoc Khanh Nguyen. BLOOM: Bimodal lattice one-out-of-many proofs and applications. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part IV*, volume 13794 of *LNCS*, pages 95–125. Springer, Cham, December 2022.

LNP$^+$24. San Ling, Khoa Nguyen, Duong Hieu Phan, Khai Hanh Tang, Huaxiong Wang, and Yanhong Xu. Fully dynamic attribute-based signatures for circuits from codes. In Tang and Teague [TT24], pages 37–73.

McE78. Robert J. McEliece. A public-key cryptosystem based on algebraic coding theory. The deep space network progress report 42-44, Jet Propulsion Laboratory, California Institute of Technology, January/February 1978. https://ipnpr.jpl.nasa.gov/progress_report2/42-44/44N.PDF.

NTWZ19. Khoa Nguyen, Hanh Tang, Huaxiong Wang, and Neng Zeng. New code-based privacy-preserving cryptographic constructions. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 25–55. Springer, Cham, December 2019.

NY90. Moni Naor and Moti Yung. Public-key cryptosystems provably secure against chosen ciphertext attacks. In *22nd ACM STOC*, pages 427–437. ACM Press, May 1990.

OTX24. Ying Ouyang, Deng Tang, and Yanghong Xu. Code-Based Zero-Knowledge from VOLE-in-the-Head and Their Applications: Simpler, Faster, and Smaller. *Cryptology ePrint Archive, Report 2024/1414*, 2024.

Roy22. Lawrence Roy. SoftSpokenOT: Quieter OT extension from small-field silent VOLE in the minicrypt model. In Yevgeniy Dodis and Thomas

Shrimpton, editors, *CRYPTO 2022, Part I*, volume 13507 of *LNCS*, pages 657–687. Springer, Cham, August 2022.

RST01. Ronald L. Rivest, Adi Shamir, and Yael Tauman. How to leak a secret. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 552–565. Springer, Berlin, Heidelberg, December 2001.

SAB$^+$22. Peter Schwabe, Roberto Avanzi, Joppe Bos, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Vadim Lyubashevsky, John M. Schanck, Gregor Seiler, Damien Stehlé, and Jintai Ding. CRYSTALS-KYBER. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

Sha90. Adi Shamir. An efficient identification scheme based on permuted kernels (extended abstract) (rump session). In Gilles Brassard, editor, *CRYPTO'89*, volume 435 of *LNCS*, pages 606–609. Springer, New York, August 1990.

SSW20. Peter Schwabe, Douglas Stebila, and Thom Wiggers. Post-quantum TLS without handshake signatures. In Jay Ligatti, Xinming Ou, Jonathan Katz, and Giovanni Vigna, editors, *ACM CCS 2020*, pages 1461–1480. ACM Press, November 2020.

Ste94. Jacques Stern. A new identification scheme based on syndrome decoding. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 13–21. Springer, Berlin, Heidelberg, August 1994.

Ste96. Jacques Stern. A new paradigm for public key identification. *IEEE Transactions on Information Theory (IEEE IT)*, 42(6):1757–1768, 1996.

TT24. Qiang Tang and Vanessa Teague, editors. *PKC 2024, Part I*, volume 14601 of *LNCS*. Springer, Cham, April 2024.

Vér97. Pascal Véron. Improved Identification Schemes Based on Error-Correcting Codes. *Applicable Algebra in Engineering, Communication and Computing*, 1997.

YSWW21. Kang Yang, Pratik Sarkar, Chenkai Weng, and Xiao Wang. QuickSilver: Efficient and affordable zero-knowledge proofs for circuits and polynomials over any field. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 2986–3001. ACM Press, November 2021.

ZCD$^+$20. Greg Zaverucha, Melissa Chase, David Derler, Steven Goldfeder, Claudio Orlandi, Sebastian Ramacher, Christian Rechberger, Daniel Slamanig, Jonathan Katz, Xiao Wang, Vladmir Kolesnikov, and Daniel Kales. Picnic. Technical report, National Institute of Standards and Technology, 2020. available at https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/round-3-submissions.

A Additional details on operations on VOLE correlations

Algorithm A.1: LinearCombination$(c_0, c_1, \dots, c_n)$

Prover's computation: P.LinearCombination$(c_0, c_1, \dots, c_n, (u_1, v_1), \dots, (u_n, v_n))$

Prover's input: Coefficients of linear combination $c_0, c_1, \dots, c_n \in \mathbb{F}_2$, VOLE correlation inputs $(u_1, v_1), \dots, (u_n, v_n) \in (\mathbb{F}_2 \times \mathbb{F}_{2^\kappa})^n$.

Prover's output: VOLE correlations (u, v) for the linear combination of secret inputs.

Computes $u = c_0 + \sum_{i=1}^n c_i u_i$ and $v = \sum_{i=1}^n c_i v_i$.

Verifier's computation: V.LinearCombination$(c_0, c_1, \dots, c_n, \Delta, q_1, \dots, q_n)$

Verifier's input: Coefficients of linear combination $c_0, c_1, \dots, c_n \in \mathbb{F}_2$, VOLE correlation inputs $\Delta, q_1, \dots q_n \in \mathbb{F}_{2^\kappa}^{n+1}$

Verifier's output: VOLE correlation q for linear combination of secret inputs.

Computes $q = c_0\Delta + \sum_{i=1}^n c_i q_i$.

In the following, let $d_1 \geq d_2$ without loss of generality, also let $d = d_1 + d_2$.

Algorithm A.2: Add$\left(\llbracket s_1 \rrbracket^{(d_1)}, \llbracket s_2 \rrbracket^{(d_2)}\right)$

Public information:

Degrees of input VOLE correlations d_1, d_2.

Prover's computation: P.Add$\left(\llbracket s_1 \rrbracket^{(d_1)}, \llbracket s_2 \rrbracket^{(d_2)}\right)$

Prover's input: VOLE correlations represented as polynomials $f_{s_1}(X)$ and $f_{s_2}(X)$.

Prover's output: VOLE correlation $\llbracket s \rrbracket^{(d_1)}$ for addition of secret inputs.

Computes $\llbracket s \rrbracket^{(d_1)} = f_s(X) = f_{s_1}(X) + f_{s_2}(X)X^{d_1 - d_2}$ where $s = s_1 + s_2$.

Verifier's computation: V.Add$(\Delta, q_{s_1}, q_{s_2})$

Verifier's input: Δ, $q_{s_1} = f_{s_1}(\Delta)$, and $q_{s_2} = f_{s_2}(\Delta)$

Verifier's output: VOLE correlation q_s for addition of secret inputs.

Computes $q_s = q_{s_1} + q_{s_2}\Delta^{d_1 - d_2}$.

Algorithm A.3: Multiply$\left(\llbracket s_1 \rrbracket^{(d_1)}, \llbracket s_2 \rrbracket^{(d_2)}\right)$

Public information:

Degrees of input VOLE correlations d_1, d_2.

Prover's computation: P.Multiply$\left([\![s_1]\!]^{(d_1)}, [\![s_2]\!]^{(d_2)}\right)$

Prover's input: VOLE correlations represented as polynomials $f_{s_1}(X)$ and $f_{s_2}(X)$.

Prover's output: VOLE correlation $[\![s]\!]^{(d_1)}$ for multiplication of secret inputs.

Computes $[\![s]\!]^{(d)} = f_s(X) = f_{s_1}(X) f_{s_2}(X)$ where $s = s_1 s_2$.

Verifier's computation: V.Multiply$(\Delta, q_{s_1}, q_{s_2})$

Verifier's input: Δ, $q_{s_1} = f_{s_1}(\Delta)$ and $q_{s_2} = f_{s_2}(\Delta)$

Verifier's output: VOLE correlation q_s for multiplication of secret inputs.

Computes $q_s = q_{s_1} q_{s_2}$.

B PoK of secret permutation

Algorithm B.1:
P.Check-Permutation$\left(\boldsymbol{P}, ([\![u_{i,k}]\!])_{i\in[0,n-1],k\in[0,d-1]}, ([\![s_{k'}]\!])_{k'\in[0,d-2]}, \mathsf{seed}\right)$

Public information and inputs

Public information: Matrix dimension n of the secret permutation matrix, $d = \lceil \log n \rceil$.

Prover's input: Secret permutation matrix $\boldsymbol{P}$ represented as n positions $(\mathsf{pos}_0, \ldots, \mathsf{pos}_{n-1})$, $n \cdot d$ VOLE correlations $[\![u_{i,k}]\!]$ for random $(u_{i,k}, v_{i,k}) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$ represented as polynomials $f_{u_{i,k}}(X) = u_{i,k}X + v_{i,k}$ for $(i,k) \in [0, n-1] \times [0, d-1]$, $(d-1)$ VOLE correlations $[\![s_{k'}]\!]$ for random $r_{k'}, s_{k'} \in \mathbb{F}_{2^\lambda}$ represented as $f_{s_{k'}}(X) = s_{k'}X + r_{k'}$ for $k' \in [0, d-2]$, $\mathsf{seed} \in \mathbb{F}_{2^\lambda}$.

Output

Prover's output: Proof $([\![a]\!], \boldsymbol{t}, \mathsf{seed})$ that $\boldsymbol{P}$ is a permutation matrix.

```
        Compute VOLE correlations for $\boldsymbol{P}$
1 :     $(\boldsymbol{t}, [\![\boldsymbol{z}]\!]^{(d)}, [\![\mathsf{ColCheck}]\!]^{(d)}) \leftarrow$ P.VOLE-Permutation$(\boldsymbol{P}, ([\![u_{i,k}]\!])_{i\in[0,n-1],k\in[0,d-1]})$
2 :     Parse $[\![\mathsf{ColCheck}]\!]^{(d)}$ as $(f_0(X), f_1(X), \ldots, f_{n-1}(X))$

        Merge polynomials and run CheckZero
3 :     $\boldsymbol{\alpha} \xleftarrow{\$,\mathsf{seed}} \in \mathbb{F}_{2^\lambda}^n$
4 :     $f(X) = \sum_{j=0}^{n-1} \alpha_j \cdot f_j(X)$
5 :     $[\![a]\!] \leftarrow$ P.CheckZero$(f(X), ([\![s_{k'}]\!])_{k'\in[0,d-2]})$

6 :     $\mathsf{proof} = ([\![a]\!], \boldsymbol{t}, \mathsf{seed})$
7 :     return $\mathsf{proof}$
```

Algorithm B.2:
V.Check-Permutation$\left(\text{proof}, (q_{u_{i,k}})_{i\in[0,n-1],k\in[0,d-1]}, (q_{s_{k'}})_{k'\in[0,d-2]}, \Delta\right)$

Public information and inputs

Public information: Matrix dimension n of the secret permutation matrix, $d = \lceil \log n \rceil$.
Verifier's input: proof $= (\llbracket a \rrbracket, \boldsymbol{t}, \text{seed})$, $q_{u_{i,k}} = f_{u_{i,k}}(\Delta)$ for $(i,k) \in [0,n-1] \times [0,d-1]$, $q_{s_{k'}} = f_{s_{k'}}(\Delta)$ for $k' \in [0,d-2]$, Δ.

Output

Verifier's output: Boolean indicating if proof is a valid proof for some permutation matrix $\boldsymbol{P}$.

```
    Compute VOLE correlations for $\boldsymbol{P}$
1 : $(q_{\boldsymbol{z}}, q_{\mathsf{ColCheck}}) \leftarrow$ V.VOLE-Permutation$(\boldsymbol{t}, (q_{u_{i,k}})_{i\in[0,n-1],k\in[0,d-1]}, \Delta)$

    Merge polynomials and run CheckZero
2 : $\boldsymbol{\alpha} \xleftarrow{\$,\mathsf{seed}} \in \mathbb{F}_{2^\lambda}^n$
3 : $q = \sum_{j=0}^{n-1} \alpha_j \cdot q_{\mathsf{ColCheck}_j}$
4 : $b \leftarrow$ V.CheckZero $(\llbracket a \rrbracket, q, (q_{s_{k'}})_{k'\in[0,d-2]}, \Delta)$

5 : return $b$.
```

C PoK for the Regular SD problem

Algorithm C.1:
P.Check-Regular-SD$\left(\boldsymbol{x}, \text{pk}, \llbracket u_{i,k} \rrbracket_{i\in[0,\omega-1],k\in[0,d-1]}, (\llbracket s_{k'} \rrbracket)_{k'\in[0,d-2]}, \text{seed}\right)$

Public information and inputs

Public information: Weight of the vector ω, length of the vector n, $d = \lceil \log n/\omega \rceil$.
Prover's input: Secret vector $\boldsymbol{x}$ represented as ω positions $(\mathsf{pos}_0, \ldots, \mathsf{pos}_{\omega-1})$, public key pk $= (\boldsymbol{H}, \boldsymbol{y}) \in \mathbb{F}_2^{m\times n} \times \mathbb{F}_2^m$, $\omega \cdot d$ VOLE correlations $\llbracket u_{i,k} \rrbracket$ for random $(u_{i,k}, v_{i,k}) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$ represented as polynomials $f_{u_{i,k}}(X) = u_{i,k}X + v_{i,k}$ for $(i,k) \in [0,\omega-1] \times [0,d-1]$, $(d-1)$ VOLE correlations $\llbracket s_{k'} \rrbracket$ for random $r_{k'}, s_{k'} \in \mathbb{F}_{2^\lambda}$ represented as $f_{s_{k'}}(X) = s_{k'}X + r_{k'}$ for $k' \in [0,d-2]$, seed $\in \mathbb{F}_{2^\lambda}$.

Output

Prover's output: Proof $(\llbracket a \rrbracket, \boldsymbol{t}, \text{seed})$ that $\boldsymbol{x}$ is a solution to the given Regular SD instance.

```
        Compute elementary vector $\boldsymbol{x}_i$ for each block
1 :     for $i \in [0, \omega - 1]$
2 :         $(\boldsymbol{t}_i, [\![\boldsymbol{x}_i]\!]^{(d)}) \leftarrow$ P.VOLE-ElementaryVector$\big(\mathsf{pos}_i, ([\![u_{i,k}]\!])_{k \in [0,d-1]}\big)$
3 :     endfor

        Concatenate $\boldsymbol{x}_i$ into $\boldsymbol{x}$ such that $w_H(\boldsymbol{x}) \leq \omega$
4 :     $[\![\boldsymbol{x}]\!] = ([\![\boldsymbol{x}_0]\!] \,||\, \cdots \,||\, [\![\boldsymbol{x}_{\omega-1}]\!])$
5 :     $\boldsymbol{t} = (\boldsymbol{t}_0, \cdots, \boldsymbol{t}_{\omega-1})$

6 :     Compute $\boldsymbol{Hx} - \boldsymbol{y}$
7 :     for $i \in [0, m - 1]$
8 :         $f_i(X) = \sum_{j=0}^{n-1} h_{i,j} \cdot [\![\boldsymbol{x}_j]\!]^{(d)} - \boldsymbol{y}_i \cdot X^d$
9 :     endfor

        Merge polynomials and run CheckZero
10 :    $\boldsymbol{\alpha} \xleftarrow{\$,\mathsf{seed}} \in \mathbb{F}_{2^\lambda}^m$
11 :    $f(X) = \sum_{i=0}^{m-1} \alpha_i \cdot f_i(X)$
12 :    $[\![a]\!] \leftarrow$ P.CheckZero$\big(f(X), ([\![s_{k'}]\!])_{k' \in [0,d-2]}\big)$

13 :    $\mathsf{proof} = ([\![a]\!], \boldsymbol{t}, \mathsf{seed})$
14 :    return $\mathsf{proof}$
```

Algorithm C.2:
V.Check-Regular-SD$\big(\mathsf{proof}, \mathsf{pk}, (q_{u_{i,k}})_{i \in [0,\omega-1], k \in [0,d-1]}, (q_{s_{k'}})_{k' \in [0,d-2]}, \varDelta\big)$

Public information and inputs

Public information: Weight of the vector ω, length of the vector n, $d = \lceil \log n/\omega \rceil$.
Verifier's input: $\mathsf{proof} = ([\![a]\!], \boldsymbol{t}, \mathsf{seed})$, public key $\mathsf{pk} = (\boldsymbol{H}, \boldsymbol{y}) \in \mathbb{F}_2^{m \times n} \times \mathbb{F}_2^m$, $q_{u_{i,k}} = f_{u_{i,k}}(\varDelta)$ for $(i,k) \in [0, \omega-1] \times [0, d-1]$, $q_{s_{k'}} = f_{s_{k'}}(\varDelta)$ for $k' \in [0, d-2]$, $\varDelta$.

Output

Verifier's output: Boolean indicating if proof is a valid proof of a Regular SD solution.

```
     Compute VOLE correlations for $\boldsymbol{x}_i$
1 :  for $i \in [0, \omega - 1]$
2 :    $q_{\boldsymbol{x}_i} \leftarrow$ V.VOLE-ElementaryVector$\big(\boldsymbol{t}_i, (q_{u_{i,k}})_{k \in [0,d-1]}, \Delta\big)$
3 :  endfor

4 :  Compute VOLE correlations for $\boldsymbol{x}$
5 :  $q_{\boldsymbol{x}} = (q_{\boldsymbol{x}_0} \,||\, \cdots \,||\, q_{\boldsymbol{x}_{\omega-1}})$

6 :  Compute VOLE correlations for $\boldsymbol{Hx} - \boldsymbol{y}$
7 :  for $i \in [0, m - 1]$
8 :    $q_i = \sum_{j=0}^{n-1} h_{i,j} \cdot q_{\boldsymbol{x}_j} - \boldsymbol{y}_i \cdot \Delta^d$
9 :  endfor

     Merge polynomials and run CheckZero
10 : $\boldsymbol{\alpha} \xleftarrow{\$,\mathsf{seed}} \in \mathbb{F}_{2^\lambda}^m$
11 : $q = \sum_{i=0}^{m-1} \alpha_i \cdot q_i$
12 : $b \leftarrow$ V.CheckZero $\big([\![a]\!], q, (q_{s_{k'}})_{k' \in [0,d-2]}, \Delta\big)$

13 : return $b$.
```

D Ring Signatures

D.1 SD-based scheme

Algorithm D.1: P.Ring-Share-PublicKey-SD$\Big(i^*, ([\![u_k]\!])_{k \in [0,\bar{d}-1]}, \mathsf{pk}_{\mathcal{R}}\Big)$

Public information and inputs

Public information: Length of the vector $\bar{n}$, $\bar{d} = \lceil \log \bar{n} \rceil$.

Prover's input: Secret position $i^* \in [0, \bar{n}-1]$, $\bar{d}$ VOLE correlations $[\![u_k]\!]$ for random $(u_k, v_k) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$ represented as polynomials $f_{u_k}(X) = u_k X + v_k$ for $k \in [0, \bar{d} - 1]$, a ring of $\bar{n}$ public keys $\mathsf{pk}_{\mathcal{R}} = ((\boldsymbol{H}_0, \boldsymbol{y}_0), \ldots, (\boldsymbol{H}_{i^*}, \boldsymbol{y}_{i^*}), \ldots (\boldsymbol{H}_{\bar{n}-1}, \boldsymbol{y}_{\bar{n}-1}))$.

Output

Prover's output: Masked secret $\bar{\boldsymbol{t}} \in \mathbb{F}_2^{\bar{d}}$, $[\![\boldsymbol{H}]\!]^{(\bar{d})}$, $[\![\boldsymbol{y}]\!]^{(\bar{d})}$.

```
Compute selection vector
1 :  $(\bar{\boldsymbol{t}}, [\![\boldsymbol{z}]\!]^{(\bar{d})}) \leftarrow$ P.VOLE-ElementaryVector$(i^*, ([\![u_k]\!])_{k\in[0,\bar{d}-1]})$

Compute $\boldsymbol{H}$ and $\boldsymbol{y}$
2 :  $[\![\boldsymbol{H}]\!]^{(\bar{d})} = \sum_{i=0}^{\bar{n}-1}[\![z_i]\!]^{(\bar{d})} \cdot \boldsymbol{H}_i$   // Multiply each coordinates of $\boldsymbol{H}_i$ by $[\![z_i]\!]^{(\bar{d})}$
     $[\![\boldsymbol{y}]\!]^{(\bar{d})} = \sum_{i=0}^{\bar{n}-1}[\![z_i]\!]^{(\bar{d})} \cdot \boldsymbol{y}_i$

3 :  return $(\bar{\boldsymbol{t}}, [\![\boldsymbol{H}]\!]^{(\bar{d})}, [\![\boldsymbol{y}]\!]^{(\bar{d})})$
```

Algorithm D.2: P.Check-Ring-SD$(\boldsymbol{x}_{i^*}, \mathsf{pk}_{\mathcal{R}}, [\![u_{i,k}]\!]_{i\in[0,\omega-1],k\in[0,d-1]}, [\![u_k]\!]_{k\in[0,\bar{d}-1]}, ([\![s_{k'}]\!])_{k'\in[0,d+\bar{d}-2]}, \mathsf{seed})$

Public information and inputs

Public information: Weight of the vector ω, length of the vector n, $d = \lceil \log n \rceil$, size $\bar{n}$ of the ring $\mathcal{R}$ and $\bar{d} = \lceil \log \bar{n} \rceil$.

Prover's input: Secret vector $\boldsymbol{x}_{i^*}$ represented as ω positions $(\mathsf{pos}_0, \dots, \mathsf{pos}_{\omega-1})$, $\mathsf{pk}_{\mathcal{R}} = ((\boldsymbol{H}_0, \boldsymbol{y}_0), \dots, (\boldsymbol{H}_{i^*}, \boldsymbol{y}_{i^*}), \dots (\boldsymbol{H}_{\bar{n}-1}, \boldsymbol{y}_{\bar{n}-1}))$ where $\forall i \in [0, \bar{n}-1]$ $(\boldsymbol{H}_i, \boldsymbol{y}_i) \in \mathbb{F}_2^{m\times n} \times \mathbb{F}_2^m$, $\omega \cdot d$ VOLE correlations $[\![u_{i,k}]\!]$ for random $(u_{i,k}, v_{i,k}) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$ represented as polynomials $f_{u_{i,k}}(X) = u_{i,k}X + v_{i,k}$ for $(i,k) \in [0, \omega-1] \times [0, d-1]$, $\bar{d}$ VOLE correlations $[\![u_k]\!]$ for random $(u_k, v_k) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$ represented as polynomials $f_{u_k}(X) = u_kX + v_k$ for $k \in [0, \bar{d}-1]$, $(d + \bar{d} - 1)$ VOLE correlations $[\![s_{k'}]\!]$ for random $r_{k'}, s_{k'} \in \mathbb{F}_{2^\lambda}$ represented as $f_{s_{k'}}(X) = s_{k'}X + r_{k'}$ for $k' \in [0, d + \bar{d} - 2]$, $\mathsf{seed} \in \mathbb{F}_{2^\lambda}$.

Output

Prover's output: Proof $([\![a]\!], \boldsymbol{t}, \bar{\boldsymbol{t}}, \mathsf{seed})$ that $\boldsymbol{x}_{i^*}$ is a solution to the SD instance defined by $\mathsf{pk}_{i^*} = (\boldsymbol{H}_{i^*}, \boldsymbol{y}_{i^*})$ and $\mathsf{pk}_{i^*} \in \mathsf{pk}_{\mathcal{R}}$.

```
      Compute $\boldsymbol{x}$ from its support
1 :   $(\boldsymbol{t}, [\![\boldsymbol{x}]\!]^{(d)}) \leftarrow$ P.VOLE-HammingWeight$\big(\boldsymbol{x}_{i^*}, ([\![u_{i,k}]\!])_{i\in[0,\omega-1],k\in[0,d-1]}\big)$

      Compute $\boldsymbol{H}$ and $\boldsymbol{y}$
2 :   $(\bar{\boldsymbol{t}}, [\![\boldsymbol{H}]\!]^{(\bar{d})}, [\![\boldsymbol{y}]\!]^{(\bar{d})}) \leftarrow$ P.Ring-Share-PublicKey-SD$\big(i^*, ([\![u_k]\!])_{k\in[0,\bar{d}-1]}, \mathsf{pk}_{\mathcal{R}}\big)$

      Compute $\boldsymbol{Hx} - \boldsymbol{y}$
3 :   for $i \in [0, m-1]$
4 :     $f_i(X) = \sum_{j=0}^{n-1} [\![h_{i,j}]\!]^{(\bar{d})} \cdot [\![\boldsymbol{x}_j]\!]^{(d)} - [\![y_i]\!]^{(\bar{d})} \cdot X^d$
5 :   endfor

      Merge polynomials and run CheckZero
6 :   $\boldsymbol{\alpha} \xleftarrow{\$,\mathsf{seed}} \in \mathbb{F}_{2^\lambda}^m$
7 :   $f(X) = \sum_{i=0}^{m-1} \alpha_i \cdot f_i(X)$
8 :   $[\![a]\!] \leftarrow$ P.CheckZero$\big(f(X), ([\![s_{k'}]\!])_{k'\in[0,d+\bar{d}-2]}\big)$

9 :   $\mathsf{proof} = ([\![a]\!], \boldsymbol{t}, \bar{\boldsymbol{t}}, \mathsf{seed})$
10 :  return proof
```

Algorithm D.3: V.Check-Ring-SD$(\mathsf{proof}, \mathsf{pk}_{\mathcal{R}}, (q_{u_{i,k}})_{i\in[0,\omega-1],k\in[0,d-1]}, (q_{u_k})_{k\in[0,\bar{d}-1]}, (q_{s_{k'}})_{k'\in[0,d+\bar{d}-2]}, \varDelta)$

Public information and inputs

Public information: Weight of the vector ω, length of the vector n, $d = \lceil \log n \rceil$, size $\bar{n}$ of the ring $\mathcal{R}$ and $\bar{d} = \lceil \log \bar{n} \rceil$.
Verifier's input: $\mathsf{proof} = ([\![a]\!], \boldsymbol{t}, \bar{\boldsymbol{t}}, \mathsf{seed})$, $\mathsf{pk}_{\mathcal{R}} := ((\boldsymbol{H}_0, \boldsymbol{y}_0), \ldots, (\boldsymbol{H}_{i^*}, \boldsymbol{y}_{i^*}), \ldots (\boldsymbol{H}_{\bar{n}-1}, \boldsymbol{y}_{\bar{n}-1}))$ where $\forall i \in [0, \bar{n}-1]$ $(\boldsymbol{H}_i, \boldsymbol{y}_i) \in \mathbb{F}_2^{m\times n} \times \mathbb{F}_2^m$, $q_{u_{i,k}} = f_{u_{i,k}}(\varDelta)$ for $(i,k) \in [0,\omega-1]\times[0,d-1]$, $q_{u_k} = f_{u_k}(\varDelta)$ for $k \in [0, \bar{d}-1]$, $q_{s_{k'}} = f_{s_{k'}}(\varDelta)$ for $k' \in [0, d+\bar{d}-2]$, $\varDelta$.

Output

Verifier's output: Boolean indicating if proof is a valid proof of a SD solution for ring $\mathcal{R}$.

```
    Compute VOLE correlations for $\boldsymbol{x}$
1 : $q_{\boldsymbol{x}} \leftarrow$ V.VOLE-HammingWeight$\big(\boldsymbol{t}, (q_{u_{i,k}})_{i\in[0,\omega-1],k\in[0,d-1]}, \Delta\big)$

    Compute VOLE correlations for selection vector
2 : $q^* \leftarrow$ V.VOLE-ElementaryVector$\big(\bar{\boldsymbol{t}}, (q_{u_k})_{k\in[0,\bar{d}-1]}, \Delta\big)$
3 :
    Compute VOLE correlations for $\boldsymbol{H}$ and $\boldsymbol{y}$
4 : $q_{\boldsymbol{H}} = \sum_{i=0}^{\bar{n}-1} q_i^* \cdot \boldsymbol{H}_i$
    $q_{\boldsymbol{y}} = \sum_{i=0}^{\bar{n}-1} q_i^* \cdot \boldsymbol{y}_i$

    Compute VOLE correlations for $\boldsymbol{Hx} - \boldsymbol{y}$
5 : for $i \in [0, m-1]$
6 :     $q_i = \sum_{j=0}^{n-1} q_{\boldsymbol{H}\,i,j} \cdot q_{\boldsymbol{x}_j} - q_{\boldsymbol{y}_i} \cdot \Delta^d$
7 : endfor

    Merge polynomials and run CheckZero
8 : $\boldsymbol{\alpha} \xleftarrow{\$,\mathsf{seed}} \in \mathbb{F}_{2^\lambda}^m$
9 : $q = \sum_{i=0}^{m-1} \alpha_i \cdot q_i$
10 : $b \leftarrow$ V.CheckZero $\big([\![a]\!], q, (q_{s_{k'}})_{k'\in[0,d+\bar{d}-2]}, \Delta\big)$

11 : return $b$.
```

D.2 PKP-based scheme

Algorithm D.4: P.Ring-Share-PublicKey-PKP$\Big(i^*, ([\![u_k]\!])_{k\in[0,\bar{d}-1]}, \mathsf{pk}_{\mathcal{R}}\Big)$

Public information and inputs

Public information: Length of the vector $\bar{n}$, $\bar{d} = \lceil \log \bar{n} \rceil$.

Prover's input: Secret position $i^* \in [0, \bar{n}-1]$, $\bar{d}$ VOLE correlations $[\![u_k]\!]$ for random $(u_k, v_k) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$ represented as polynomials $f_{u_k}(X) = u_k X + v_k$ for $k \in [0, \bar{d}-1]$, a ring of $\bar{n}$ public keys $\mathsf{pk}_{\mathcal{R}} = ((\boldsymbol{H}_0, \boldsymbol{x}_0), \ldots, (\boldsymbol{H}_{i^*}, \boldsymbol{x}_{i^*}), \ldots (\boldsymbol{H}_{\bar{n}-1}, \boldsymbol{x}_{\bar{n}-1}))$.

Output

Prover's output: Masked secret $\bar{\boldsymbol{t}} \in \mathbb{F}_2^{\bar{d}}$, $[\![\boldsymbol{H}]\!]^{(\bar{d})}$, $[\![\boldsymbol{x}]\!]^{(\bar{d})}$.

```
Compute selection vector
1 : $(\bar{\boldsymbol{t}}, [\![\boldsymbol{z}]\!]^{(\bar{d})}) \leftarrow$ P.VOLE-ElementaryVector$\big(i^*, ([\![u_k]\!])_{k\in[0,\bar{d}-1]}\big)$

Compute $\boldsymbol{H}$ and $\boldsymbol{x}$
2 : $[\![\boldsymbol{H}]\!]^{(\bar{d})} = \sum_{i=0}^{\bar{n}-1}[\![z_i]\!]^{(\bar{d})} \cdot \boldsymbol{H}_i$   // Multiply each coordinates of $\boldsymbol{H}_i$ by $[\![z_i]\!]^{(\bar{d})}$
3 : $[\![\boldsymbol{x}]\!]^{(\bar{d})} = \sum_{i=0}^{\bar{n}-1}[\![z_i]\!]^{(\bar{d})} \cdot \boldsymbol{x}_i$

4 : return $(\bar{\boldsymbol{t}}, [\![\boldsymbol{H}]\!]^{(\bar{d})}, [\![\boldsymbol{x}]\!]^{(\bar{d})})$
```

Algorithm D.5: P.Check-Ring-PKP($\boldsymbol{P}_{i^*}$, $\mathsf{pk}_{\mathcal{R}}$, $[\![u_{i,k}]\!]_{i\in[0,n-1],k\in[0,d-1]}$, $[\![u_k]\!]_{k\in[0,\bar{d}-1]}$, $([\![s_{k'}]\!])_{k'\in[0,d+2\cdot\bar{d}-2]}$, seed)

Public information and inputs

Public information: Matrix dimension n of the secret permutation matrix, $d = \lceil \log n \rceil$, size $\bar{n}$ of the ring $\mathcal{R}$ and $\bar{d} = \lceil \log \bar{n} \rceil$.

Prover's input: Secret permutation matrix $\boldsymbol{P}_{i^*}$ represented as n positions $(\mathsf{pos}_0, \dots, \mathsf{pos}_{n-1})$, $\mathsf{pk}_{\mathcal{R}} = ((\boldsymbol{H}_0, \boldsymbol{x}_0), \dots, (\boldsymbol{H}_{i^*}, \boldsymbol{x}_{i^*}), \dots (\boldsymbol{H}_{\bar{n}-1}, \boldsymbol{x}_{\bar{n}-1}))$ where $\forall i \in [0, \bar{n}-1]\, (\boldsymbol{H}_i, \boldsymbol{x}_i) \in \mathbb{F}_q^{m\times n} \times \mathbb{F}_q^n$, $n \cdot d$ VOLE correlations $[\![u_{i,k}]\!]$ for random $(u_{i,k}, v_{i,k}) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$ represented as polynomials $f_{u_{i,k}}(X) = u_{i,k}X + v_{i,k}$ for $(i,k) \in [0, n-1] \times [0, d-1]$, $\bar{d}$ VOLE correlations $[\![u_k]\!]$ for random $(u_k, v_k) \in \mathbb{F}_2 \times \mathbb{F}_{2^\lambda}$ represented as polynomials $f_{u_k}(X) = u_k X + v_k$ for $k \in [0, \bar{d}-1]$, $(d + 2\cdot\bar{d} - 1)$ VOLE correlations $[\![s_{k'}]\!]$ for random $r_{k'}, s_{k'} \in \mathbb{F}_{2^\lambda}$ represented as $f_{s_{k'}}(X) = s_{k'}X + r_{k'}$ for $k' \in [0, d + 2\cdot\bar{d} - 2]$, $\mathsf{seed} \in \mathbb{F}_{2^\lambda}$.

Output

Prover's output: Proof $([\![a]\!], \boldsymbol{t}, \bar{\boldsymbol{t}}, \mathsf{seed})$ that $\boldsymbol{P}_{i^*}$ is a solution to the PKP instance defined by $\mathsf{pk}_{i^*} = (\boldsymbol{H}_{i^*}, \boldsymbol{x}_{i^*})$ and $\mathsf{pk}_{i^*} \in \mathsf{pk}_{\mathcal{R}}$.

```
    Compute $\boldsymbol{P}$ in matrix form
1 : $(\boldsymbol{t}, [\![\boldsymbol{z}]\!]^{(d)}, [\![\mathsf{ColCheck}]\!]^{(d)}) \leftarrow$ P.VOLE-Permutation$\big(\boldsymbol{P}_{i^*}, ([\![u_{i,k}]\!])_{i\in[0,n-1],k\in[0,d-1]}\big)$
2 : Parse $[\![\mathsf{ColCheck}]\!]^{(d)}$ as $(f_0(X), f_1(X), \ldots, f_{n-1}(X))$

    Compute $\boldsymbol{H}$ and $\boldsymbol{x}$
3 : $(\bar{\boldsymbol{t}}, [\![\boldsymbol{H}]\!]^{(\bar{d})}, [\![\boldsymbol{x}]\!]^{(\bar{d})}) \leftarrow$ P.Ring-Share-PublicKey-PKP$\big(i^*, ([\![u_k]\!])_{k\in[0,\bar{d}-1]}, \mathsf{pk}_{\mathcal{R}}\big)$

    Compute $\boldsymbol{x}' = \boldsymbol{P}\boldsymbol{x}$
4 : for $i \in [0, n-1]$
5 :     $[\![\boldsymbol{x}'_i]\!]^{(d+\bar{d})} = \sum_{j=0}^{n-1} [\![z_{i,j}]\!]^{(d)} \cdot [\![\boldsymbol{x}_j]\!]^{(\bar{d})}$
6 : endfor

    Compute $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x}'$
7 : for $i \in [0, m-1]$
8 :     $[\![\boldsymbol{y}_i]\!]^{(d+2\cdot\bar{d})} = f_{i+n}(X) = \sum_{j=0}^{n-1} [\![h_{i,j}]\!]^{(\bar{d})} \cdot [\![\boldsymbol{x}'_j]\!]^{(d+\bar{d})}$
9 : endfor

    Merge polynomials and run CheckZero
10 : $\boldsymbol{\alpha} \xleftarrow{\$,\mathsf{seed}} \in \mathbb{F}_{2^\lambda}^{n+m}$
11 : $f(X) = \sum_{j=0}^{n+m-1} \alpha_j \cdot f_j(X)$
12 : $[\![a]\!] \leftarrow$ P.CheckZero$\big(f(X), ([\![s_{k'}]\!])_{k'\in[0,d+2\cdot\bar{d}-2]}\big)$

13 : $\mathsf{proof} = ([\![a]\!], \boldsymbol{t}, \bar{\boldsymbol{t}}, \mathsf{seed})$
14 : return proof
```

Algorithm D.6: V.Check-Ring-PKP(proof, $\mathsf{pk}_{\mathcal{R}}, (q_{u_{i,k}})_{i\in[0,n-1],k\in[0,d-1]}, (q_{u_k})_{k\in[0,\bar{d}-1]}, (q_{s_{k'}})_{k'\in[0,d+2\cdot\bar{d}-2]}, \Delta$)

Public information and inputs

Public information: Weight of the vector ω, length of the vector n, $d = \lceil \log n \rceil$, size $\bar{n}$ of the ring $\mathcal{R}$ and $\bar{d} = \lceil \log \bar{n} \rceil$.
Verifier's input: $\mathsf{proof} = ([\![a]\!], \boldsymbol{t}, \bar{\boldsymbol{t}}, \mathsf{seed})$, $\mathsf{pk}_{\mathcal{R}} := ((\boldsymbol{H}_0, \boldsymbol{x}_0), \ldots, (\boldsymbol{H}_{i^*}, \boldsymbol{x}_{i^*}), \ldots (\boldsymbol{H}_{\bar{n}-1}, \boldsymbol{x}_{\bar{n}-1}))$ where $\forall i \in [0, \bar{n}-1]$ $(\boldsymbol{H}_i, \boldsymbol{x}_i) \in \mathbb{F}_q^{m\times n} \times \mathbb{F}_q^n$, $q_{u_{i,k}} = f_{u_{i,k}}(\Delta)$ for $(i,k) \in [0, \omega-1] \times [0, d-1]$, $q_{u_k} = f_{u_k}(\Delta)$ for $k \in [0, \bar{d}-1]$, $q_{s_{k'}} = f_{s_{k'}}(\Delta)$ for $k' \in [0, d+2\cdot\bar{d}-2]$, Δ.

Output

Verifier's output: Boolean indicating if proof is a valid proof of a PKP solution for ring $\mathcal{R}$.

```
      Compute VOLE correlations for $\boldsymbol{P}$
1 :   $(q_{\boldsymbol{z}}, q_{\mathsf{ColCheck}}) \leftarrow$ V.VOLE-Permutation$\big(\boldsymbol{t}, (q_{u_{i,k}})_{i\in[0,n-1],k\in[0,d-1]}, \varDelta\big)$

      Compute VOLE correlations for selection vector
2 :   $q^* \leftarrow$ V.VOLE-ElementaryVector$\big(\bar{\boldsymbol{t}}, (q_{u_k})_{k\in[0,\bar{d}-1]}, \varDelta\big)$
3 :
      Compute VOLE correlations for $\boldsymbol{H}$ and $\boldsymbol{x}$
4 :   $q_{\boldsymbol{H}} = \sum_{i=0}^{\bar{n}-1} q_i^* \cdot \boldsymbol{H}_i$
      $q_{\boldsymbol{x}} = \sum_{i=0}^{\bar{n}-1} q_i^* \cdot \boldsymbol{x}_i$

      Compute VOLE correlations for $\boldsymbol{x}' = \boldsymbol{P}\boldsymbol{x}$
5 :   for $i \in [0, n-1]$
6 :     $q_{\boldsymbol{x}'_i} = \sum_{j=0}^{n-1} q_{i,j} \cdot q_{\boldsymbol{x}_j}$
7 :   endfor

      Compute VOLE correlations for $\boldsymbol{y} = \boldsymbol{H}\boldsymbol{x}'$
8 :   for $i \in [0, m-1]$
9 :     $q_{\boldsymbol{y}_i} = \sum_{j=0}^{n-1} q_{\boldsymbol{H}_{i,j}} \cdot q_{\boldsymbol{x}'_j}$
10 :  endfor

      Merge polynomials and run CheckZero
11 :  $\boldsymbol{\alpha} \xleftarrow{\$,\mathsf{seed}} \in \mathbb{F}_{2^\lambda}^{n+m}$
12 :  $q = \sum_{j=0}^{n-1} \alpha_j \cdot q_{\mathsf{ColCheck}_j} + \sum_{j=n}^{n+m-1} \alpha_j \cdot q_{\boldsymbol{y}_{j-n}}$
13 :  $b \leftarrow$ V.CheckZero $\big([\![a]\!], q, (q_{s_{k'}})_{k'\in[0,d+2\cdot\bar{d}-2]}, \varDelta\big)$

14 :  return $b$.
```