ReSolveD: Shorter Signatures from Regular Syndrome Decoding and VOLE-in-the-Head

Hongrui Cui
Shanghai Jiao Tong University
rickfreeman@sjtu.edu.cn

Hanlin Liu
Shanghai Qi Zhi Institute
hans1024@sjtu.edu.cn

Di Yan
State Key Laboratory of Cryptology
yand@sklc.org

Kang Yang
State Key Laboratory of Cryptology
yangk@sklc.org

Yu Yu
Shanghai Jiao Tong University
Shanghai Qi Zhi Institute
yuyu@yuyu.hk

Kaiyi Zhang
Shanghai Jiao Tong University
kzoacn@sjtu.edu.cn

Abstract

We present ReSolveD, a new candidate post-quantum signature scheme under the regular syndrome decoding (RSD) assumption for random linear codes, which is a well-established variant of the well-known syndrome decoding (SD) assumption. Our signature scheme is obtained by designing a new zero-knowledge proof for proving knowledge of a solution to the RSD problem in the recent VOLE-in-the-head framework using a sketching scheme to verify that a vector has weight exactly one. We achieve a signature size of 3.99 KB with a signing time of 27.3 ms and a verification time of 23.1 ms on a single core of a standard desktop for a 128-bit security level. Compared to the state-of-the-art code-based signature schemes, our signature scheme achieves $1.5\times \sim 2\times$ improvement in terms of the common "signature size + public-key size" metric, while keeping the computational efficiency competitive.

1 Introduction

Zero-knowledge (ZK) proof is an important cryptographic tool that enables a prover to convince a verifier of the validity of a statement without revealing any further information. ZK proofs find a lot of applications in various contexts, e.g., secure multi-party computation (MPC), machine learning, and blockchain. Using the Fiat-Shamir heuristic [FS87], we can transform public-coin zero-knowledge proofs into signature schemes. In particular, this is the main approach to building code-based signature schemes. The recent call of NIST for standardizing post-quantum signatures expressed its primary interest in additional signature schemes that are not based on structured lattices [NIS22], which promotes the research of non-lattice-based signature schemes, particularly code-based signatures.

The well-known syndrome decoding (SD) problem over a binary field $\mathbb{F}_2$ asks, given a matrix $\mathbf{H} \in \mathbb{F}_2^{(m-k)\times m}$ and a target vector $\boldsymbol{y} \in \mathbb{F}_2^{m-k}$, to recover a noise vector $\boldsymbol{e} \in \mathbb{F}_2^m$ such that $\mathbf{H} \cdot \boldsymbol{e} = \boldsymbol{y}$ for some sparse $\boldsymbol{e}$ of exact weight $w \ll m$. The worst-case SD problem in certain parameter regimes is known to be NP-hard [BMvT78, Bar94], and its average-case analogue is one of the most promising assumptions for post-quantum cryptography. In the seminal work from three decades ago, Stern [Ste94] introduced the first zero-knowledge proof to prove knowledge of

a solution to the SD problem. However, the communication cost is significant due to the high soundness error, and thus the signature size would be very large (i.e. about 37 KB for 128-bit security), when being compiled using the Fiat-Shamir transform. Since then, a few prior works (e.g., [Vér96, GG07, CVA10, MGS11, ACBH13]) optimized Stern's protocol, but the communication cost is still high. To avoid the issue of high soundness error, subsequent code-based signature schemes resort to different code-based problems, e.g., (a) LESS (and some subsequent improvements) [BMPS20, BBPS21, PS23] adopts the linear/permutation code equivalence problem, and (b) Durandal [ABG+19] depends on the rank SD problem over $\mathbb{F}_{2^m}$. Recently, based on the SD problem, the works [GPS22, FJR23, BGKM23, FJR22, CCJ23, AGH+23, MHJ+23] obtain significantly lower soundness errors by building zero-knowledge proofs based on the MPC-in-the-head paradigm [IKOS07], and achieve the best efficiency for now in terms of the common "signature size + public-key size" metric. We summarize the efficiency and assumptions of recent code-based signature schemes in Table 1. Among these schemes, Wave [DST19] is the only signature scheme that departs from the line that transforms zero-knowledge proofs with Fiat-Shamir. Instead, Wave adopts the hash-and-sign paradigm that depends on the existence of a code-based trapdoor permutation, which leads to a very large size of public keys. Wave achieves smaller signature sizes but relies on a non-standard code-based assumption. For the common "signature size + public-key size" metric, the Fiat-Shamir-based schemes still offer better performance.

In this work, we focus on designing a new code-based signature scheme under the regular syndrome decoding (RSD) assumption [AFS03], a well-established variant of the well-known SD assumption. Specifically, RSD is the same as SD, except for requiring that the noise vector $\boldsymbol{e}$ is regular, i.e., $\boldsymbol{e} \in \mathbb{F}_2^m$ is divided into w consecutive blocks of length m/w, where each block has exactly one noisy coordinate. Recent works [FJR22, LWYY22] presented a reduction from SD to RSD, which builds confidence in the hardness of the RSD problem from a theoretic point of view. Furthermore, the hardness of the RSD problem was thoroughly analyzed by Carozza et al. [CCJ23], which gives us more confidence. As far as we know, the regular structure of noises does not lead to significantly better attacks when the code rate is kept large (and thus the recent algebraic attack [BØ23] can be bypassed).

1.1 Our Contributions

In this paper, we put forward a new zero-knowledge (ZK) protocol on proving knowledge of a solution to the RSD problem over $\mathbb{F}_2$. By using the Fiat-Shamir transform, we compile the zero-knowledge protocol into a code-based signature scheme (called ReSolveD). Compared to the state-of-the-art code-based signature schemes, ReSolveD reduces the total size of the signature and public key by a factor of $1.5\times \sim 2\times$, while keeping the signing and verification performance competitive. In Table 1, we compare the efficiency of ReSolveD with the recent code-based signature schemes at the 128-bit security level. Note that LESS [PS23] and CF-LESS [CPS23] lack of parameter sets for 128-bit security, instead we use their parameters for NIST category 1 security. While the work [AGH+23] describes three variants of their code-based signature scheme, Table 1 only shows the third one that has the best performance among the three. The shortest version of the signature scheme [AGH+23] has a signature size that is closest to the first variant of ReSolveD among prior code-based signature schemes, but the signing and verification timings are more than $10\times$ larger than our scheme. Notice that the shortest version of [AGH+23] also offers the smallest size in the "signature + public key" metric. Our second variant achieves $1.5\times$ improvement in terms of that metric with half of the running time. In Section 6.2, we also compare ReSolveD with other kinds of post-quantum signature schemes.

While most code-based signature schemes in the Fiat-Shamir line adopt the MPC-in-the-head

Table 1: **Comparison of code-based signature schemes for 128-bit security level.** Reported runtimes are extracted from original publications, using a 3.5 GHz Intel Xeon E3-1240 v5 for Wave [DST19], a 2.8 GHz Intel Core i5-7440HQ for Durandal [ABG+19], a 2.1 GHz Intel Core i7-12700 CPU for LESS [PS23], a 3.8 GHz Intel Core i7 supports AVX2 and AES instructions for [FJR23, FJR22], a 3.1 GHz Intel Core i9-9990K using AVX2 for [AGH+23], an Intel Xeon E-2378 with frequency fixed at 2.6 GHz for [MHJ+23] and a conservative upper bound assuming a 3.8 GHz CPU for [CCJ23]. We benchmarked our ReSolveD-128 on a Ubuntu 20.04 LTS machine with AMD Ryzen 5 3600 CPU and 16GB of RAM using AVX2.

Scheme	Sizes in KB			Runtimes in ms		Assumption
	$\|\mathsf{sig}\|$	$\|\mathsf{pk}\|$	$\|\mathsf{sig}\|+\|\mathsf{pk}\|$	t_{sign}	t_{verify}	
Wave [DST19]	1.59	3276.8	3278.39	300	–	large-weight SD over $\mathbb{F}_3$, $(U, U+V)$-codes indist.
Durandal-I [ABG+19]	4.06	15.25	19.31	4	5	Rank-SD over $\mathbb{F}_{2^m}$
Durandal-II [ABG+19]	5.02	18.61	23.63	5	6	Rank-SD over $\mathbb{F}_{2^m}$
LESS-FM-I [BBPS21]	15.2	9.77	24.97	–	–	Linear Code Equivalence
LESS-FM-II [BBPS21]	5.25	205.74	210.95	–	–	Perm. Code Equivalence
LESS-FM-III [BBPS21]	10.39	11.57	21.96	–	–	Perm. Code Equivalence
LESS-1b [PS23]	8.4	13.6	22	125.52	129.24	Linear Code Equivalence
LESS-1i [PS23]	5.8	40.8	46.6	121.10	125.43	Linear Code Equivalence
LESS-1s [PS23]	5.0	95.2	100.2	98.38	101.62	Linear Code Equivalence
CF-LESS-1(s=2) [CPS23]	2.42	13.61	16.04	–	–	CF Code Equivalence
CF-LESS-1(s=4) [CPS23]	1.80	40.81	42.61	–	–	CF Code Equivalence
[GPS22]-256	23.98	0.11	24.09	–	–	SD over $\mathbb{F}_{256}$
[GPS22]-1024	19.76	0.12	19.88	–	–	SD over $\mathbb{F}_{1024}$
[FJR23]-fast	22.6	0.09	22.69	12.9	12.2	SD over $\mathbb{F}_2$
[FJR23]-short	16.0	0.09	16.09	62.3	56.6	SD over $\mathbb{F}_2$
[BGKM23]-Sig1	24.0	0.1	24.1	–	–	SD over $\mathbb{F}_2$
[BGKM23]-Sig2	19.3	0.2	19.5	–	–	(QC)SD over $\mathbb{F}_2$
[BGKM23]-Sig3	15.6	0.2	15.8	–	–	(QC)SD over $\mathbb{F}_2$
[FJR22]-Var1f	15.6	0.09	15.69	–	–	SD over $\mathbb{F}_2$
[FJR22]-Var1s	10.9	0.09	10.99	–	–	SD over $\mathbb{F}_2$
[FJR22]-Var2f	17.0	0.09	17.09	13.4	12.7	SD over $\mathbb{F}_2$
[FJR22]-Var2s	11.8	0.09	11.89	64.2	60.7	SD over $\mathbb{F}_2$
[FJR22]-Var3f	11.5	0.14	11.64	6.4	5.9	SD over $\mathbb{F}_{256}$
[FJR22]-Var3s	8.26	0.14	8.4	29.5	27.1	SD over $\mathbb{F}_{256}$
[AGH+23]-fast	11.83	0.14	11.97	1.30	0.98	SD over $\mathbb{F}_{256}$
[AGH+23]-short	8.28	0.14	8.42	2.87	2.59	SD over $\mathbb{F}_{256}$
[AGH+23]-shorter	6.63	0.14	6.77	26.43	25.79	SD over $\mathbb{F}_{256}$
[AGH+23]-shortest	5.56	0.14	5.7	320.66	312.67	SD over $\mathbb{F}_{256}$
[MHJ+23]-Vanilla-short	8.27	0.14	8.6	4.5	4.17	SD over $\mathbb{F}_{256}$
[MHJ+23]-Vanilla-shorter	6.6	0.14	6.94	45.06	42.02	SD over $\mathbb{F}_{256}$
[MHJ+23]-PoW-short	7.78	0.14	8.11	4.34	4	SD over $\mathbb{F}_{256}$
[MHJ+23]-PoW-shorter	6.06	0.14	6.34	42.55	39.75	SD over $\mathbb{F}_{256}$
[CCJ23]-rsd-f	12.52	0.09	12.61	2.8*	–	RSD over $\mathbb{F}_2$
[CCJ23]-rsd-m1	9.69	0.09	9.78	17*	–	RSD over $\mathbb{F}_2$
[CCJ23]-rsd-m2	9.13	0.09	9.22	31*	–	RSD over $\mathbb{F}_2$
[CCJ23]-rsd-s	8.55	0.09	8.64	65*	–	RSD over $\mathbb{F}_2$
ReSolveD-128-Var1	3.99	0.08	4.07	27.3	23.1	RSD over $\mathbb{F}_2$
ReSolveD-128-Var2	3.43	0.08	3.51	158.73	153.11	RSD over $\mathbb{F}_2$

framework to design ZK proofs, we construct a ZK proof on RSD problems in the VOLE-in-the-head framework [BBdSG+23b]. In the VOLE-in-the-head framework, Baum et al. present a non-interactive version of the SoftSpokenOT technique [Roy22] to generate information-theoretic message authentication codes (IT-MACs), and then transform a designated-verifier ZK proof based on IT-MACs (e.g., QuickSilver [YSWW21]) to a publicly-verifiable ZK proof. Our starting point is to prove knowledge of a solution to the RSD problem using a ZK proof based on IT-MACs, and then transform it to a public-coin ZK proof using the VOLE-in-the-head paradigm. Due to the

additive homomorphism and unforgeability of IT-MACs, the equation $\mathbf{H} \cdot \boldsymbol{e} = \boldsymbol{y}$ is easy to prove. The key point is to prove that $\boldsymbol{e}$ is a regular noise with an exact Hamming weight t.

Notice that we can use the approaches implied in previous code-based signature schemes (e.g., [FJR22, AGH+23]) to prove the validity of $\boldsymbol{e}$, but fail to obtain a code-based signature scheme with signature size shorter than the state-of-the-art schemes listed in Table 1. This requires us to exploit an approach that has better compatibility with VOLE-in-the-head. In particular, we refine the sketching technique [BGI16], which is used in verifiable function secret sharing (FSS), to prove the constraint on noise $\boldsymbol{e}$. Additionally, we adopt the recent half-tree technique [GYW+23] to optimize the computation of GGM-based random vector commitments. See Section 1.2 for more details of our technique.

1.2 Technical Overview

We give a high-level overview of the technical route underlying the ReSolveD signature scheme, then we highlight the technical contributions, which include a novel method for validating the noise vector of an RSD problem and the half-tree optimization integrated into the VOLE-in-the-Head framework.

Code-based Signatures from VOLE-in-the-Head. A canonical paradigm in code-based signatures is to first design a public-coin ZK proof for code-based problems and then apply the Fiat-Shamir transform to make it a signature scheme. While there are multiple choices in the design of ZK proofs, the recent VOLE-in-the-Head framework [BBdSG+23b] provides a promising new direction. In particular, within this framework, we can generate IT-MAC relations in a public-coin fashion and then convert designated-verifier ZK (which relies on such relations) into publicly-verifiable ones. Given the rapid development of designated-verifier ZK [WYKW21, DIO21, BMRS21, YSWW21, WYX+21, BBMH+21, DILO22, WYY+22, BBMHS22], this inspires us to design a designated-verifier ZK tailored to the RSD problem and convert it into a code-based signature scheme using VOLE-in-the-Head.

In more detail, the IT-MAC generation of VOLE-in-the-Head begins with the prover committing to a series of GGM trees. For each tree, the prover opens all but one leaf node to the verifier, which allows them to generate a small field VOLE correlation with the punctured index as the global key using the technique in SoftSpokenOT [Roy22]. By applying de-randomization and consistency checking [KOS15, PSS17, OOS17, Roy22], the small field VOLE correlations can be aggregated so that the global key size is large enough to ensure the binding property of IT-MAC. Then the generated IT-MAC correlations are utilized by the subsequent designated-verifier ZK protocol.

One caveat of the above process is that to open the GGM-based vector commitment, the prover needs to know the punctured index, which is also the IT-MAC global key. Nevertheless, once the global key is known by the prover, the binding property fails to hold, and so does the soundness of the designated-verifier ZK upon which it relies. The crucial observation in [CDD+19, BBdSG+23b] is that since the DVZK proof is public-coin, the vector commitment opening can be postponed until the proof has been completed. In this way, even if the prover learns the global key, it can no longer change the proof messages that have already been sent.

Checking the Noise Vector using IT-MAC. We introduce the design of IT-MAC-based designated-verifier ZK for the RSD problem. Our starting point is the QuickSilver protocol [YSWW21] that provides an efficient method to verify the quadratic relations among multiple IT-MAC authenticated triples. We now explain how the validity check of the RSD noise vector can be streamlined into the verification of multiple quadratic relations, a task in which the QuickSilver protocol excels.

We assume without loss of generality that the public matrix $\mathbf{H}$ is in the systematic form, i.e., $\mathbf{H} = [\mathbf{I} \| \mathbf{H_B}]$ and that the witness is split accordingly as $\boldsymbol{e} = [\boldsymbol{e_A} \| \boldsymbol{e_B}]$. Since IT-MACs are linearly homomorphic, the prover can commit to $\boldsymbol{e_B}$ and both parties check the Hamming weight constraint on the "virtual" witness $\boldsymbol{e} = [\boldsymbol{y} - \mathbf{H_B} \cdot \boldsymbol{e_B} \| \boldsymbol{e_B}]$ to implicitly check the linear constraint.

Instead of relying on polynomials [FJR22, AGH+23] or share-conversions [CCJ23] to check the weight constraint, we prove the validity for each noise block by utilizing the sketching technique introduced by Boyle, Gilboa and Ishai [BGI16]. More concretely, when proving the validity of the solution $\boldsymbol{e} = [\boldsymbol{e}_0 \| \boldsymbol{e}_1 \| \dots \| \boldsymbol{e}_{w-1}] \in \mathbb{F}_2^m$ to a RSD problem, we first define w matrices $L^i \in \mathbb{F}_{2^\lambda}^{4 \times m/w}$ each consisting of four rows for all $i \in [0, w)$ (i.e., the linear sketches). The first two rows are uniformly sampled and defined as $\boldsymbol{r}_0^i, \boldsymbol{r}_1^i \leftarrow \mathbb{F}_{2^\lambda}^{m/w}$. The third row is defined as the component-wise product of the first two rows, namely $\boldsymbol{r}_0^i \circ \boldsymbol{r}_1^i$. The last row is an all-1 vector. Then we can compute the sketch $[\![z^i]\!]$ by right-multiplying L^i with $[\![\boldsymbol{e}_i]\!]$ where $[\![\boldsymbol{e}_i]\!]$ is the noise block authenticated using IT-MAC [BDOZ11, NNOB12].

$$[\![z^i]\!] = \begin{bmatrix} [\![z_0^i]\!] \\ [\![z_1^i]\!] \\ [\![z_2^i]\!] \\ [\![z_3^i]\!] \end{bmatrix} = \begin{bmatrix} (\boldsymbol{r}_0^i)^T \\ (\boldsymbol{r}_1^i)^T \\ (\boldsymbol{r}_0^i \circ \boldsymbol{r}_1^i)^T \\ 1 \dots 1 \end{bmatrix} \cdot [\![\boldsymbol{e}_i]\!] \ .$$

Finally, the verification procedure checks that the sketch $z^i = (z_0^i, z_1^i, z_2^i, z_3^i)$ satisfies the condition that $z_0^i \cdot z_1^i - z_2^i = 0$ and $z_3^i - 1 = 0$ for all $i \in [0, w)$. We view the above expression as a degree-2 polynomial in $r_0^i, r_1^i, \dots, r_{m/w}^i$ whose coefficients are determined by $\boldsymbol{e}_i$. Note that if $\boldsymbol{e}_i$ is not a unit vector, then the condition holds with probability less than $2/2^\lambda$ from Schwartz-Zippel Lemma [Sch80, Zip79].

Using the sketching technique, we can convert the validity check of each noise block into the verification of a simple multiplication relation. By running the QuickSilver protocol which shows a way to prove low-degree polynomials with very high efficiency, we can reduce all w linear checks to a single check by a random-linear combination. We note that due to the application of the sketching technique, our protocol still outperforms the polynomial-based protocols of [FJR22, AGH+23] even if we replace the MPC-in-the-Head proof with VOLE-in-the-Head.

Half-tree Optimization. We observe that a large portion of the computational overhead in the VOLE-in-the-Head framework originates from generating the vector commitments. By applying the half-tree technique [GYW+23] we can populate the GGM tree with half the number of calls to symmetric-key ciphers in the random permutation model [GKWY20]. In VOLE-in-the-Head, the number of GGM trees is linearly correlated with the communication overhead. Therefore, by optimizing the computational complexity in GGM tree generation, we can use fewer but deeper trees, opening more possibilities in the communication-computation trade-off.

1.3 Paper Organization

This paper is organized as follows: In Section 2 and Section 3, we introduce notations and definitions for necessary background knowledge on regular syndrome decoding problem, VOLE-in-the-Head paradigm and linear sketching technique. We present our more efficient zero-knowledge proof and signature scheme which yields shorter proof and signature size in Section 4 and Section 5 respectively. To conclude, we provide experimental evaluations of our construction and make comparisons with other state-of-the-art signature schemes in Section 6.

Table 2: Symbols and their meanings in this paper.

Notation	Meaning
$\boldsymbol{a}, [\![\boldsymbol{a}]\!]$	A vector and its authentication
$a_i, \boldsymbol{a}[i,j]$	The i-th coordinate and sub-vector with indices $[i,j]$ of $\boldsymbol{a}$
$\mathbf{U}, \mathbf{U}[i], \mathbf{U}_j$	A matrix, its i-th row and its j-th column
$\mathbf{I}_k$	The $k \times k$ identity matrix
$\mathsf{M}[\boldsymbol{a}], \mathsf{K}[\boldsymbol{a}]$	IT-MAC tag and local key
m	Length of the noise vector in RSD
$m-k$	Length of the syndrome vector in RSD
w, d	Hamming weight of the noise vector in RSD
$\mathbb{F}_2, \mathbb{F}_{2^{\tau'}}, \mathbb{F}_{2^{\tau\tau'}}$	The base field in syndrome decoding and two extension fields
τ, τ'	Degrees of field extensions where $\tau' = O(\log(\lambda))$ and $\tau\tau' \geq \lambda$
$\mathsf{diag}(\boldsymbol{\Delta})$	The diagonal matrix with vector $\boldsymbol{\Delta}$ on its diagonal
$[\boldsymbol{a}\|\boldsymbol{b}], [\mathbf{A}\|\mathbf{B}]$	The concatenation of two vectors or two matrices
$[1 \dots 1]$	The all-one row vector
$\boldsymbol{a} \circ \boldsymbol{b}$	The component-wise multiplication between two vectors
$\mathsf{len}(\boldsymbol{a}), \mathsf{wt}(\boldsymbol{a})$	The length and Hamming weight of a vector $\boldsymbol{a}$

2 Preliminaries

2.1 Notation

We use λ to denote the computational security parameter. We use log to denote logarithms in base 2. We define $[a,b) = \{a, \dots, b-1\}$ and write $[a,b] = \{a, \dots, b\}$ and $[n] = [1,n]$. We write $x \leftarrow S$ to denote sampling x uniformly at random from a finite set S. We use $\{x_i\}_{i \in S}$ to denote the set that consists of all elements with indices in set S. When the context is clear, we abuse the notation and use $\{x_i\}$ to denote such a set.

We use bold lower-case letters like $\boldsymbol{a}$ for column vectors and bold uppercase letters like $\mathbf{A}$ for matrices. We let a_i denote the i-th component of $\boldsymbol{a}$ (with a_0 the first entry) and $\boldsymbol{a}[i,j]$ denote the subvector of $\boldsymbol{a}$ with indices $[i,j]$. Let $\mathsf{len}(\boldsymbol{a})$ be the length of the vector $\boldsymbol{a}$. Let $\mathsf{wt}(\boldsymbol{a})$ be the Hamming weight of the vector $\boldsymbol{a}$ and let $[\boldsymbol{a}\|\boldsymbol{b}]$, $[\mathbf{A}\|\mathbf{B}]$ denote the concatenation of two vectors and matrices, respectively. Let $\mathbf{I}_k$ denote the $k \times k$ identity matrix and $[1 \dots 1]$ denote the all-one row vector where the dimension is implicit in the context. Let $\mathsf{diag}(\boldsymbol{a})$ be the diagonal matrix with the vector $\boldsymbol{a}$ on its main diagonal. We use the notation $\boldsymbol{a} \circ \boldsymbol{b}$ to denote the component-wise multiplication between two vectors $\boldsymbol{a}$ and $\boldsymbol{b}$.

We consider the regular syndrome decoding problem over $\mathbb{F}_2$ in this work. Let $\tau, \tau' \in \mathbb{N}$ and fix two monic irreducible polynomials $f_1(X), f_2(X)$ of degrees τ, τ' respectively. We define $\mathbb{F}_{2^{\tau'}} \cong \mathbb{F}_2[X]/f_2(X)$ and $\mathbb{F}_{2^{\tau\tau'}} \cong \mathbb{F}_{2^{\tau'}}[X]/f_1(X)$. Therefore, we can *pack* τ elements in $\mathbb{F}_{2^{\tau'}}$ or $\tau\tau'$ elements in $\mathbb{F}_2$ into one element in $\mathbb{F}_{2^\lambda}$. We also require $\tau\tau' \geq \lambda$. We list the symbols and their definitions of this paper in Table 2.

2.2 Hash Functions

In our protocol, we utilize universal hash functions and circular correlation robust hash functions. Following prior works [GKWY20], we define the security requirements in Definition 1 and Definition 2.

Definition 1. *A linear ϵ-almost universal family of hashes is a family of matrices $\mathcal{H} \subseteq \mathbb{F}_2^{r\times(n+h)}$ such that for any nonzero $\boldsymbol{v} \in \mathbb{F}_2^{n+h}$, $\Pr_{\mathsf{H}\leftarrow\mathcal{H}}[\mathsf{H}\cdot\boldsymbol{v}=0] \leq \epsilon$. A matrix H is n-hiding if the distribution $\mathsf{H}\cdot\boldsymbol{v}$ is independent of $\boldsymbol{v}[0,n)$ for a uniformly random $\boldsymbol{v} \leftarrow \mathbb{F}_2^{n+h}$. The hash family $\mathcal{H}$ is n-hiding if every hash function in this family is n-hiding.*

Definition 2. *Let $\mathsf{H} : \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a function. For $\Gamma \in \{0,1\}^\lambda$, define $\mathcal{O}_\Delta^{\mathsf{ccr}}(x,b) = \mathsf{H}(x \oplus \Gamma) \oplus b\cdot\Gamma$. We don't allow the distinguisher to query the same x with both 0 and 1 to avoid the trivial attack. For a distinguisher $\mathcal{D}$, we define the following advantage*

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

where $\mathcal{F}_{\lambda+1,\lambda}$ denotes the set of all functions mapping $(\lambda+1)$-bit inputs to λ-bit outputs. H is (t,q,ϵ)-circular correlation robust if for all $\mathcal{D}$ running in time t and making at most q queries to the oracle we have $\mathsf{Adv}_{\mathsf{H}}^{\mathsf{ccr}} \leq \epsilon$.

2.3 Regular Syndrome Decoding

We recall the regular syndrome decoding problem (RSD) where the noise vector is the concatenation of several *unit vectors*. We inherit the notations from [FJR22] (named d-split syndrome decoding in that paper).

Definition 3. *Let m, k, w, d be positive integers such that $m > k$, $m > w$ and $d = w$. The regular noise syndrome decoding problem with parameters (m,k,w,d) is the following problem: Let $\mathbf{H}, \boldsymbol{e}$ and $\boldsymbol{y}$ be such that:*

1. *$\mathbf{H}$ is uniformly sampled from $\mathbb{F}_2^{(m-k)\times m}$,*
2. *$\boldsymbol{e}$ is uniformly sampled from $\{[\boldsymbol{e}_0\|...\|\boldsymbol{e}_{w-1}] : \forall i \in [0,w), \boldsymbol{e}_i \in \mathbb{F}_2^{\frac{m}{w}}, \|\boldsymbol{e}_i\|_0 = 1\}$,*
3. *$\boldsymbol{y}$ is defined as $\boldsymbol{y} := \mathbf{H}\cdot\boldsymbol{e}$. From $(\mathbf{H},\boldsymbol{y})$, find $\boldsymbol{e}$.*

Systematic Form of RSD. Following previous works [FJR22, CCJ23] we assume without loss of generality that the matrix $\mathbf{H}$ is in the systematic form, i.e., $\mathbf{H} = [\mathbf{I}_{m-k}\|\mathbf{H_B}]$. Therefore, in the zero-knowledge protocol, the solution $\boldsymbol{e} = [\boldsymbol{e_A}\|\boldsymbol{e_B}]$ can be compressed into a smaller one $\boldsymbol{e_B}$ since the complete witness can be linearly expressed as $\boldsymbol{e} = [\boldsymbol{y} - \mathbf{H_B}\cdot\boldsymbol{e_B}\|\boldsymbol{e_B}]$.

The benefit of this optimization is two-fold. Firstly, by compressing the witness we can reduce the communication complexity of the ZK protocol, and therefore the signature size. Secondly, the linear expression implicitly enforces the constraint that $\mathbf{H}\cdot\boldsymbol{e} = \boldsymbol{y}$, and thus we only need to check the Hamming weight constraint on the "virtual" vector $\boldsymbol{e} = [\boldsymbol{y} - \mathbf{H_B}\cdot\boldsymbol{e_B}\|\boldsymbol{e_B}]$ in the ZK protocol.

The Hardness of RSD. A number of works studied the hardness of regular syndrome decoding under different parameter regimes [HOSS18, LWYY22]. In particular, some recent works utilize the regular noise structure into the state-of-the-art cryptanalysis algorithms of syndrome decoding [CCJ23, BØ23, ES23]. To the best of our knowledge, the chosen parameters for our signature scheme lie in a region where the exact relationship between the hardness of RSD and SD remains unclear [ES23]. In Table 3, we choose parameters such that the RSD solution is unique ($\binom{\frac{m}{w}}{w} < 2^{m-k}$) while the same parameter would lead to multiple solutions if we drop the regularity constraint ($\binom{m}{w} > 2^{m-k}$). In this region, the regular structure of noises does not lead to significantly better attacks [CCJ23, BØ23, ES23].

Functionality $\mathcal{F}_{\mathsf{DVZK}}$

This functionality runs with a prover $\mathcal{P}$ and a verifier $\mathcal{V}$, and operates as follows:

- Upon receiving $(\mathsf{dvzk}, sid, \ell, \{[\![x_i]\!], [\![y_i]\!], [\![z_i]\!]\}_{i\in[0,\ell)})$ from $\mathcal{P}$ and $\mathcal{V}$ if there exists some $i \in [0, \ell)$ such that one of $[\![x_i]\!], [\![y_i]\!], [\![z_i]\!]$ is not valid, output (sid, false) to $\mathcal{V}$ and abort.
- Check that $z_i = x_i \cdot y_i$ for all $i \in [0, \ell)$. If the check passes, then output (sid, true) to $\mathcal{V}$, else output (sid, false) to $\mathcal{V}$.

Figure 1: Functionality for DVZK proofs of authenticated multiplication triples.

2.4 Information-Theoretic Message Authentication Codes

We use information-theoretic message authentication codes (IT-MACs) [BDOZ11, NNOB12] over $\mathbb{F}_{2^\lambda}$. Specifically, let $\Delta \in \mathbb{F}_{2^\lambda}$ be a *global key*. We use $[\![x]\!] = (\mathsf{K}[x], \mathsf{M}[x], x)$ to denote that an element $x \in \mathbb{F}$ (where $\mathbb{F} \in \{\mathbb{F}_2, \mathbb{F}_{2^\lambda}\}$ known by one party can be authenticated by the other party who holds Δ and a *local key* $\mathsf{K}[x] \in \mathbb{F}_{2^\lambda}$, where a MAC tag $\mathsf{M}[x] = \mathsf{K}[x] + x \cdot \Delta$ is given to the party holding x. For a vector $\boldsymbol{x} \in \mathbb{F}_{2^\lambda}^\ell$, we denote by $[\![\boldsymbol{x}]\!] = ([\![x_0]\!], ..., [\![x_{\ell-1}]\!])$ a vector of authenticated values. For a constant value $c \in \mathbb{F}_{2^\lambda}$, it is easy to define $[\![c]\!] = (-c \cdot \Delta, 0, c)$. It is well known that IT-MACs are additively homomorphic. That is, given public coefficients $c_0, c_1, \ldots, c_\ell \in \mathbb{F}_{2^\lambda}$, two parties can *locally* compute $[\![y]\!] := \sum_{i=0}^{\ell-1} c_i \cdot [\![x_i]\!] + c_\ell$.

The IT-MAC authenticated value $[\![x]\!]$ can be opened by revealing x and $\mathsf{M}[x]$ and the validity can be enforced by checking that $\mathsf{M}[x] = \mathsf{K}[x] + x \cdot \Delta$. The security holds since opening $[\![x]\!]$ to any other value $(\mathsf{K}[x], \mathsf{M}[x'], x')$ is equivalent to guessing the global key since $\Delta = (\mathsf{M}[x] - \mathsf{M}[x']) \cdot (x - x')^{-1}$.

We can open multiple values $[\![x_0]\!], ..., [\![x_{\ell-1}]\!]$ in a batch by sending one MAC tag as follows. The sender first reveals $x'_0, ..., x'_{\ell-1}$, then using the additive homomorphism of IT-MAC, both parties can define $[\![y_i]\!] = [\![x_i]\!] - x'_i$ for $i \in [0, \ell)$. Now it suffices to check that $\forall i \in [0, \ell), y_i = 0$.

For task of checking multiple zero values $[\![y_0]\!], ..., [\![y_{\ell-1}]\!]$, we can save communication by opening a random linear combination $\chi_0 \cdot [\![y_0]\!] + ... + \chi_{\ell-1} \cdot [\![y_{\ell-1}]\!]$. In particular, the sender can only send $\sum_{i=0}^{\ell-1} \chi_i \cdot \mathsf{M}[y_i]$ since $\sum_{i=0}^{\ell-1} \chi_i \cdot y_i = 0$ for uniformly random $\chi_i \leftarrow \mathbb{F}_{2^\lambda}$.

2.5 Designated-Verifier Zero-Knowledge for Quadratic Relations

Based on IT-MACs, a family of streamable designated-verifier zero-knowledge (DVZK) proofs with fast prover time and a small memory footprint has been proposed [WYKW21, DIO21, BMRS21, YSWW21, WYX+21, BBMH+21, DILO22, WYY+22, BBMHS22]. While these DVZK proofs can prove arbitrary circuits, we only need them to prove a simple multiplication relation for our purpose. Specifically, given a set of authenticated triples $\{([\![x_i]\!], [\![y_i]\!], [\![z_i]\!])\}_{i\in[0,\ell)}$ over $\mathbb{F}_{2^\lambda}$, these DVZK protocols can enable a prover $\mathcal{P}$ to convince a verifier $\mathcal{V}$ that $z_i = x_i \cdot y_i$ for all $i \in [0, \ell)$. This is modeled by an ideal functionality shown in Figure 1. In this functionality, an authenticated value $[\![x]\!]$ is input by two parties $\mathcal{P}$ and $\mathcal{V}$, meaning that $\mathcal{P}$ inputs (x, M) and $\mathcal{V}$ inputs (K, Δ). We say that $[\![x]\!]$ is valid, if $\mathsf{M} = \mathsf{K} + x \cdot \Delta$.

QuickSilver. We use the QuickSilver protocol [YSWW21] to instantiate $\mathcal{F}_{\mathsf{DVZK}}$. The benefit of using this protocol is two-fold. Firstly, the protocol is public-coin in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model, making it compatible with the VOLE-in-the-Head technique to be explained next. Secondly, the QuickSilver protocol excels at proving many quadratic relations as required in proving the Hamming weight constraint in the RSD problem, which only requires sending 2λ bits in total.

We briefly sketch how to prove multiple quadratic constraints in QuickSilver. Suppose the prover wants to prove $z_i = x_i \cdot y_i$ for $i \in [0, \ell)$, the verifier samples random challenges $\chi_0, ..., \chi_{\ell-1} \in \mathbb{F}$ and evaluates the following value using the IT-MAC relation $\mathsf{M}[x] = \mathsf{K}[x] + x \cdot \Delta$.

$$\sum_{i\in[0,\ell)} \chi_i \cdot (\mathsf{K}[x_i] \cdot \mathsf{K}[y_i] + \Delta \cdot \mathsf{K}[z_i]) = \sum_i \chi_i \cdot (x_i y_i - z_i) \cdot \Delta^2 + \sum_i \chi_i \cdot (-x_i \mathsf{M}[y_i] - y_i \mathsf{M}[x_i] + \mathsf{M}[z_i]) \cdot \Delta + \sum_i \chi_i \cdot \mathsf{M}[x_i]\mathsf{M}[y_i] \ .$$

If the quadratic relations hold then this value should be a linear function of Δ. To prove this, $\mathcal{P}$ simply sends the masked coefficients c_1, c_0 of that function to $\mathcal{V}$, who checks that $c_1 \cdot \Delta + c_0$ equals the masked left-hand side.

2.6 The Zero-Knowledge Functionality

We recall the definition of the ideal zero-knowledge functionality in Figure 2. Looking ahead, we will construct a public-coin designated-verifier zero-knowledge protocol that realizes the functionality $\mathcal{F}_{\mathsf{RSD-ZK}}$ in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model, which can then be transformed into a publicly-verifiable zero-knowledge protocol using the techniques in [BBdSG+23b].

Functionality $\mathcal{F}_{\mathsf{RSD-ZK}}$

Both parties $\mathcal{P}$ and $\mathcal{V}$ have access to the RSD instance $\boldsymbol{y} = \mathbf{H} \cdot \boldsymbol{e}$ for $\boldsymbol{y} \in \mathbb{F}_2^{m-k}$ and $\mathbf{H} \in \mathbb{F}_2^{(m-k)\times m}$. We assume the matrix $\mathbf{H} = [\mathbf{I}_{m-k} \| \mathbf{H_B}]$ is in the systematic form.

- **Input.** Upon receiving $(\texttt{input}, \boldsymbol{e})$ from $\mathcal{P}$ and $(\texttt{input})$ from $\mathcal{V}$, the functionality stores the input $\boldsymbol{e}$. Let $\boldsymbol{e} = [\boldsymbol{e}_0 \| ... \| \boldsymbol{e}_{w-1}]$, where $\mathsf{len}(\boldsymbol{e}_i) = m/w$ for $i \in [0, w)$.
- **Prove.** Upon receiving $(\texttt{prove})$ from both parties, the functionality checks that $\boldsymbol{e}$ satisfies the following two constraints:
  - Linear Constraint: It holds that $\mathbf{H} \cdot \boldsymbol{e} = \boldsymbol{y}$.
  - Hamming Weight Constraint: It holds that $\mathsf{wt}(\boldsymbol{e}_i) = 1$ for $i \in [0, w)$.

  If both constraints are satisfied, the functionality sends true to $\mathcal{V}$. Otherwise, it sends false.

Figure 2: The zero-knowledge functionality for regular syndrome decoding

3 VOLE-in-the-Head and Linear Sketching

In this section, we recapture the VOLE-in-the-head technique in [BBdSG+23b] as well as the sketching technique of [BGI16] that form the basis of our signature scheme in Section 5.

3.1 VOLE-in-the-Head

VOLE-in-the-Head is a technique proposed by Baum et al. [BBdSG+23b] which allows transforming the public-coin designated-verifier zero-knowledge protocols in the VOLE-hybrid model into the publicly verifiable counterparts[1]. At the core of this technique is the observation that GGM-style

[1]This technique somewhat resembles the classical MPC-in-the-head technique.

vector commitment can realize an all-but-one random oblivious transfer functionality, which can then be transformed into a VOLE protocol using the technique in SoftSpokenOT [Roy22]. One caveat is that to facilitate the simulation of OT from commitment, the verifier has to send its choice in the clear; Nevertheless, this suffices for a public-coin protocol since the verifier's action is merely sending public coins and the OT's output can be delayed to the very end of the protocol.

GGM-style Vector Commitment. Given a n-level GGM tree, let r_j^i denotes the j-th node on the i-th level where $0 \leq i < n$ and $0 \leq j < 2^i$. It's well-known that if the root node is uniformly random and the tree is generated as $r_{2j}^{i+1} \| r_{2j+1}^{i+1} := \mathsf{PRG}(r_j^i)$ for some length-doubling PRG then the leaf nodes are pseudorandom. Moreover, for each leaf node, we can derive a random message and an authenticator. Then all messages can be committed by hashing the authenticators while all but one of them can be opened by presenting the sibling nodes on the punctured path and the authenticator of the punctured message. We model this vector commitment as an ideal functionality $\mathcal{F}_{\mathsf{VC}}$ in Figure 3.

Functionality $\mathcal{F}_{\mathsf{VC}}$

Let $\tau' \in \mathbb{N}$, $\tau' \geq 1$ and $N = 2^{\tau'}$. The functionality is run between two parties $\mathcal{P}$ and $\mathcal{V}$ which are possibly corrupted by the adversary $\mathcal{A}$. We have the following commands.

- Upon receiving (`commit`) from $\mathcal{P}$ and $\mathcal{V}$, the functionality samples $m_i \leftarrow \{0,1\}^\lambda$ for $i \in [0, N)$. If $\mathcal{P}$ is corrupted then it receives $\{m_i\}_{i \in [0,N)}$ from the adversary and locally records the values. Then it sends (`done`) to both parties.
- Upon receiving (`get`, α) from $\mathcal{P}$ and $\mathcal{V}$, the functionality sends $\{m_i\}_{i \in [0,N), i \neq \alpha}$ to $\mathcal{V}$.

Figure 3: The ideal vector commitment scheme supporting all-but-one opening.

SoftSpokenOT. Let $\mathsf{PRG} : \mathbb{F}_2^\lambda \to \mathbb{F}_2^n$ be a pseudorandom generator. SoftSpokenOT [Roy22] utilizes the fact that the all-but-one OT correlation is equivalent to the subfield VOLE correlation over the polynomial-sized extension field $\mathbb{F}_{2^{\tau'}}$. Let $\Delta \in \mathbb{F}_{2^{\tau'}}$ be the OT index. The key observation (which is implicit in the classical IKNP protocol) is that by defining $\boldsymbol{u}' := \sum_{i \in [0,2^{\tau'})} \mathsf{PRG}(m_i)$, $\boldsymbol{v} := \sum_{i \in [0,2^{\tau'})} i \cdot \mathsf{PRG}(m_i)$, and $\boldsymbol{w}' := \sum_{i \in [0,2^\ell)} (i + \Delta) \cdot \mathsf{PRG}(m_i)$, the sender and receiver can locally compute the respective values and the transformation from OT to subfield VOLE can be done non-interactively. Notice that in the expression of $\boldsymbol{w}'$, the value $\mathsf{PRG}(m_\Delta)$ which is unknown to the receiver is multiplied by 0 and the receiver can efficiently compute $\boldsymbol{w}'$. Therefore, $\boldsymbol{w}' = \boldsymbol{v} + \boldsymbol{u}' \cdot \Delta$.

Since the field $\mathbb{F}_{2^{\tau'}}$ need to be enumerated, we require that $\tau' = O(\log \lambda)$ (i.e. small-field VOLE). Nevertheless, the VOLE global key needs to contain enough entropy to ensure soundness. Therefore, we need to repeat the base protocol $\lceil \frac{\lambda}{\tau'} \rceil$ times and apply a consistency checking protocol to ensure that the same vector $\boldsymbol{u}$ is used in all small-field VOLE instances (so that the global keys can be concatenated.)

In particular, the sender and receiver would run the above small-field VOLE protocol for $\tau := \lceil \frac{\lambda}{\tau'} \rceil$ times, acquiring $[\![\boldsymbol{u}'_0]\!], ..., [\![\boldsymbol{u}'_{\tau-1}]\!]$, where $\boldsymbol{u}'_i \in \mathbb{F}_2^{n+h}$. Then by viewing each row of the concatenated matrix $\mathbf{U}' := [\boldsymbol{u}'_0 \| ... \| \boldsymbol{u}'_{\tau-1}]$ as a noisy codeword of the length-τ repetition code, as the sender sends the syndrome $\mathbf{C}$ of all the codewords to the receiver. Then the sender corrects the matrix $\mathbf{U}'$ into a structured matrix $\mathbf{U} := [1 \, ... \, 1] \cdot \boldsymbol{u}$ where each row is a repetition codeword while the receiver sets $\mathbf{W} = \mathbf{W}' + [0 \| \mathbf{C}] \cdot \mathsf{diag}(\boldsymbol{\Delta})$ where $\boldsymbol{\Delta}$ denotes the concatenation of all small-field

VOLE global keys. Notice that with the matrix $\mathbf{U}$ being structured, we can transform each row of $\mathbf{W}, \mathbf{V}$ as well as $\boldsymbol{\Delta}$ as elements in the extension field $\mathbb{F}_{2^\lambda}$, which gives the IT-MAC format.

Finally, we need to check that $\mathbf{U}$ is indeed structured. We do this by sacrificing the last h rows of $\mathbf{U}$. In particular, the sender sends $\tilde{\boldsymbol{u}} = \mathsf{H}^{\mathsf{UHF}} \cdot \boldsymbol{u}$ and $\tilde{\mathbf{V}} = \mathsf{H}^{\mathsf{UHF}} \cdot \mathbf{V}$ for some linear universal hash function $\mathsf{H}^{\mathsf{UHF}} \in \mathbb{F}_2^{r \times (n+h)}$ while the receiver checks that $\tilde{\mathbf{V}} + \tilde{\boldsymbol{u}} \cdot [1 \dots 1] \cdot \mathsf{diag}(\boldsymbol{\Delta}) = \mathsf{H}^{\mathsf{UHF}} \cdot \mathbf{W}$.

VOLE-in-the-Head. We put together all the pieces and explain the technique in [BBdSG+23b]. Recall that our goal is to transform a designated-verifier zero-knowledge protocol in the VOLE-hybrid model into a publicly verifiable one. We additionally require that the DVZK protocol be public-coin. The transformation proceeds as follows. We state the protocol in the interactive setting but the interaction can be removed using Fiat-Shamir [FS87].

1. The prover locally runs the SoftSpokenOT protocol, instantiating the all-but-one random OT with vector commitment. In particular, the prover generates the GGM trees and sends the commitments to the verifier. Then the prover simulates the SoftSpokenOT protocol, sending the correction syndrome and checking information to the verifier.

2. With the IT-MAC correlations from previous step, the parties simulate the zero-knowledge protocol using the previous subfield VOLE correlations.

3. When all interactions of the zero-knowledge protocol are completed, the verifier simply sends the VOLE global key Δ to the prover, who then replies with the corresponding vector decommitment. The verifier then checks that

   (a) the vector commitment openings are correct;

   (b) the consistency checks inside SoftSpokenOT are correct;

   (c) the zero-knowledge verification passes.

   If all checks pass then the verifier accepts. Otherwise, it rejects the proof.

Intuitively, since the inner ZK protocol in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model is public-coin, the parties can still simulate the protocol before sampling the global key Δ, and since the proof information is already sent in step 2, revealing the global key in step 3 does not grant the prover any advantage. In [BBdSG+23b], this intuition is characterized by an ideal functionality $\mathcal{F}_{\mathsf{sVOLE}}^{p,q,S_\Delta,\mathcal{C},\ell,\mathcal{L}}$ where the receiver's outputs are revealed after the prover commits to its inputs. In this work, we only consider a special case of it, namely we only consider using repetition code and fixing the set S_Δ to be the entire field $\mathbb{F}_{2^{\tau'}}$. We recall the functionality in Figure 4.

3.2 The Linear Sketching Technique

To verify the Hamming weight constraint, we use the linear sketching technique of Boyle et al. [BGI16]. For general field $\mathbb{F}$, given an IT-MAC authenticated vector $[\![\boldsymbol{u}]\!]$ where $\boldsymbol{u} \in \mathbb{F}^n$, we can easily check that $\|\boldsymbol{u}\|_0 = 1$. We first sample two public random vectors $\boldsymbol{r}_0, \boldsymbol{r}_1 \in \mathbb{F}^n$ and define z_0, z_1, z_2, z_3 as follows.

$$\boldsymbol{z} = \begin{bmatrix} z_0 \\ z_1 \\ z_2 \\ z_3 \end{bmatrix} = \begin{bmatrix} \boldsymbol{r}_0^T \\ \boldsymbol{r}_1^T \\ (\boldsymbol{r}_0 \circ \boldsymbol{r}_1)^T \\ 1 \dots 1 \end{bmatrix} \cdot \boldsymbol{u} \ .$$

Functionality $\mathcal{F}_{\mathsf{sVOLE}}$

The functionality is parameterized by the base field $\mathbb{F}_2$ and its extension $\mathbb{F}_{2^{\tau'}}$. We also define an integer n as the number of random sVOLE correlations to produce and τ as the repetition parameter such that $\tau\tau' \geq \lambda$.

Upon receiving (`init`) from $\mathcal{P}$ and $\mathcal{V}$, the functionality does the following.

- Sample $\boldsymbol{u} \leftarrow \mathbb{F}_2^n$, $\mathbf{V} \leftarrow \mathbb{F}_{2^{\tau'}}^{n \times \tau}$ and $\boldsymbol{\Delta} \leftarrow \mathbb{F}_{2^{\tau'}}^{\tau}$. Let $\mathbf{W} = \mathbf{V} + \boldsymbol{u} \cdot [1 \dots 1] \cdot \mathsf{diag}(\boldsymbol{\Delta})$.
  - If $\mathcal{P}$ is corrupted, then receive $\boldsymbol{u}, \mathbf{V}$ from the adversary $\mathcal{A}$ and recompute $\mathbf{W}$.
  - If $\mathcal{V}$ is corrupted, then receive $\boldsymbol{\Delta}, \mathbf{W}$ from the adversary $\mathcal{A}$ and recompute $\mathbf{V} = \mathbf{W} - \boldsymbol{u} \cdot [1 \dots 1] \cdot \mathsf{diag}(\boldsymbol{\Delta})$.
- Send $(\boldsymbol{u}, \mathbf{V})$ to $\mathcal{P}$.
- If $\mathcal{P}$ is corrupted, then receive a leakage query L from $\mathcal{A}$.

Upon receiving (`get`) from $\mathcal{P}$ and $\mathcal{V}$, the functionality does the following.

- If $\boldsymbol{\Delta} \notin L$, then send (`check-failed`) to $\mathcal{V}$ and abort.
- Otherwise, send $(\boldsymbol{\Delta}, \mathbf{W})$ to $\mathcal{V}$.

Figure 4: The subspace VOLE functionality.

Here $\circ$ denotes the component-wise product between two vectors. Finally, we check that $z_0 \cdot z_1 = z_2$ and that $z_3 = 1$. The first check ensures that $\|\boldsymbol{u}\|_0 \leq 1$. Conditioned on passing the first check, the second check ensures that $\boldsymbol{u}$ is a unit vector. The second check is straightforward and we will elaborate on the intuition of the first check.

When viewing $(\boldsymbol{r}_0 \circ \boldsymbol{r}_1)^T \cdot \boldsymbol{u} - \left(\boldsymbol{r}_0^T \cdot \boldsymbol{u}\right) \cdot \left(\boldsymbol{r}_1^T \cdot \boldsymbol{u}\right)$ as a multivariate polynomial over $\boldsymbol{r}_0, \boldsymbol{r}_1$, we have that if $\|\boldsymbol{u}\|_0 > 1$ then the polynomial is non-zero and has degree of two. Therefore, with the Schwartz-Zippel lemma [Sch80, Zip79], we can show that the equation $z_2 = z_0 \cdot z_1$ holds except with probability $\frac{2}{|\mathbb{F}|}$ over the choices of $r_0, ..., r_{n-1}$. Formally, we have the following lemma by Boyle et al. [BGI16].

Lemma 1. *Let $\mathbb{F}$ be any finite field. Suppose $\boldsymbol{u} \in \mathbb{F}^n$ is not a unit vector then we have the probability*

$$\Pr[L \leftarrow \mathcal{L}(\mathbb{F}, n), \boldsymbol{z} = L \cdot \boldsymbol{u} : z_0 \cdot z_1 = z_2 \wedge z_3 = 1] \leq \frac{2}{|\mathbb{F}|} \ ,$$

where the distribution $\mathcal{L}(\mathbb{F}, n)$ is defined by sampling $\boldsymbol{r}_0, \boldsymbol{r}_1 \leftarrow \mathbb{F}^n$ and returning

$$L = \begin{bmatrix} \boldsymbol{r}_0^T \\ \boldsymbol{r}_1^T \\ (\boldsymbol{r}_0 \circ \boldsymbol{r}_1)^T \\ 1 \dots 1 \end{bmatrix} .$$

Notice that since IT-MAC is linear homomorphic, we can get the authentication of $\boldsymbol{z}$ by evaluating $[\![\boldsymbol{z}]\!] = L \cdot [\![\boldsymbol{u}]\!]$. Then, we can use the IT-MAC opening operation to check that $z_3 = 1$ and use QuickSilver to prove that $z_0 \cdot z_1 = z_2$.

In our protocol, we perform the checking on $\boldsymbol{u} \in \mathbb{F}_2$ over the extension field $\mathbb{F}_{2^\lambda}$ to get negligible soundness error. For RSD over larger fields, we can adapt the above method to prove that the non-zero element is equal to an arbitrary value in the field. Nevertheless, we focus on RSD over $\mathbb{F}_2$ in this work and using the above sketching technique is sufficient.

4 Designated-Verifier ZK from Linear Sketching

In this section, we present an efficient zero-knowledge proof for the RSD problem in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model and give a security proof for its soundness and zero-knowledge property.

Protocol $\Pi_{\mathsf{RSD-DVZK}}$

Both parties $\mathcal{P}$ and $\mathcal{V}$ have access to the RSD instance $(\mathbf{H}, \boldsymbol{y})$ where $\mathbf{H} \in \mathbb{F}_2^{(m-k)\times m}$ and $\boldsymbol{y} = \mathbf{H} \cdot \boldsymbol{e}$. $\mathcal{P}$ has the witness $\boldsymbol{e} = [\boldsymbol{e}_0 \| ... \| \boldsymbol{e}_{w-1}]$ where $\|\boldsymbol{e}_i\|_0 = 1$ for $i \in [0, w)$. We assume the matrix $\mathbf{H} = [\mathbf{I}_{m-k} \| \mathbf{H}_\mathbf{B}]$ is in the systematic form and the vector $\boldsymbol{e}$ is split into $\boldsymbol{e}_\mathbf{A}$ and $\boldsymbol{e}_\mathbf{B}$. We assume without loss of generality that $\tau\tau' = \lambda$.

Input: The prover commits to the witness as follows.

1. Both parties call $\mathcal{F}_{\mathsf{sVOLE}}$ on input (`init`) and $\mathcal{P}$ gets $\boldsymbol{u} \in \mathbb{F}_2^n$ and $\mathbf{V} \in \mathbb{F}_{2^{\tau'}}^{n\times\tau}$. Let $n = k + \lambda$. For $i \in [0, k)$ the prover defines $\mathbf{V}_i \hookrightarrow \mathsf{M}[u_i]$. The prover lifts the last λ coordinates of $\boldsymbol{u}$ into a mask $u^{\mathsf{QS}} \in \mathbb{F}_{2^\lambda}$ and we define its IT-MAC tag $\mathsf{M}[u^{\mathsf{QS}}]$ similarly by lifting the respective rows in $\mathbf{V}$. Finally, the prover sends $\boldsymbol{d} = \boldsymbol{e}_\mathbf{B} - \boldsymbol{u}[0, k)$ to the verifier.

Prove: The parties check that the committed witness is valid as follows.

2. The verifier samples the sketch functions $L^{(i)}$ according to Lemma 1 for $i \in [0, w)$ and random challenges $\boldsymbol{\chi}^{\mathsf{QS}}, \boldsymbol{\chi}^{\mathsf{open}} \leftarrow \mathbb{F}_{2^\lambda}^w$ and sends them to the prover.

3. The prover defines $\mathsf{M}[\boldsymbol{e}_\mathbf{B}]$ as the first k coordinates of $\mathsf{M}[\boldsymbol{u}]$ and reconstructs $\mathsf{M}[\boldsymbol{e}]$ using the linear relation $\boldsymbol{e} = [\boldsymbol{y} - \mathbf{H}_\mathbf{B} \cdot \boldsymbol{e}_\mathbf{B} \| \boldsymbol{e}_\mathbf{B}]$. Let $\boldsymbol{e} = [\boldsymbol{e}_0 \| ... \| \boldsymbol{e}_{w-1}]$. Using the linear homomorphism of IT-MAC, we define the authentication of $\boldsymbol{z}^{(i)} = L^{(i)} \cdot \boldsymbol{e}_i$ for $i \in [0, w)$.

   Then the prover uses QuickSilver to prove that $z_0^{(i)} \cdot z_1^{(i)} = z_2^{(i)}$ for $i \in [0, w)$. In particular, it computes $m_0^{\mathsf{QS}} = \mathsf{M}[u^{\mathsf{QS}}] + \sum_{i\in[0,w)} \chi_i^{\mathsf{QS}} \cdot \mathsf{M}[z_0^{(i)}] \cdot \mathsf{M}[z_1^{(i)}]$ and $m_1^{\mathsf{QS}} = u^{\mathsf{QS}} + \sum_{i\in[0,w)} \chi_i^{\mathsf{QS}} \cdot (z_0^{(i)} \cdot \mathsf{M}[z_1^{(i)}] + z_1^{(i)} \cdot \mathsf{M}[z_0^{(i)}] + \mathsf{M}[z_2^{(i)}])$. The prover also defines $m_0^{\mathsf{open}} = \sum_{i\in[0,w)} \chi_i^{\mathsf{open}} \cdot \mathsf{M}[z_3^{(i)}]$. The prover sends $m_0^{\mathsf{QS}}, m_1^{\mathsf{QS}}, m_0^{\mathsf{open}}$ to the verifier.

4. Both parties call the functionality $\mathcal{F}_{\mathsf{sVOLE}}$ on input (`get`) and the verifier gets $\boldsymbol{\Delta} = (\Delta_0, ..., \Delta_{\tau-1}) \in \mathbb{F}_{2^{\tau'}}^\tau$ and $\mathbf{W} \in \mathbb{F}_{2^{\tau'}}^{n\times\tau}$. The verifier defines $\boldsymbol{\Delta} \hookrightarrow \Delta$ and $\mathbf{W}_i \hookrightarrow \mathsf{K}[u_i]$ for $i \in [0, k)$ as well as the IT-MAC local key $\mathsf{K}[u^{\mathsf{QS}}]$ using the last λ rows of $\mathbf{W}$. Then it defines $\mathsf{K}[\boldsymbol{e}_\mathbf{B}] = \mathsf{K}[\boldsymbol{u}[0, k)] + \boldsymbol{d} \cdot \Delta$ and reconstructs $\mathsf{K}[\boldsymbol{e}]$ using the linear relation $\boldsymbol{e} = [\boldsymbol{y} - \mathbf{H}_\mathbf{B} \cdot \boldsymbol{e}_\mathbf{B} \| \boldsymbol{e}_\mathbf{B}]$. It also defines the IT-MAC local keys of $\boldsymbol{z}^{(i)}$ for $i \in [0, w)$. The verifier accepts if the following relation holds and rejects otherwise.

$$m_0^{\mathsf{QS}} + m_1^{\mathsf{QS}} \cdot \Delta = \mathsf{K}[u^{\mathsf{QS}}] + \sum_{i\in[0,w)} \chi_i^{\mathsf{QS}} \cdot (\mathsf{K}[z_0^{(i)}] \cdot \mathsf{K}[z_1^{(i)}] + \Delta \cdot \mathsf{K}[z_2^{(i)}]) \text{ and}$$

$$m_0^{\mathsf{open}} + (\sum_{i\in[0,w)} \chi_i^{\mathsf{open}}) \cdot \Delta = \sum_{i\in[0,w)} \chi_i^{\mathsf{open}} \cdot \mathsf{K}[z_3^{(i)}] \ .$$

Figure 5: The ZK protocol for syndrome decoding based on linear sketch in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model.

4.1 Protocol Description

Since we may view the matrix $\mathbf{H}$ in its systematic form and therefore implicitly enforce the linear constraint, we can turn our focus to proving the Hamming weight constraint. Using the linear sketching technique [BGI16] we can check that the segment of the witness vector has a Hamming weight of exactly 1 by verifying a quadratic relation and performing an IT-MAC opening. Using the QuickSilver protocol, we can prove all w quadratic relations corresponding to the entire witness

vector in a batch using a random linear combination, with essentially the same communication cost as proving one quadratic relation. The cost of w openings can also be reduced using another random linear combination. We describe the protocol in detail in Figure 5 and prove its security in the next subsection.

4.2 Security Proof

We prove that the protocol $\Pi_{\mathsf{RSD-DVZK}}$ is an honest verifier zero-knowledge protocol for regular syndrome decoding in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model in Theorem 1. Our proof is a straightforward extension of the proof in [BBdSG+23b]. The only difference is that we use the linear sketch technique from [BGI16] to check the validity of the witness vector.

Theorem 1. *The protocol $\Pi_{\mathsf{RSD-DVZK}}$ realizes the functionality $\mathcal{F}_{\mathsf{RSD-ZK}}$ in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model. The security holds against a malicious prover or a semi-honest verifier and the soundness error in the former case is bounded by $\frac{7}{2^\lambda}$.*

Proof. Correctness of the proof follows by definition. In the following, we construct simulators for the malicious prover and verifier cases to argue soundness and zero-knowledge properties respectively.

Malicious Prover. The simulator $\mathcal{S}_{\mathcal{P}}$ is constructed as follows.

1. $\mathcal{S}_{\mathcal{P}}$ simulates the (`init`) command of the functionality $\mathcal{F}_{\mathsf{sVOLE}}$ by receiving the $\boldsymbol{u}, \mathbf{V}$ values from $\mathcal{A}$. It also receives the difference vector $\boldsymbol{d}$ and recovers the witness $\boldsymbol{e}_{\mathbf{B}} = \boldsymbol{d} + \boldsymbol{u}[0, k)$. Let $\boldsymbol{e} = [\boldsymbol{y} - \mathbf{H}_{\mathbf{B}} \cdot \boldsymbol{e}_{\mathbf{B}} \| \boldsymbol{e}_{\mathbf{B}}]$. $\mathcal{S}_{\mathcal{P}}$ sends message (`input`, $\boldsymbol{e}$) to the functionality $\mathcal{F}_{\mathsf{RSD-ZK}}$.
2. $\mathcal{S}_{\mathcal{P}}$ samples the random challenges $L^{(i)}$ for $i \in [0, w)$ and $\boldsymbol{\chi}^{\mathsf{QS}}, \boldsymbol{\chi}^{\mathsf{open}} \leftarrow \mathbb{F}_{2^\lambda}^w$ and sends them to the adversary.
3. $\mathcal{S}_{\mathcal{P}}$ receives the QuickSilver proof messages $m_0^{\mathsf{QS}}, m_1^{\mathsf{QS}}, m_0^{\mathsf{open}}$ from the adversary.
4. $\mathcal{S}_{\mathcal{P}}$ simulates the (`get`) command of $\mathcal{F}_{\mathsf{sVOLE}}$ and the QuickSilver checking phase. In particular, $\mathcal{S}_{\mathcal{P}}$ sends $\perp$ to the ideal functionality in the following two cases.
   - Let $\boldsymbol{e} = [\boldsymbol{e}_0 \| ... \| \boldsymbol{e}_{w-1}]$. There exists $i \in [0, w)$ s.t. $\|\boldsymbol{e}_i\|_0 \neq 1$ or $\|\boldsymbol{e}_i\|_0 = 1$ but the non-zero element is not 1.
   - Let $e_0^{\mathsf{QS}} = \mathsf{M}[u^{\mathsf{QS}}] + \sum_{i \in [0,w)} \chi_i^{\mathsf{QS}} \cdot \mathsf{M}[z_0^{(i)}] \cdot \mathsf{M}[z_1^{(i)}] - m_0^{\mathsf{QS}}$ and $e_1^{\mathsf{QS}} = u^{\mathsf{QS}} + \sum_{i \in [0,w)} \chi_i^{\mathsf{QS}} \cdot (z_0^{(i)} \cdot \mathsf{M}[z_1^{(i)}] + z_1^{(i)} \cdot \mathsf{M}[z_0^{(i)}] - \mathsf{M}[z_2^{(i)}]) - m_1^{\mathsf{QS}}$ be the errors in the QuickSilver messages while $e_0^{\mathsf{open}} = \sum_{i \in [0,w)} \chi_i^{\mathsf{open}} \cdot \mathsf{M}[z_3^{(i)}] - m_0^{\mathsf{open}}$ be the error in the opening message. We have $e_0^{\mathsf{QS}} \neq 0$ or $e_1^{\mathsf{QS}} \neq 0$ or $e_0^{\mathsf{open}} \neq 0$.

   Otherwise, $\mathcal{S}_{\mathcal{P}}$ sends `continue` to the ideal functionality.

Since the protocol is public-coin, the simulation of the verifier's messages is identically distributed with the interaction of the real verifier. Now we analyze the soundness error, which captures the difference between the abort probability of the real case and the ideal case. If the verifier in the real world rejects, then either the relation does not hold (i.e. the witness has too large or zero Hamming weight) or the QuickSilver messages are malformed. In both cases, the ideal verifier also rejects the proof.

Now we focus on the case where the real verifier accepts while the simulator rejects. If the extracted witness $\boldsymbol{e}$ does not satisfy the Hamming weight constraint, then by Lemma 1 we conclude

that except with probability $\frac{2}{2^\lambda}$ there exists at least one index $i \in [0, w)$ such that the quadratic relation $z_0^{(i)} \cdot z_1^{(i)} = z_2^{(i)}$ and $z_3^{(i)} = 1$ does not hold. In this case, we can re-write the real verifier's first acceptance condition as follows.

$$\begin{aligned} m_0^{\mathsf{QS}} + m_1^{\mathsf{QS}} \cdot \Delta = \mathsf{K}[u^{\mathsf{QS}}] + (\sum_{i\in[0,w)} \chi_i^{\mathsf{QS}} \cdot (z_0^i \cdot z_1^i - z_2^i)) \cdot \Delta^2 \\ + \sum_{i\in[0,w)} \chi_i^{\mathsf{QS}} \cdot (z_0^i \cdot \mathsf{M}[z_1^i] + z_1^i \cdot \mathsf{M}[z_0^i] + \mathsf{M}[z_2^i]) \cdot \Delta \\ + \sum_{i\in[0,w)} \chi_i^{\mathsf{QS}} \cdot \mathsf{M}[z_0^i] \cdot \mathsf{M}[z_1^i] \end{aligned}$$

Since the $\boldsymbol{\chi}^{\mathsf{QS}}$ challenge is sampled uniformly at random and independent from other randomness, except with probability $\frac{1}{2^\lambda}$ the quadratic term of the above equation is non-zero. In this case, there exists at most two solutions to the equation. Since Δ is sampled uniformly at random and independent from other randomness, the equation holds with at most $\frac{2}{2^\lambda}$ probability.

Moreover, if $z_3^{(i)} \neq 1$ for some $i \in [0, w)$, since $\boldsymbol{\chi}^{\mathsf{open}}$ is uniformly random over $\mathbb{F}_{2^\lambda}^w$, the equality $\sum_{i\in[0,w)} \chi_i^{\mathsf{open}} \cdot z_3^{(i)} = \sum_{i\in[0,w)} \chi_i^{\mathsf{open}}$ holds except with $\frac{1}{2^\lambda}$ probability. Assuming the equality does not hold, the adversary can pass the check except it correctly guesses the Δ value, which happens except with $\frac{1}{2^\lambda}$ probability. Using the union bound, we conclude that the soundness error is upper bounded by $\frac{7}{2^\lambda}$.

Semi-Honest Verifier. The simulator $\mathcal{S}_{\mathcal{V}}$ is constructed as follows.

1. $\mathcal{S}_{\mathcal{V}}$ simulates the (`init`) command of the functionality $\mathcal{F}_{\mathsf{sVOLE}}$ and receives the messages $\boldsymbol{\Delta}$ and $\mathbf{W}'$ from the adversary. Then it samples $\boldsymbol{d} \leftarrow \mathbb{F}_2^k$ and sends it to the adversary.
2. $\mathcal{S}_{\mathcal{V}}$ receives the random challenges $L^{(i)}$ for $i \in [0, w)$ and $\boldsymbol{\chi}^{\mathsf{QS}}, \boldsymbol{\chi}^{\mathsf{open}} \in \mathbb{F}_{2^\lambda}^w$ from the adversary.
3. $\mathcal{S}_{\mathcal{V}}$ samples a random value $m_1^{\mathsf{QS}} \leftarrow \mathbb{F}_{2^\lambda}$ and computes $m_0^{\mathsf{QS}} = \mathsf{K}[u^{\mathsf{QS}}] + \sum_{i\in[0,w)} \chi_i^{\mathsf{QS}} \cdot (\mathsf{K}[z_0^{(i)}] \cdot \mathsf{K}[z_1^{(i)}] + \Delta \cdot \mathsf{K}[z_2^{(i)}]) - m_1^{\mathsf{QS}} \cdot \Delta$. It also prepares $m_0^{\mathsf{open}} = \sum_{i\in[0,w)} \chi_i^{\mathsf{open}} \cdot (\mathsf{K}[z_3^{(i)}] + \Delta)$. It sends $m_0^{\mathsf{QS}}, m_1^{\mathsf{QS}}, m_0^{\mathsf{open}}$ to the adversary.
4. $\mathcal{S}_{\mathcal{V}}$ simulates the (`get`) command of $\mathcal{F}_{\mathsf{sVOLE}}$ by sending $\boldsymbol{\Delta}, \mathbf{W}'$ to the adversary.

Notice that we only argue for security against a semi-honest adversary. Due to the masking of $\boldsymbol{u}[0, k)$, the message $\boldsymbol{d}$ is uniformly random in the view of the adversary. Also due to the masking of $\boldsymbol{u}[k, k+\lambda)$, the message m_1^{QS} is also uniformly random. Moreover, the messages $m_0^{\mathsf{QS}}, m_0^{\mathsf{open}}$ can be deterministically evaluated using $\Delta, \boldsymbol{d}, \{L^{(i)}\}, \boldsymbol{\chi}^{\mathsf{QS}}, \boldsymbol{\chi}^{\mathsf{open}}, \mathbf{W}', m_1^{\mathsf{QS}}$. Therefore, we conclude that the adversary's view is identical between the simulated case and the real case. □

5 ReSolveD: Shorter Signatures from RSD and VOLEitH

We apply the transformation in [BBdSG+23b] to convert the public-coin protocol $\Pi_{\mathsf{RSD-DVZK}}$ in the $\mathcal{F}_{\mathsf{sVOLE}}$-hybrid model into a publicly-verifiable zero-knowledge proof $\Pi_{\mathsf{RSD-PVZK}}$. Then we apply the Fiat-Shamir transform and present ReSolveD, a post-quantum digital signature scheme from RSD and VOLE-in-the-Head. We present the signature scheme in Figure 6.

- $\mathsf{KeyGen}()$
  1. Samples a generator matrix in systematic form $\mathbf{H} = [\mathbf{I} \| \mathbf{H_B}] \in \mathbb{F}_2^{(m-k)\times m}$ as well as a regular noise $\boldsymbol{e} \in \mathbb{F}_2^m$.
  2. Output $\mathsf{pk} = (\mathbf{H}, \boldsymbol{y} = \mathbf{H} \cdot \boldsymbol{e}), \mathsf{sk} = (\mathsf{pk}, \boldsymbol{e})$.
- $\mathsf{Sign}(\mathsf{sk}, m)$
  1. The signer executes the prover's actions of $\Pi_{\mathsf{RSD-PVZK}}$. For all challenges $\mathsf{H}^{\mathsf{UHF}}, \boldsymbol{\chi}^{\mathsf{QS}}, \boldsymbol{\chi}^{\mathsf{open}}, \boldsymbol{\Delta}$, the signer sends the protocol's transcript concatenated with the message m to the random oracle to get the respective challenges.
  2. Output signature σ as complete transcript of $\Pi_{\mathsf{RSD-PVZK}}$.
- $\mathsf{Verify}(\mathsf{pk}, m, \sigma)$
  1. The verifier executes the verifier's actions of $\Pi_{\mathsf{RSD-PVZK}}$. The verifier uses the prover's messages extracted from the signature σ while for the verifier's challenges, the verifier also hashes the partial transcript concatenated with the signed message m.
  2. The verifier accepts the signature if in the simulated execution of $\Pi_{\mathsf{RSD-PVZK}}$, the simualted verifier also accepts. It rejects if otherwise.

Figure 6: The ReSolveD signature scheme. We assume that the unary form of the security parameter λ is the implicit input of all three algorithms.

5.1 Signature Description

We describe the protocol $\Pi_{\mathsf{RSD-PVZK}}$ in Figure 8. We apply the half-tree optimization [GYW+23] when constructing the vector commitment scheme, which we recall in Figure 7 and prove its security in Lemma 2. Notice that in this construction, we utilize the circular correlation robustness property which is usually instantiated in the ideal cipher model [GKWY20], of which we recall a simplified version in Definition 2.

Lemma 2. *Let G_1, G_2 be two random oracles and H be a (t, q, ϵ)-circular correlation robust hash function. Then the vector commitment scheme $\Pi_{\mathsf{VC-cGGM}}$ (Figure 7) securely implements the vector commitment functionality $\mathcal{F}_{\mathsf{VC}}$ (Figure 3).*

Proof. The protocol correctness follows by definition. Now we argue security against a malicious $\mathcal{P}$ and $\mathcal{V}$ respectively.

Malicious prover. The simulator $\mathcal{S}_{\mathcal{P}}$ receives the commitment $\mathtt{com}$ from the adversary and then recovers the hashed values $\{\mathtt{com}_i\}_{i\in[0,N)}$ and $\{r_i^{\tau'}\}_{i\in[0,N)}$ from the random oracle queries and send $\{m_i\}_{i\in[0,N)}$ to $\mathcal{F}_{\mathsf{VC}}$ where $(m_i, \mathtt{com}_i) = G(r_i^{\tau'})$ for $i \in [0, N)$.

For the $(\mathtt{get}, \alpha)$ command, $\mathcal{S}_{\mathcal{P}}$ receives the de-commitment information $\mathtt{decom}_\alpha = \{K^i_{\bar{\alpha}_i}\}_{i\in[1,\tau']}$ from the adversary and runs the checking procedures of the verifier. If the check fails then it sends $\perp$ to $\mathcal{F}_{\mathsf{VC}}$.

Unless there exists a collision in the random oracle queries, then the ideal execution successfully extracts the committed messages. The collision probability is upper bounded by $\frac{Q}{2^\lambda}$ where Q is the number of random oracle queries from by $\mathcal{A}$.

Malicious verifier. The simulator $\mathcal{S}_{\mathcal{V}}$ samples $\mathtt{com} \leftarrow \{0,1\}^\lambda$ and sends it to $\mathcal{A}$ to simulate $(\mathtt{commit})$. For the $(\mathtt{get}, \alpha)$ command, the simulator samples $K^i_{\bar{\alpha}_i} \leftarrow \{0,1\}^\lambda$ for $i \in [1, \tau']$, $\mathtt{com}_\alpha \leftarrow \{0,1\}^\lambda$ and sends them to $\mathcal{A}$. Then it receives the $\{m_i\}_{i\in[0,N),i\neq\alpha}$ message from $\mathcal{F}_{\mathsf{VC}}$ and programs

Protocol $\Pi_{\mathsf{VC-cGGM}}$

Let $G_1 : \{0,1\}^* \to \{0,1\}^{2\lambda}, G_2 : \{0,1\}^\lambda \to \{0,1\}^\lambda \times \{0,1\}^{2\lambda}$ be two random oracles and $\mathsf{H} : \{0,1\}^\lambda \times \{0,1\}^\lambda \to \{0,1\}^\lambda$ be a hash function. Let $\tau' \in \mathbb{N}$ be the tree depth and define $N = 2^{\tau'}$.

- For the `commit` command, $\mathcal{P}$ and $\mathcal{V}$ perform the following steps.
  1. $\mathcal{P}$ samples $\Gamma \leftarrow \mathbb{F}_2^\lambda$, $r_0^1 \leftarrow \mathbb{F}_2^\lambda$ and computes $r_1^1 = r_0^1 \oplus \Gamma$. It evaluates the full binary tree using the recursive relation $r_{2j}^{i+1} = \mathsf{H}(r_j^i)$, $r_{2j+1}^{i+1} = \mathsf{H}(r_j^i) \oplus r_j^i$ for $i \in [2, \tau'), j \in [0, 2^i)$ and computes $(m_i, \mathtt{com}_i) \leftarrow G_2(r_i^{\tau'})$ for $i \in [0, 2^{\tau'})$ and the sum of all even indexed nodes on each level $\{K_0^i\}_{i\in[1,\tau']} = \sum_{j\in[0,2^{i-1})} r_{2j}^i$.
  2. $\mathcal{P}$ sends the commitment $\mathtt{com} := G_1(\mathtt{com}_0, ..., \mathtt{com}_{N-1})$ to $\mathcal{V}$ and locally stores the de-commitment information $\mathtt{decom} := (\Gamma, \{K_0^i\}_{i\in[1,\tau']})$ and the messages $\{m_i\}_{i\in[0,N)}$.
- For the ($\mathtt{get}$, α) command, $\mathcal{P}$ and $\mathcal{V}$ perform the following steps.
  1. Let $\alpha_1, \alpha_2, ..., \alpha_{\tau'}$ be the binary decomposition of $\alpha \in [0, N)$, $\mathcal{P}$ sends the opening information $\mathtt{decom}_\alpha := (\{K_{\bar{\alpha}_i}^i := K_0^i \oplus \bar{\alpha}_i \cdot \Gamma\}_{i\in[1,\tau']}, \mathtt{com}_\alpha)$ to $\mathcal{V}$.
  2. Upon receiving the opening information, $\mathcal{V}$ defines $r_{\bar{\alpha}_1}^1 = K_{\bar{\alpha}_1}^i$ from $\mathtt{decom}_\alpha$. For $i \in [2, \tau']$, $j \in [0, 2^{i-1})$ and $j \neq \alpha_1 \| ... \| \alpha_{i-1}$, it evaluates $r_{2j}^i = \mathsf{H}(r_j^{i-1})$ and $r_{2j+1}^i = \mathsf{H}(r_j^{i-1}) \oplus r_j^{i-1}$ and defines $r_{\alpha_1\|...\|\alpha_{i-1}\|\bar{\alpha}_i}^i = K_{\bar{\alpha}_i}^i \oplus \sum_{j\in[2^{i-1}], j\neq\alpha_1\|...\|\alpha_{i-1}} r_{2j+\bar{\alpha}_i}^i$.
  3. For each leaf node $i \in [0, N) \setminus \alpha$, $\mathcal{V}$ derives $(m_i, \mathtt{com}_i) = G_2(r_i^{\tau'})$. $\mathcal{V}$ checks that $\mathtt{com} = G_1(\mathtt{com}_0, ..., \mathtt{com}_{N-1})$ using the $\mathtt{com}_\alpha$ information in $\mathtt{decom}_\alpha$. If the equality does not hold then it outputs $\perp$. Otherwise, it outputs $\{m_i\}_{i\in[0,N), i\neq\alpha}$.

Figure 7: The correlated GGM tree construction.

the random oracle such that when evaluating the leaf nodes $r_i^{\tau'}$ for $i \in [0, N), i \neq \alpha$ the verification process would pass.

We argue indistinguishability via a hybrid argument.

- **Hybrid$_1$**. This is the real distribution of a malicious verifier.
- **Hybrid$_2$**. We sample $\mathtt{com}_\alpha$ uniformly at random and update $\mathtt{com} = G_2(\mathtt{com}_0, ..., \mathtt{com}_{N-1})$ accordingly. Since G_2 is a random oracle, the only way that an adversary can distinguish between **Hybrid$_1$** and **Hybrid$_2$** is by querying the pre-image of $\mathtt{com}_\alpha$ in **Hybrid$_1$**, which implies extracting Γ and contradicts the CCR security of H.
- **Hybrid$_3$**. In this hybrid, we sample $\{K_{\bar{\alpha}_i}^i\}_{i\in[1,\tau']}$ uniformly at random. We show in Lemma 3 that the adversary's advantage can be bounded by the CCR security of H.
- **Hybrid$_4$**. This is the ideal distribution, which is identical to **Hybrid$_3$**.

□

Lemma 3. *The advantage of distinguishing* **Hybrid$_2$** *and* **Hybrid$_3$** *in the proof of Lemma 2 can be bounded by the circular correlation robustness of the hash function* H.

Proof. We can sample the adversary's view using an oracle $\mathcal{O}(\cdot)$ such that the view corresponds to **Hybrid$_2$** (resp. **Hybrid$_3$**) if $\mathcal{O}$ is the real oracle $\mathcal{O}_\Gamma^{\mathsf{ccr}}$ (resp. the ideal oracle f) as follows.

- For $i = 1$, we sample $K^1_{\bar{\alpha}_1}$ uniformly at random.
- For $i \in [2, \tau']$ we compute

$$K^i_{\bar{\alpha}_i} = \begin{cases} \mathcal{O}(\bigoplus_{j=1}^{i-1} K^j_{\bar{\alpha}_j}, 0) & \text{if } \bar{\alpha}_i = 0, \\ \mathcal{O}(\bigoplus_{j=1}^{i-1} K^j_{\bar{\alpha}_j}, 1) \bigoplus_{j=1}^{i-1} K^j_{\bar{\alpha}_j} & \text{if } \bar{\alpha}_i = 1. \end{cases}$$

- Sample $\mathtt{com}_\alpha$ uniformly at random and compute $\mathtt{com} = G_2(\mathtt{com}_0, ..., \mathtt{com}_{N-1})$.

Notice that if $\mathcal{O}$ is a random function then the output distribution is **Hybrid$_3$** whereas if $\mathcal{O} = \mathcal{O}^{\mathsf{ccr}}_\Gamma(\cdot)$ then we have if $\bar{\alpha}_i = 0$ then

$$\begin{aligned} K^i_{\bar{\alpha}_i} &= \mathsf{H}(\bigoplus_{j=1}^{i-1} K^j_{\bar{\alpha}_j}) \\ &= \mathsf{H}(r^{i-1}_{\alpha_{i-1}\|...\|\alpha_1}) \end{aligned}$$

And if $\bar{\alpha}_i = 1$ then

$$\begin{aligned} K^i_{\bar{\alpha}_i} &= \mathsf{H}(\bigoplus_{j=1}^{i-1} K^j_{\bar{\alpha}_j}) \oplus \Delta \oplus \bigoplus_{j=1}^{i-1} K^j_{\bar{\alpha}_j} \\ &= \mathsf{H}(r^{i-1}_{\alpha_{i-1}\|...\|\alpha_1}) \oplus r^{i-1}_{\alpha_{i-1}\|...\|\alpha_1}, \end{aligned}$$

which is the same as in **Hybrid$_2$**. □

Remark 1. *We note that the construction of CCR hash functions in the random permutation model [GKWY20] requires a permutation on λ-bit strings. For some block ciphers (e.g. AES-128) the offered security level matches the block size and we can model the block cipher as a random permutation and apply the construction in [GKWY20]. Whereas other block ciphers with λ-bit security level do not provide a permutation on λ-bit strings (e.g. AES-192 and AES-256 has block size of 128 bits despite having higher security levels.) In this case, we use the standard GGM tree construction based on length-doubling PRG. We leave the efficient construction of CCR hash functions at the security level beyond 128 from standard symmetric primitives in the latter case (e.g. AES-192 and AES-256) as a future work.*

5.2 Security Proof

We state the security of our protocol $\Pi_{\mathsf{RSD-PVZK}}$ under Fiat-Shamir transformation in Theorem 2. Since we prove the protocol $\Pi_{\mathsf{VC-cGGM}}$ securely realizes the vector commitment functionality $\mathcal{F}_{\mathsf{VC}}$ in Lemma 2, the security proof of the conversion from designated-verifier zero-knowledge (Figure 5) to the non-interactive zero-knowledge is identical to the work of [BBdSG+23b], we omit it in this paper.

Theorem 2. *Let $\mathsf{PRG} : \{0,1\}^\lambda \to \mathbb{F}_2^n$ be a pseudorandom generator, the zero-knowledge protocol $\Pi_{\mathsf{RSD-PVZK}}$, after Fiat-Shamir transformation, is a zero-knowledge non-interactive proof system in the random oracle model.*

Protocol $\Pi_{\mathsf{RSD-PVZK}}$

Both parties $\mathcal{P}$ and $\mathcal{V}$ have access to the RSD instance $(\mathbf{H}, \boldsymbol{y})$ where $\mathbf{H} \in \mathbb{F}_2^{(m-k)\times m}$ and $\boldsymbol{y} = \mathbf{H} \cdot \boldsymbol{e}$. The prover also knows the witness $\boldsymbol{e} = [\boldsymbol{e}_0 \| ... \| \boldsymbol{e}_{w-1}]$ where $\|\boldsymbol{e}_i\|_0 = 1$ for $i \in [0, w)$. We assume the matrix $\mathbf{H} = [\mathbf{I}_{m-k} \| \mathbf{H}_{\mathbf{B}}]$ is in the systematic form and the vector $\boldsymbol{e}$ is split into $\boldsymbol{e}_{\mathbf{A}}$ and $\boldsymbol{e}_{\mathbf{B}}$. Let $n = k+\lambda$, $N = 2^{\tau'}$ and $\mathcal{H}^{\mathsf{UHF}} \subseteq \mathbb{F}_2^{r\times(n+h)}$ be a family of n-hiding, ϵ-universal hash function. Let $\mathsf{PRG} : \{0,1\}^\lambda \to \mathbb{F}_2^{n+h}$ be a pseudorandom generator. We assume without loss of generality that $\tau\tau' = \lambda$.

1. $\mathcal{P}$ and $\mathcal{V}$ run τ instances of the vector commitment functionality $\mathcal{F}_{\mathsf{VC}}$ and send (`commit`) to them. Denote the messages as $\{k_j^i\}$ for $i \in [0, \tau)$, $j \in [0, N)$. The prover then defines $\mathbf{U}', \mathbf{V}$ for the index $i \in [0, \tau)$.

$$\mathbf{U}' = \begin{bmatrix} | & & | \\ \sum_j \mathsf{PRG}(k_j^0) & \dots & \sum_j \mathsf{PRG}(k_j^{\tau-1}) \\ | & & | \end{bmatrix}, \mathbf{V} = \begin{bmatrix} | & & | \\ \sum_j j \cdot \mathsf{PRG}(k_j^0) & \dots & \sum_j j \cdot \mathsf{PRG}(k_j^{\tau-1}) \\ | & & | \end{bmatrix}$$

   The prover also defines $\boldsymbol{u} = \mathbf{U}'[0]$ as the first column of $\mathbf{U}'$ and $\mathbf{C} := [\mathbf{U}'[1] \oplus \boldsymbol{u} \| \cdots \| \mathbf{U}'[\tau-1] \oplus \boldsymbol{u}] \in \mathbb{F}_2^{n\times(\tau-1)}$ where $\mathbf{U}'[i]$ denotes the i-th column of $\mathbf{U}'$. $\mathcal{P}$ sends $\mathbf{C}$ to $\mathcal{V}$.

2. $\mathcal{V}$ samples random challenge $\mathsf{H}^{\mathsf{UHF}} \leftarrow \mathcal{H}^{\mathsf{UHF}}$ and sends it to $\mathcal{P}$.

3. $\mathcal{P}$ defines the SoftSpokenOT check messages $\tilde{\boldsymbol{u}} = \mathsf{H}^{\mathsf{UHF}} \cdot \boldsymbol{u}$ and $\tilde{\mathbf{V}} = \mathsf{H}^{\mathsf{UHF}} \cdot \mathbf{V}$ and sends them to $\mathcal{V}$.

4. $\mathcal{P}$ and $\mathcal{V}$ run step 1–3 of $\Pi_{\mathsf{RSD-DVZK}}$ using the first n rows of $\boldsymbol{u}$ and $\mathbf{V}$.

5. $\mathcal{V}$ samples the random challenge $\boldsymbol{\Delta} = (\Delta_0, ..., \Delta_{\tau-1})$ and call (`get`, Δ_i) for $i \in [0, \tau)$. With the opened messages, it computes

$$\mathbf{W}' := \begin{bmatrix} | & & | \\ \sum_j (j + \Delta_0) \cdot \mathsf{PRG}(k_j^0) & \dots & \sum_j (j + \Delta_{\tau-1}) \cdot \mathsf{PRG}(k_j^{\tau-1}) \\ | & & | \end{bmatrix}$$

If the following two checks pass then $\mathcal{V}$ accepts the proof. Otherwise, $\mathcal{V}$ rejects.

- **SoftSpokenOT.** $\mathcal{V}$ checks that

$$\tilde{\mathbf{V}} + \tilde{\boldsymbol{u}} \cdot [1 \dots 1] \cdot \mathsf{diag}(\boldsymbol{\Delta}) = \mathsf{H}^{\mathsf{UHF}} \cdot (\mathbf{W}' + [0 \| \mathbf{C}] \cdot \mathsf{diag}(\boldsymbol{\Delta})) \quad .$$

- **QuickSilver.** The verifier runs the consistency check in step 4 of $\Pi_{\mathsf{RSD-DVZK}}$.

Figure 8: The publicly verifiable zero-knowledge protocol for regular syndrome decoding.

5.3 Communication

We theoretically estimate the communication cost of $\Pi_{\mathsf{RSD-PVZK}}$. Firstly, during the inner protocol $\Pi_{\mathsf{RSD-DVZK}}$ the prover needs to send $\boldsymbol{d}$ and $m_0^{\mathsf{QS}}, m_1^{\mathsf{QS}}, m_0^{\mathsf{open}}$, which takes k elements in $\mathbb{F}_2$ and 3 elements in $\mathbb{F}_{2^\lambda}$ respectively.

Moreover, during the simulation of the $\mathcal{F}_{\mathsf{sVOLE}}$ setup, the prover needs to run τ instances of the vector commitment protocol $\Pi_{\mathsf{VC-cGGM}}$, each of which the communication cost is $(\tau' + 4) \cdot \lambda$ bits. Then, in SoftSpokenOT the prover needs to send the de-randomization matrix $\mathbf{C}$ as well as the checking information $\tilde{\boldsymbol{u}}, \tilde{\mathbf{V}}$, which takes $(\tau - 1) \cdot (k + \lambda + h)$ elements in $\mathbb{F}_2$, r elements in $\mathbb{F}_2$ and $r \cdot \tau$ elements in $\mathbb{F}_{2^{\tau'}}$ respectively.

Optimizations. We can use some existing techniques in the literature to optimize communication [CCJ23, BBdSG+23a]. We list three main optimizations as follows.

- When running τ instances of $\Pi_{\mathsf{VC-cGGM}}$ the commitment message $\mathtt{com}$ can be combined by hashing all the leaf nodes across τ binary trees at once, saving $2\lambda \cdot (\tau - 1)$ bits of communication.
- Since in the RSD witness all elements in a block XOR to 1, the prover can commit to the first $\frac{m}{w} - 1$ coordinates of each block and linearly express the remaining element. Thus, we can reduce the witness length by a ratio of $\frac{w}{m}$.
- The values $\tilde{\mathbf{V}}, m_0^{\mathsf{QS}}, m_0^{\mathsf{open}}$ can be computed by the verifier and therefore to check for equality, it suffices for the prover to send a hash of those values.

Taking into account all the optimizations outlined above, we conclude the theoretical estimate of the communication of $\Pi_{\mathsf{RSD-PVZK}}$ as follows.

$$\text{Comm} = \underbrace{\left((1-\frac{w}{m})k + \lambda + h\right) \cdot (\tau - 1)}_{\mathbf{C}} + \underbrace{r}_{\tilde{\boldsymbol{u}}} + \underbrace{(1-\frac{w}{m})k}_{\boldsymbol{d}} + \underbrace{\lambda}_{m_1^{\mathsf{QS}}} + \underbrace{((\tau'+2)\cdot\tau + 2)\cdot\lambda}_{\text{VC Openings}} + \underbrace{2\lambda}_{\text{EQ}} \text{ bits.}$$

6 Performance Evaluation

In this section, we implement the ReSolveD signature scheme, which achieves highly competitive performance and a much smaller signature size when compared to other state-of-the-art code-based signature schemes. We first describe the parameters and implementation details of the scheme, then we report the evaluation results in terms of signature and key sizes as well as running time.

6.1 Parameters

We follow the approach from prior art [CCJ23] in the selection of parameters for the regular syndrome decoding instance. In particular, we select the minimal parameters that can offer the required bit security against state-of-the-art attacks that account for the regularity of the noise vector. Specifically, we estimate the complexity of the linearization attack, information syndrome decoding (ISD) attack and birthday paradox according to the formulas in [CCJ23] and take their minimal as the estimation of bit security. Using this estimation, we choose the smallest parameters that have complexity estimation of 2^{128}, 2^{143}, 2^{207} and 2^{272} according to the practice of previous works and the NIST's L1, L3 and L5 security levels.[2]

Table 3: The parameters for the ReSolveD signature scheme.

Parameter Set	m	k	w	τ	Estimated Bit Security
ReSolveD-128-Var1	1302	738	217	14	128.20
ReSolveD-128-Var2	1302	738	217	10	128.20
ReSolveD-L1	1470	834	245	11	143.20
ReSolveD-L3	2196	1248	366	17	207.48
ReSolveD-L5	2934	1668	489	22	272.29

[2]We select the parameters of ReSolveD-128 and ReSolveD-L{1,3,5} independently, because the former targets at signatures with 128-bit security while the latter targets other NIST submissions.

Regarding the parameters of VOLE-in-the-Head, we follow the approach in the specification of FAEST [BBdSG+23a]. In particular, with the security parameter λ and the repetition parameter τ, we compute $\tau_0' = \lceil \lambda/\tau \rceil$, $\tau_1' = \lfloor \lambda/\tau \rfloor$, and $\tau_0 = \lambda \mod \tau$, $\tau_1 = \tau - \tau_0$. In this way, since $\tau_0\tau_0' + \tau_1\tau_1' = \lambda$, we can ensure that by sampling τ_0 instances of $\mathcal{F}_{\mathsf{VC}}$ with depth τ_0' and τ_1 instances of $\mathcal{F}_{\mathsf{VC}}$ with depth τ_1' we can get a global key with λ bits of entropy.

We select parameters such that our scheme demonstrates better performance in terms of the "signature size + public-key size" metric while still maintaining comparable running time compared to other NIST submissions. The parameters are shown in Table 3. We note that the parameter selection listed in Table 3 has considered recent attacks that exploit the regular noise structure [BØ23, ES23].

We implement our signature scheme by adapting the implementation of FAEST[3].

We run the experiments on a Ubuntu 20.04 LTS machine with an AMD Ryzen 5 3600 CPU and 16GB of RAM. For the time being, we only optimized the 128-bit version of ReSolveD with the AVX2 instruction set while we leave the respective optimization of ReSolveD-L1, ReSolveD-L3 and ReSolveD-L5 to a future work. The performance of ReSolveD under the first two sets of parameters with AVX2 optimization is reported in Table 1, while we compare the un-optimized version of ReSolveD under the other three sets of parameters with the reference implementation of other NIST submissions in the next subsection.

6.2 Comparison with Other Post-Quantum Signature Schemes

We give a detailed comparison between ReSolveD and NIST's new submissions SDitH [MFG+23] and FAEST [BBdSG+23a] in Table 4. This is because SDitH shares a similarity in the underlying intractability assumption and FAEST utilizes the same VOLE-in-the-Head technique. In summary, ReSolveD and FAEST share almost the same secret key size, but ReSolveD is smaller in signature size while faster in signing and verification time than the short version of FAEST and its EM variant, with only slightly larger public key size and slower key generation time in the same security level. The size of ReSolveD outperforms SDitH where the signature size (resp., secret key size) is more than 2× (resp., 12×) smaller than that in SDitH. However, the running time of our scheme is much slower.

We also compare our ReSolveD with post-quantum signatures to be standardized by NIST including Dilithium [LDK+22], Falcon [PFH+22] and SPHINCS$^+$ [HBD+22] and other previous/new submissions to NIST such as Picnic [ZCD+20] and SPHINCS-α [ZCY23] in Table 5. Lattice-based signatures are currently the most efficient post-quantum signature schemes which achieve both smaller signature sizes and faster running times. However, these schemes are based on structured lattice problems such as Ring/Module-LWE and NTRU, on the contrary, our ReSolveD relies on no algebraic or geometric structures. Meanwhile, ReSolveD is competitive with Dilithium in the "signature size + public-key size" metric (with the former being 3.91 KB and the latter being 3.64 KB), although the runtimes of Dilithium significantly outperform us. Compared with the SPHINCS family and the Picnic family, our ReSolveD also achieves about $2\times \sim 4\times$ smaller in sizes than SPHINCS$^+$ and SPHINCS-α, and is more than 3× smaller than Picnic. Nevertheless, our ReSolveD is slower in terms of signing and verification. We plan to develop an optimized implementation of our scheme in the future work.

[3]We adapted the reference implementation of FAEST at https://github.com/faest-sign/faest-ref. We also note that OpenSSL is required to facilitate fast evaluation

Table 4: Detailed comparison of ReSolveD compared to NIST's new submissions SDitH and FAEST with its EM variants for NIST security L1, L3 and L5.

Scheme	Sizes in Bytes				Runtimes in ms			Assumption
	|sig|	|sk|	|pk|	|sig| + |pk|	t_{keygen}	t_{sign}	t_{verify}	
ReSolveD-L1	3916	32	96	4012	4.36	97.51	80.21	RSD over $\mathbb{F}_2$
ReSolveD-L3	8532	48	143	8675	9.97	257.37	226.71	RSD over $\mathbb{F}_2$
ReSolveD-L5	14944	64	191	15135	17.66	537.54	469.72	RSD over $\mathbb{F}_2$
FAEST-L1-S	5006	32	32	5038	0.19	129.14	124.89	AES
FAEST-L3-S	12744	56	64	12808	1.01	401.76	371.87	AES
FAEST-L5-S	22100	64	64	22164	1.47	624.62	586.12	AES
FAEST_EM-L1-S	4566	32	32	4598	0.18	112.06	108.85	EM-AES
FAEST_EM-L3-S	10824	48	48	10872	0.46	297.66	288.40	EM-AES
FAEST_EM-L5-S	20956	64	64	21020	1.41	540.35	540.04	EM-AES
SDitH-L1-gf256	8224	404	120	8344	6.08	33.23	28.62	SD over $\mathbb{F}_{256}$
SDitH-L1-gf251	8224	404	120	8344	4.41	14.76	12.32	SD over $\mathbb{F}_{251}$
SDitH-L3-gf256	19544	616	183	19727	7.31	113.98	98.82	SD over $\mathbb{F}_{256}$
SDitH-L3-gf251	19544	616	183	19727	5.30	34.46	28.32	SD over $\mathbb{F}_{251}$
SDitH-L5-gf256	33992	812	234	34226	10.59	209.67	186.77	SD over $\mathbb{F}_{256}$
SDitH-L5-gf251	33992	812	234	34226	8.74	59.33	54.85	SD over $\mathbb{F}_{251}$

Acknowledgements

We thank anonymous reviewers for their helpful comments. Yu Yu is supported by the National Key Research and Development Program of China (Grant No. 2020YFA0309705), the National Natural Science Foundation of China (Grant Nos. 62125204 and 92270201), and the Major Program of Guangdong Basic and Applied Research (Grant No. 2019B030302008). Yu Yu also acknowledges the support from the XPLORER PRIZE. Kang Yang is supported by the National Natural Science Foundation of China (Grant Nos. 62102037 and 61932019).

References

[ABG+19] Nicolas Aragon, Olivier Blazy, Philippe Gaborit, Adrien Hauteville, and Gilles Zémor. Durandal: A rank metric based signature scheme. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part III*, volume 11478 of *LNCS*, pages 728–758. Springer, Heidelberg, May 2019.

[ACBH13] Sidi Mohamed El Yousfi Alaoui, Pierre-Louis Cayrel, Rachid El Bansarkhani, and Gerhard Hoffmann. Code-based identification and signature schemes in software. In Alfredo Cuzzocrea, Christian Kittl, Dimitris E. Simos, Edgar R. Weippl, and Lida Xu, editors, *Security Engineering and Intelligence Informatics - CD-ARES 2013 Workshops: MoCrySEn and SeCIHD, Regensburg, Germany, September 2-6, 2013. Proceedings*, volume 8128 of *Lecture Notes in Computer Science*, pages 122–136. Springer, 2013.

Table 5: Comparison of signature sizes and runtimes at NIST L1 security for some standardized schemes and previous/new submissions from the NIST PQC standardization project. Numbers for Picnic are taken from [KZ20] running on a 3.6GHz Intel Xeon W-2133, others are taken from their technical report with a base clock frequency of up 2.6 GHz Intel Core i7-6600U CPU for Dilithium, a 2.3 GHz Intel Core i5-8259U for Falcon, a 3.1 GHz Intel Xeon E3-1220 CPU for SPHINCS$^+$ and a 3.6 GHz AMD Ryzen 5 3600 CPU for SPHNICS-α.

Scheme	Sizes in KB			Runtimes in ms		Assumption
	$\|\mathsf{sig}\|$	$\|\mathsf{pk}\|$	$\|\mathsf{sig}\| + \|\mathsf{pk}\|$	t_{sign}	t_{verify}	
Dilithium2 [LDK+22]	2.36	1.28	3.64	0.128	0.046	MLWE
Falcon-512 [PFH+22]	0.65	0.88	1.53	0.168	0.036	NTRU
SPHINCS$^+$-SHAKE-L1-F [HBD+22]	16.69	0.03	16.72	18.37	1.08	Hash
SPHINCS$^+$-SHAKE-L1-S [HBD+22]	7.67	0.03	7.70	355.64	0.38	Hash
SPHINCS$^+$-SHA2-L1-F [HBD+22]	16.69	0.03	16.72	10.86	0.69	Hash
SPHINCS$^+$-SHA2-L1-S [HBD+22]	7.67	0.03	7.70	207.98	0.28	Hash
SPHINCS-α-SHAKE-L1-F [ZCY23]	16.33	0.03	16.36	15.85	0.99	Hash
SPHINCS-α-SHAKE-L1-S [ZCY23]	6.72	0.03	6.75	316.60	1.36	Hash
SPHINCS-α-SHA2-L1-F [ZCY23]	16.33	0.03	16.36	7.40	0.56	Hash
SPHINCS-α-SHA2-L1-S [ZCY23]	6.72	0.03	6.75	149.18	0.75	Hash
Picnic1-L1-FS [ZCD+20]	32.09	0.03	32.12	1.37	1.10	AES
Picnic2-L1-FS [ZCD+20]	12.05	0.03	12.08	40.95	18.20	AES
Picnic3-L1 [ZCD+20]	12.30	0.03	12.33	5.17	3.96	AES
Picnic3-L1-K12 [ZCD+20]	12.30	0.03	12.33	3.98	2.87	AES
Picnic3-L1-64 [ZCD+20]	11.14	0.03	11.17	23.25	17.21	AES
Picnic3-5-L1 [ZCD+20]	13.38	0.03	13.41	5.59	4.63	AES
ReSolveD-L1	3.82	0.09	3.91	95.51	80.21	RSD

[AFS03] Daniel Augot, Matthieu Finiasz, and Nicolas Sendrier. A fast provably secure cryptographic hash function. Cryptology ePrint Archive, Report 2003/230, 2003. https://eprint.iacr.org/2003/230.

[AGH+23] Carlos Aguilar Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. The return of the SDitH. LNCS, pages 564–596. Springer, Heidelberg, June 2023.

[Bar94] S. Barg. Some new NP-complete coding problems. *Probl. Inf. Transm.*, 30(3):209–214, 1994.

[BBdSG+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: Algorithm Specifications. Technical report, National Institute of Standards and Technology, 2023. available at https://faest.info/faest-spec-v1.1.pdf.

[BBdSG+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, 2023.

[BBMH+21] Carsten Baum, Lennart Braun, Alexander Munch-Hansen, Benoît Razet, and Peter Scholl. Appenzeller to brie: Efficient zero-knowledge proofs for mixed-mode arithmetic and Z2k. In Giovanni Vigna and Elaine Shi, editors, *ACM CCS 2021*, pages 192–211. ACM Press, November 2021.

[BBMHS22] Carsten Baum, Lennart Braun, Alexander Munch-Hansen, and Peter Scholl. Moz$\mathbb{Z}_{2^k}$arella: Efficient vector-OLE and zero-knowledge proofs over $\mathbb{Z}_{2^k}$. In *CRYPTO 2022, Part IV*, LNCS, pages 329–358. Springer, Heidelberg, August 2022.

[BBPS21] Alessandro Barenghi, Jean-François Biasse, Edoardo Persichetti, and Paolo Santini. LESS-FM: Fine-tuning signatures from the code equivalence problem. In Jung Hee Cheon and Jean-Pierre Tillich, editors, *Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021*, pages 23–43. Springer, Heidelberg, 2021.

[BDOZ11] Rikke Bendlin, Ivan Damgård, Claudio Orlandi, and Sarah Zakarias. Semi-homomorphic encryption and multiparty computation. In Kenneth G. Paterson, editor, *EUROCRYPT 2011*, volume 6632 of *LNCS*, pages 169–188. Springer, Heidelberg, May 2011.

[BGI16] Elette Boyle, Niv Gilboa, and Yuval Ishai. Function secret sharing: Improvements and extensions. In Edgar R. Weippl, Stefan Katzenbeisser, Christopher Kruegel, Andrew C. Myers, and Shai Halevi, editors, *ACM CCS 2016*, pages 1292–1303. ACM Press, October 2016.

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

[BMPS20] Jean-François Biasse, Giacomo Micheli, Edoardo Persichetti, and Paolo Santini. LESS is more: Code-based signatures without syndromes. In Abderrahmane Nitaj and Amr M. Youssef, editors, *AFRICACRYPT 20*, volume 12174 of *LNCS*, pages 45–65. Springer, Heidelberg, July 2020.

[BMRS21] Carsten Baum, Alex J. Malozemoff, Marc B. Rosen, and Peter Scholl. Mac'n'cheese: Zero-knowledge proofs for boolean and arithmetic circuits with nested disjunctions. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part IV*, volume 12828 of *LNCS*, pages 92–122, Virtual Event, August 2021. Springer, Heidelberg.

[BMvT78] Elwyn R. Berlekamp, Robert J. McEliece, and Henk C. A. van Tilborg. On the inherent intractability of certain coding problems (corresp.). *IEEE Trans. Inf. Theory*, 24(3):384–386, 1978.

[BØ23] Pierre Briaud and Morten Øygarden. A new algebraic approach to the regular syndrome decoding problem and implications for PCG constructions. LNCS, pages 391–422. Springer, Heidelberg, June 2023.

[CCJ23] Eliana Carozza, Geoffroy Couteau, and Antoine Joux. Short signatures from regular syndrome decoding in the head. LNCS, pages 532–563. Springer, Heidelberg, June 2023.

[CDD+19] Ignacio Cascudo, Ivan Damgård, Bernardo David, Nico Döttling, Rafael Dowsley, and Irene Giacomelli. Efficient UC commitment extension with homomorphism for free (and applications). In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part II*, volume 11922 of *LNCS*, pages 606–635. Springer, Heidelberg, December 2019.

[CPS23] Tung Chou, Edoardo Persichetti, and Paolo Santini. On linear equivalence, canonical forms, and digital signatures. Cryptology ePrint Archive, Paper 2023/1533, 2023. https://eprint.iacr.org/2023/1533.

[CVA10] 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, 2010.

[DILO22] Samuel Dittmer, Yuval Ishai, Steve Lu, and Rafail Ostrovsky. Improving line-point zero knowledge: Two multiplications for the price of one. pages 829–841. ACM Press, 2022.

[DIO21] Samuel Dittmer, Yuval Ishai, and Rafail Ostrovsky. Line-point zero knowledge and its applications. In *2nd Conference on Information-Theoretic Cryptography*, 2021.

[DST19] Thomas Debris-Alazard, Nicolas Sendrier, and Jean-Pierre Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In Steven D. Galbraith and Shiho Moriai, editors, *ASIACRYPT 2019, Part I*, volume 11921 of *LNCS*, pages 21–51. Springer, Heidelberg, December 2019.

[ES23] Andre Esser and Paolo Santini. Not just regular decoding: Asymptotics and improvements of regular syndrome decoding attacks. Cryptology ePrint Archive, Paper 2023/1568, 2023. https://eprint.iacr.org/2023/1568.

[FJR22] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In *CRYPTO 2022, Part II*, LNCS, pages 541–572. Springer, Heidelberg, August 2022.

[FJR23] Thibauld Feneuil, Antoine Joux, and Matthieu Rivain. Shared permutation for syndrome decoding: new zero-knowledge protocol and code-based signature. *Des. Codes Cryptogr.*, 91(2):563–608, 2023. First appeared online at https://eprint.iacr.org/2021/1576.

[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, Heidelberg, August 1987.

[GG07] Philippe Gaborit and Marc Girault. Lightweight code-based identification and signature. In *IEEE International Symposium on Information Theory, ISIT 2007, Nice, France, June 24-29, 2007*, pages 191–195. IEEE, 2007.

[GKWY20] Chun Guo, Jonathan Katz, Xiao Wang, and Yu Yu. Efficient and secure multiparty computation from fixed-key block ciphers. In *2020 IEEE Symposium on Security and Privacy*, pages 825–841. IEEE Computer Society Press, May 2020.

[GPS22] Shay Gueron, Edoardo Persichetti, and Paolo Santini. Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup. *Cryptogr.*, 6(1):5, 2022.

[GYW+23] Xiaojie Guo, Kang Yang, Xiao Wang, Wenhao Zhang, Xiang Xie, Jiang Zhang, and Zheli Liu. Half-tree: Halving the cost of tree expansion in COT and DPF. In *EUROCRYPT 2023, Part I*, LNCS, pages 330–362. Springer, Heidelberg, June 2023.

[HBD+22] Andreas Hülsing, Daniel J. Bernstein, Christoph Dobraunig, Maria Eichlseder, Scott Fluhrer, Stefan-Lukas Gazdag, Panos Kampanakis, Stefan Kölbl, Tanja Lange, Martin M. Lauridsen, Florian Mendel, Ruben Niederhagen, Christian Rechberger, Joost Rijneveld, Peter Schwabe, Jean-Philippe Aumasson, Bas Westerbaan, and Ward Beullens. SPHINCS+. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[HOSS18] Carmit Hazay, Emmanuela Orsini, Peter Scholl, and Eduardo Soria-Vazquez. TinyKeys: A new approach to efficient multi-party computation. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 3–33. Springer, Heidelberg, August 2018.

[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.

[KOS15] Marcel Keller, Emmanuela Orsini, and Peter Scholl. Actively secure OT extension with optimal overhead. In Rosario Gennaro and Matthew J. B. Robshaw, editors, *CRYPTO 2015, Part I*, volume 9215 of *LNCS*, pages 724–741. Springer, Heidelberg, August 2015.

[KZ20] Daniel Kales and Greg Zaverucha. Improving the performance of the Picnic signature scheme. *IACR TCHES*, 2020(4):154–188, 2020. https://tches.iacr.org/index.php/TCHES/article/view/8680.

[LDK+22] Vadim Lyubashevsky, Léo Ducas, Eike Kiltz, Tancrède Lepoint, Peter Schwabe, Gregor Seiler, Damien Stehlé, and Shi Bai. CRYSTALS-DILITHIUM. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[LWYY22] Hanlin Liu, Xiao Wang, Kang Yang, and Yu Yu. The hardness of LPN over any integer ring and field for PCG applications. Cryptology ePrint Archive, Report 2022/712, 2022. https://eprint.iacr.org/2022/712.

[MFG+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. The Syndrome Decoding in the Head (SD-in-the-Head) Signature Scheme. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/SDitH-spec-web.pdf.

[MGS11] Carlos Aguilar Melchor, Philippe Gaborit, and Julien Schrek. A new zero-knowledge code based identification scheme with reduced communication. In *2011 IEEE Information Theory Workshop, ITW 2011, Paraty, Brazil, October 16-20, 2011*, pages 648–652. IEEE, 2011.

[MHJ+23] Carlos Aguilar Melchor, Andreas Hülsing, David Joseph, Christian Majenz, Eyal Ronen, and Dongze Yue. Sdith in the QROM. In Jian Guo and Ron Steinfeld, editors, *ASIACRYPT 2023, Part VII*, volume 14444 of *LNCS*, pages 317–350. Springer, 2023.

[NIS22] NIST. Call for additional digital signature schemes for the post-quantum cryptography standardization process, 2022. https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/call-for-proposals-dig-sig-sept-2022.pdf.

[NNOB12] Jesper Buus Nielsen, Peter Sebastian Nordholt, Claudio Orlandi, and Sai Sheshank Burra. A new approach to practical active-secure two-party computation. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 681–700. Springer, Heidelberg, August 2012.

[OOS17] Michele Orrù, Emmanuela Orsini, and Peter Scholl. Actively secure 1-out-of-N OT extension with application to private set intersection. In Helena Handschuh, editor, *CT-RSA 2017*, volume 10159 of *LNCS*, pages 381–396. Springer, Heidelberg, February 2017.

[PFH+22] Thomas Prest, Pierre-Alain Fouque, Jeffrey Hoffstein, Paul Kirchner, Vadim Lyubashevsky, Thomas Pornin, Thomas Ricosset, Gregor Seiler, William Whyte, and Zhenfei Zhang. FALCON. Technical report, National Institute of Standards and Technology, 2022. available at https://csrc.nist.gov/Projects/post-quantum-cryptography/selected-algorithms-2022.

[PS23] Edoardo Persichetti and Paolo Santini. A new formulation of the linear equivalence problem and shorter less signatures. Cryptology ePrint Archive, Paper 2023/847, 2023.

[PSS17] Arpita Patra, Pratik Sarkar, and Ajith Suresh. Fast actively secure OT extension for short secrets. In *NDSS 2017*. The Internet Society, February / March 2017.

[Roy22] Lawrence Roy. SoftSpokenOT: Quieter OT extension from small-field silent VOLE in the minicrypt model. In *CRYPTO 2022, Part I*, LNCS, pages 657–687. Springer, Heidelberg, August 2022.

[Sch80] Jacob T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *J. ACM*, 27(4):701–717, 1980.

[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, Heidelberg, August 1994.

[Vér96] Pascal Véron. Improved identification schemes based on error-correcting codes. *Appl. Algebra Eng. Commun. Comput.*, 8(1):57–69, 1996.

[WYKW21] Chenkai Weng, Kang Yang, Jonathan Katz, and Xiao Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. In *2021 IEEE Symposium on Security and Privacy*, pages 1074–1091. IEEE Computer Society Press, May 2021.

[WYX+21] Chenkai Weng, Kang Yang, Xiang Xie, Jonathan Katz, and Xiao Wang. Mystique: Efficient conversions for zero-knowledge proofs with applications to machine learning. In Michael Bailey and Rachel Greenstadt, editors, *USENIX Security 2021*, pages 501–518. USENIX Association, August 2021.

[WYY+22] Chenkai Weng, Kang Yang, Zhaomin Yang, Xiang Xie, and Xiao Wang. AntMan: Interactive zero-knowledge proofs with sublinear communication. pages 2901–2914. ACM Press, 2022.

[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.

[ZCY23] Kaiyi Zhang, Hongrui Cui, and Yu Yu. SPHINCS-alpha. Technical report, National Institute of Standards and Technology, 2023. available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures.

[Zip79] Richard Zippel. Probabilistic algorithms for sparse polynomials. In Edward W. Ng, editor, *Symbolic and Algebraic Computation, EUROSAM '79, An International Symposiumon Symbolic and Algebraic Computation, Marseille, France, June 1979, Proceedings*, volume 72 of *Lecture Notes in Computer Science*, pages 216–226. Springer, 1979.

Supplementary Material

A Parameter Selection Script

Here we present the parameter selection script written in SageMath. The bit security estimation functions are derived from the formulas in [CCJ23].

```
def bitsec_lin(m,k,w):
    d = m / w
    cost_mat = (m-k)^3
    num = binomial(d-1, d-1-floor((m-k)/w))
    den = binomial(d-2,d-1-floor((m-k)/w))
    prob_inv = ( num / den )^((1-1/d)*w)
    return log(cost_mat * prob_inv, 2)

def bitsec_isd(m,k,w):
    f = lambda q : q * (1 - log(m/w,2)/(2*m/w)) + \
        log(q / (m-k-q), 2) - k * log(m/w,2) / (2*m/w)
    low, high = 1, m-k-1
    mid = (high-low)/2 + low + 1
    while mid != high:
        if f(mid) > 0:
            high = mid
        elif f(mid) < 0:
            low = mid
        else:
            break
        mid = floor(low + (high - low) / 2 + 1)
    q = mid
    cost1 = q * (k + q) / (m / w) * (m / w)^((k+q)/12)
    cost2 = (m / w)^((k+q)/(m/w)) / 2^q * (m - k - q) * (k+q)/(m/w)
    return log(cost1+cost2, 2)

def bitsec_gba(m,k,w):
    r = k - w
    cost = (m / w)^(r/(2*m/w)-1) * r * k
    return log(cost, 2)

def comm_cost (m, k, w):
    return k * (1 - w / m)

def bitsec (m,k,w):
    return min (bitsec_lin(m, k, w),
                bitsec_isd(m, k, w),
                bitsec_gba(m, k, w) )

def find_k (params):
    csp = params[0]
    d   = params[1]
    w   = params[2]
    results = params[3]
    m   = w * d
    khigh = floor(m - w * log(d,2))
```

```
    klow = ceil (m - log( binomial (m, w), 2 ) )
    min_cost = 999999999999
    min_k = -1
    if bitsec(m,khigh,w) < csp:
        return
    for k in [klow..khigh]:
        if k % d != 0:
            continue
        security = bitsec(m,k,w)
        if security >= csp and comm_cost(m,k,w) < min_cost:
            min_cost = comm_cost(m,k,w)
            min_k = k
            break
    if min_cost != 999999999999:
        results[(csp, d, w)] = min_k

csp_list = [128, 143, 207, 272]
results = dict()
for csp in csp_list:
    for d in [2..10]:
        for w in [100..500]:
            results [(csp, d, w)] = None

for csp in csp_list:
    min_cost = 999999999999
    min_param = (-1, -1, -1)
    for d in [2..10]:
        for w in [100..500]:
            find_k([csp, d, w, results])
            if results[(csp, d, w)] is not None:
                k = results[(csp, d, w)]
                m = d * w
                if comm_cost(m,k,w) < min_cost:
                    min_cost = comm_cost(m,k,w)
                    min_param = (m,k,w)
    m,k,w = min_param
    print (csp, m, k, w, bitsec(m, k, w).n(20), sep='\t')
```