Short Signatures from Regular Syndrome Decoding in the Head

Eliana Carozza[1], Geoffroy Couteau[2], and Antoine Joux[3]

[1] IRIF, Université Paris Cité, Paris, France.
carozza@irif.fr
[2] CNRS, IRIF, Université Paris Cité, Paris, France.
couteau@irif.fr
[3] CISPA Helmholtz Center for Information Security, Saarbrücken, Germany. joux@cispa.de

Abstract. We introduce a new candidate post-quantum digital signature scheme from the regular syndrome decoding (RSD) assumption, an established variant of the syndrome decoding assumption which asserts that it is hard to find w-regular solutions to systems of linear equations over $\mathbb{F}_2$ (a vector is regular if it is a concatenation of w unit vectors). Our signature is obtained by introducing and compiling a new 5-round zero-knowledge proof system constructed using the MPC-in-the-head paradigm. At the heart of our result is an efficient MPC protocol in the preprocessing model that checks correctness of a regular syndrome decoding instance by using a share ring-conversion mechanism.

The analysis of our construction is non-trivial and forms a core technical contribution of our work. It requires careful combinatorial analysis and combines several new ideas, such as analyzing soundness in a relaxed setting where a cheating prover is allowed to use any witness *sufficiently close* to a regular vector. We complement our analysis with an in-depth overview of existing attacks against RSD.

Our signatures are competitive with the best-known code-based signatures, ranging from 12.52 KB (fast setting, with signing time of the order of a few milliseconds on a single core of a standard laptop) to about 9 KB (short setting, with estimated signing time of the order of 15 ms).

1 Introduction

In this work, we introduce a new zero-knowledge proof for proving knowledge of a solution to the regular syndrome decoding problem, using the MPC-in-the-head paradigm. Compiling our zero-knowledge proof into a signature scheme using the Fiat-Shamir paradigm yields a new scheme with plausible post-quantum security and highly competitive performances compared to the state of the art.

Zero-knowledge, signatures, and syndrome decoding. Zero-knowledge proofs of knowledge allow a prover to convince a verifier of his knowledge of a witness for a NP statement without revealing anything beyond this. Zero-knowledge proofs enjoy countless applications in cryptography. In particular, the Fiat-Shamir transform [FS87] allows to convert any public-coin zero-knowledge proof system into a signature scheme; this transformation is one of the leading approaches to the construction of efficient signature schemes.

The syndrome decoding problem asks, given a matrix $H \in \mathbb{F}_2^{k\times K}$ and a target vector $y \in \mathbb{F}_2^k$, to find a vector $x \in \mathbb{F}_2^K$ of Hamming weight w such that $H \cdot x = y$. The average-case hardness of the syndrome decoding problem (for random matrices H and appropriate parameters (K, k, w)) is one of the leading candidate post-quantum cryptographic assumptions. The first zero-knowledge proof of knowledge for the syndrome decoding problem was introduced in the seminal work of Stern [Ste94] three decades ago. Unfortunately, Stern's proof has a large *soundness error*: a cheating prover can convince a verifier with probability 2/3 without knowing a correct solution x. To achieve a low soundness error, e.g. 2^{-128}, the protocol must therefore be repeated τ times, with τ such that $(2/3)^\tau \leq 2^{-128}$. This adds a significant communication overhead, resulting in a large signature size after compilation with Fiat-Shamir.

Code-based signatures. Digital signatures form the backbone of authentication on the Internet. However, essentially all deployed constructions will be rendered insecure in the presence of a quantum computer [Sho94]. This motivates the search for alternative constructions of digital signature schemes, that rely on assumptions conjectured to withstand quantum computers. The recent call of the NIST[4] for standardizing post-quantum primitives has boosted the research for efficient post-quantum signatures, particularly code-based signatures.

Among the many candidate code-based signature schemes, the Fiat-Shamir approach, used in the seminal work of Stern, has received careful scrutiny [Beu20, GPS21, FJR21, BGKM22, FJR22a]. Indeed, while some alternative approaches such as Wave [DST19] and Durandal [ABG$^+$19] manage to reach smaller signature sizes (under somewhat more exotic but plausible assumptions), they typically require the signer to know a *trapdoor* associated with the code matrix, leading to huge public keys (since the public key must include the full matrix H). In contrast, Fiat-Shamir signatures require no such trapdoor, and the random matrix H can be heuristically compressed to a short seed using a pseudorandom generator, yielding comparatively tiny public keys (in addition to relying on more traditional assumptions). This comes at the expense of slightly larger signature sizes. Nevertheless, the standard efficiency measure (size of the signature + size of the public key) strongly favors the Fiat-Shamir line of work.

The MPC in the head paradigm. Several recent works on Fiat-Shamir code-based digital signatures use the *MPC in the head* paradigm, introduced in the seminal work of [IKOS07] (MPC stands for *multiparty computation*). At a high level, this paradigm lets the prover run an MPC protocol in his head, where the (virtual) parties are given shares of the witness, and the target function verifies that the witness is correct. Then, the prover commits to the views of all parties, and the verifier asks to see a random subset of the views, checks that they are consistent, and that the output indeed corresponds to the witness being correct. Soundness stems from the fact that a cheating prover (not knowing a valid witness) cannot produce consistent views for all parties, and zero-knowledge follows from the security of the MPC protocol against a honest-but-curious adversary (which gets to see the views of a subset of corrupted parties).

The MPC in the head paradigm reduces the construction of efficient zero-knowledge proofs to the search for suitable MPC protocols with low communication overhead. In recent years, it has led to some of the most competitive candidate post-quantum signature schemes [KKW18, BdK$^+$21], and was used in particular in the most efficient Fiat-Shamir code-based signature scheme (and the most efficient code-based signature scheme overall, under the signature + public key size metric) known to date [FJR22a].

1.1 Our Contribution

In this work, we introduce a new zero-knowledge proof system for a variant of the syndrome decoding problem, using the MPC in the head paradigm. The variant of the syndrome decoding problem which we consider is the *regular syndrome decoding* (RSD) problem. Given a matrix $H \in \mathbb{F}_2^{k\times K}$ and a syndrome $y \in \mathbb{F}_2^k$, the RSD problem with parameters (k, K, w) asks to find a weight-w *regular* solution $x \in \mathbb{F}_2^K$ to $H \cdot x = y$, where regular means that x is a concatenation of w unit vectors (*i.e.*, x is divided in w equal-length blocks, and has a single 1 per block). The regular syndrome decoding problem is a well-established variant of syndrome decoding: it was introduced in 2003 in [AFS03] as the assumption underlying the FSB candidate to the NIST hash function competition, and was subsequently analyzed in [FGS07, MDCYA11, BLPS11], among others. It has also been used and analyzed in many recent works on secure computation, such as [HOSS18, BCGI18, BCG$^+$19b, BCG$^+$19a, BCG$^+$20, YWL$^+$20, WYKW21, RS21, CRR21, BCG$^+$22].

Brief overview of our approach. While we use the MPC in the head paradigm, as in previous works [GPS21, FJR21, BGKM22, FJR22a], our choice of the underlying MPC protocol departs significantly from all previous work. Our starting point is the observation that checking $H \cdot x = y$ and checking the structure of x can each be done using linear operations, over $\mathbb{F}_2$ for the former, and over $\mathbb{Z}$ for the latter. In standard MPC protocol, linear operations over a ring $\mathcal{R}$ are usually "for free",

[4] https://csrc.nist.gov/projects/post-quantum-cryptography

provided that the values are shared over $\mathcal{R}$. Therefore, the only component that requires communication is a *share conversion* mechanism, to transform shares over $\mathbb{F}_2$ into shares over a larger integer ring. We introduce a share conversion protocol which exhibit very good performances. However, our protocol works in the preprocessing model, where the parties are initially given correlated random string by a trusted dealer. The use of preprocessing in the MPC in the head paradigm has appeared in previous works [KKW18, GPS21], and handling the preprocessing phase usually incurs a significant communication overhead (due to the need to check that it was correctly emulated by the prover).

Nevertheless, a core technical contribution of our work is a method, tailored to our setting, to handle the preprocessing phase *for free* (*i.e.* without incurring any communication overhead). At a high level, we achieve this by letting the verifier *randomly shuffle* the preprocessing strings, instead of verifying them. A careful and non-trivial combinatorial analysis shows that a cheating prover has very low probability of providing an accepting proof for *any* choice of the initial (pre-permutation) preprocessing strings, over the choice of the verifier permutation. Furthermore, we observe that the cheating probability becomes much lower if we focus on cheating provers using a witness which is *far* from a regular witness (in the sense that it has multiple non-weight-1 blocks). For an appropriate setting of the parameters, the hardness of finding solutions *close* to regular witnesses becomes equivalent to the standard regular syndrome decoding assumption (where the solution must be strictly regular), hence this relaxation of the soundness still yields a signature scheme (after compilation with Fiat-Shamir) whose security reduces to the standard RSD assumption.

To complement our analysis, we also provide an analysis of the RSD assumption. We analyze the relation of RSD to the standard syndrome decoding assumption depending on the parameter regime, and reviewed existing attacks on RSD from the literature, fine-tuning and improving the attacks on several occasions. Eventually, we develop a new "adversary-optimistic" attack against RSD, showing how a linear-time solution to the approximate birthday problem would yield faster algorithms for RSD (in our parameter choices, we assumed that such an algorithm exists for the sake of choosing conservative parameters). We provide a more in-depth overview in the technical overview (Section 3).

Performances. While analyzing our approach is relatively involved, the protocol structure is extremely simple. The computation of our zero-knowledge proof is mostly dominated by simple XORs, calls to a length-doubling PRG (which can be instantiated very efficiently from AES over platforms with hardware support for AES) and calls to a hash function. This is in contrast with previous works, which always involved much more complex operations, such as FFTs [FJR22a] or compositions of random permutations [FJR21, BGKM22]. While we do not yet have an optimized implementation of our new signature scheme (we plan to get such an implementation in a future work), we carefully estimated the runtime of all operations using standard benchmarks, making conservative choices when the exact cost was unclear (we explain our calculations in details in Section 7). Our conservative choices likely overestimate the real runtime of these operations. Of course, the runtimes extrapolated this way ignore other costs such as the cost of copying and allocating memory. Nevertheless, in Banquet, another candidate post-quantum signature scheme using the MPC-in-the-head paradigm, the memory costs were estimated to account for 25% of the total runtime. We therefore expect our extrapolated number to be relatively close to real runtimes with a proper implementation. Our numbers indicate that our signature scheme is highly competitive with the state of the art, even if our extrapolated runtimes are off by more than a factor two, which we view as a strong indication that an optimized implementation will achieve competitive runtimes.

For communication, we provide eight sets of parameters. The first four parameters use RSD parameters which guarantee a security reduction to the standard RSD assumption, and we view them as our main candidate parameters. They correspond respectively to a fast signature (rsd-f), two medium signatures (rsd-m1 and rsd-m2) achieving a reasonable speed/size tradeoff, and a short signature (rsd-s). The last four parameters (arsd-f, arsd-m1, arsd-m2, and arsd-s) use a more aggressive setting of the RSD parameters, where security reduce instead to a more exotic assumption, namely, the security of RSD when the adversary is allowed to find an *almost regular* solution (with some fixed number of "faulty blocks" allowed). Since this variant has not yet been thoroughly analyzed, we view these parameters mainly as a motivation for future cryptanalysis of variants of RSD with almost-regular solutions.

We represent in Table 1 the results of our estimations and compare them to the state-of-the-art in code-based signature schemes. Compared to the best-known code-based signature scheme of [FJR22a], our conservative scheme (under standard RSD) achieves significantly smaller signature sizes than their scheme based on syndrome decoding over $\mathbb{F}_2$ (12.52 KB for our fast variant versus 17 KB for Var2f, and 9.69 to 8.55 KB for our shorter variants versus 11.8 KB for Var2s). In terms of runtime, our estimates are significantly faster than their reported runtimes (except rsd-s, which is on par with Var2s), hence our runtimes should remain competitive with a proper implementation, even if memory costs turn out to be higher than expected. Their most efficient scheme (variants Var3f and Var3s) relies on the conjectured hardness of syndrome decoding assumption over $\mathbb{F}_{256}$, which has been much less investigated. Yet, our conservative RSD-based schemes remain competitive even with their most efficient scheme: we get slightly larger signatures (12.42 KB versus 11.5 KB, and 9.13 KB versus 8.26 KB), and comparable runtimes. Since the RSD assumption over $\mathbb{F}_2$ has been more investigated, we view our signature scheme as a competitive and viable alternative.

Table 1. Comparison of our signature scheme with other code-based signature schemes from the literature, for 128 bits of security. All timings are in millisecond. Reported timings are those extracted in [FJR22a] from the original publications, using a 3.5 Ghz Intel Xeon E3-1240 v5 for Wave, a 2.8 Ghz Intel Core i5-7440HQ for Durandal, and a 3.8 GHz Intel Core i7 for [FJR21,FJR22a]. Our timings are estimated runtimes with the methodology given in Section 7.2.

Scheme	$\lvert\mathsf{sgn}\rvert$	$\lvert\mathsf{pk}\rvert$	t_{sgn}	Assumption
Wave	2.07 KB	3.2 MB	300	large-weight SD over $\mathbb{F}_3$, $(U, U+V)$-codes indist.
Durandal - I	3.97 KB	14.9 KB	4	Rank SD over $\mathbb{F}_{2^m}$
Durandal - II	4.90 KB	18.2 KB	5	Rank SD over $\mathbb{F}_{2^m}$
LESS-FM - I	9.77 KB	15.2 KB	-	Linear Code Equivalence
LESS-FM - II	206 KB	5.25 KB	-	Perm. Code Equivalence
LESS-FM - III	11.57 KB	10.39 KB	-	Perm. Code Equivalence
[GPS21] - 256	24.0 KB	0.11 KB	-	SD over $\mathbb{F}_{256}$
[GPS21] - 256	19.8 KB	0.12 KB	-	SD over $\mathbb{F}_{1024}$
[FJR21] (fast)	22.6 KB	0.09 KB	13	SD over $\mathbb{F}_2$
[FJR21] (short)	16.0 KB	0.09 KB	62	SD over $\mathbb{F}_2$
[BGKM22] Sig1	23.7 KB	0.1 KB	-	SD over $\mathbb{F}_2$
[BGKM22] Sig2	20.6 KB	0.2 KB	-	(QC)SD over $\mathbb{F}_2$
[FJR22a] - Var1f	15.6 KB	0.09 KB	-	SD over $\mathbb{F}_2$
[FJR22a] - Var1s	10.9 KB	0.09 KB	-	SD over $\mathbb{F}_2$
[FJR22a] - Var2f	17.0 KB	0.09 KB	13	SD over $\mathbb{F}_2$
[FJR22a] - Var2s	11.8 KB	0.09 KB	64	SD over $\mathbb{F}_2$
[FJR22a] - Var3f	11.5 KB	0.14 KB	6	SD over $\mathbb{F}_{256}$
[FJR22a] - Var3s	8.26 KB	0.14 KB	30	SD over $\mathbb{F}_{256}$
Our scheme - rsd-f	12.52 KB	0.09 KB	2.8^*	RSD over $\mathbb{F}_2$
Our scheme - rsd-m1	9.69 KB	0.09 KB	17^*	RSD over $\mathbb{F}_2$
Our scheme - rsd-m2	9.13 KB	0.09 KB	31^*	RSD over $\mathbb{F}_2$
Our scheme - rsd-s	8.55 KB	0.09 KB	65^*	RSD over $\mathbb{F}_2$
Our scheme - arsd-f	11.25 KB	0.09 KB	2.4^*	f-almost-RSD over $\mathbb{F}_2$
Our scheme - arsd-m1	8.76 KB	0.09 KB	15^*	f-almost-RSD over $\mathbb{F}_2$
Our scheme - arsd-m2	8.28 KB	0.09 KB	28^*	f-almost-RSD over $\mathbb{F}_2$
Our scheme - arsd-s	7.77 KB	0.09 KB	57^*	f-almost-RSD over $\mathbb{F}_2$

* Runtimes obtained using conservative upper bounds on the cycle counts of all operations as described in Section 7.2, and assuming that the signature is ran on one core of a 3.8GHz CPU. We stress that these parameters ignore costs such as copying or allocating memory, and should be seen only as a first-order approximation of the real runtimes.

2 Preliminaries

Given an integer $n \in \mathbb{N}$, we denote by $[n]$ the set $\{1, \cdots, n\}$. We use bold lowercase for vectors, and uppercase for matrices. Given a vector $\mathbf{v} \in \mathbb{F}^n$ and a permutation $\pi : [n] \mapsto [n]$, we write $\pi(\mathbf{v})$ to denote the vector $(v_{\pi(1)}, v_{\pi(2)}, \cdots, v_{\pi(n)})$. Given $\mathbf{u}, \mathbf{v} \in \{0,1\}^n$, we write $\mathbf{u} \oplus \mathbf{v}$ for the bitwise-XOR of $\mathbf{u}$ and $\mathbf{v}$, and $\mathbf{u} \odot \mathbf{v}$ for the bitwise-AND (also called Schur product, or Hadamard product) of $\mathbf{u}$ and $\mathbf{v}$, and $\mathsf{HW}(\mathbf{u})$ for the Hamming weight of $\mathbf{u}$ (*i.e.* its number of nonzero entries). Given a set S, we write $s \leftarrow_r S$ to indicate that s is sampled uniformly from S. Given a probabilistic Turing machine $\mathcal{A}$ and an input x, we write $y \leftarrow_r \mathcal{A}(x)$ to indicate that y is sampled by running $\mathcal{A}$ on x with a uniform random tape, or $y \leftarrow \mathcal{A}(x; r)$ when we want to make the random coins explicit. We assume familiarity with some basic cryptographic notions, such as commitment schemes, collision-resistant hash functions, and the random oracle model.

Given a vector $\mathbf{u} \in \mathbb{Z}_T^\ell$ and an integer T, we write $(\mathbf{u_1}, \cdots, \mathbf{u_n}) \leftarrow_r [\![\mathbf{u}]\!]_T$ to indicate that the vectors $\mathbf{u_i}$ (called the *i-th additive share* of $\mathbf{u}$) are sampled uniformly at random over $\mathbb{Z}_T^\ell$ conditioned on $\sum_i \mathbf{u_i} = \mathbf{u}$. We sometime abuse this notation and write $[\![\mathbf{u}]\!]_T$ to denote the tuple $(\mathbf{u_1}, \cdots, \mathbf{u_n})$. For a vector $\mathbf{v} \in \{0,1\}^\ell$, we write $[\![\mathbf{v}]\!]_T$ with $T > 2$ using the natural embedding of $\{0,1\}$ into $\mathbb{Z}_T$.

2.1 Syndrome Decoding Problems

Given a weight parameter w, the syndrome decoding problem asks to find a solution of Hamming weight w (under the promise that it exists) to a random system of linear equations over $\mathbb{F}_2$. Formally, let $\mathcal{S}_w^K$ denote the set of all vectors of Hamming weight w over $\mathbb{F}_2^K$. Then:

Definition 1 (Syndrome Decoding Problem). *Let K, k, w be three integers, with $K > k > w$. The syndrome decoding problem with parameters (K, k, w) is defined as follows:*

- *(Problem generation) Sample $H \leftarrow_r \mathbb{F}_2^{k \times K}$ and $x \leftarrow_r \mathcal{S}_w^K$. Set $y \leftarrow H \cdot x$. Output (H, y)*
- *(Goal) Given (H, y), find $x \in \mathcal{S}_w^K$ such that $H \cdot x = y$.*

A pair (H, y) is called an *instance* of the syndrome decoding problem. In this work, we also consider variants of the syndrome decoding problem, with different restrictions on the solution vector x. In our context, it is useful to rephrase the constraint on x as a linear equation over $\mathbb{N}$: the solution vector x must satisfy the constraint $\langle x, \mathbf{1} \rangle = w$, where $\mathbf{1}$ is the all-1 vector, and the inner product is computed over the integers (note that this view is of course specific to syndrome decoding over $\mathbb{F}_2$). Other standard variants of syndrome decoding from the literature can also be viewed as instances of a more general notion of *syndrome decoding under* $\mathbb{N}$*-linear constraints*, which we introduce below:

Definition 2 (Syndrome Decoding under $\mathbb{N}$-Linear Constraints). *Let K, k, w, c be four integers, with $K > k > w$ and $k > c$. Let $L \in \mathbb{N}^{c \times K}$ be a matrix and $\mathbf{v} \in \mathbb{N}^c$ be a vector; we call $(L, \mathbf{v})$ the* $\mathbb{N}$-linear constraint. *We say that $(L, \mathbf{v})$ is a* feasible constraint *if it is possible to sample a uniformly random element from the set $\{x \in \{0,1\}^K : L \cdot x = \mathbf{v}\}$ in time* $\mathsf{poly}(K)$.

The syndrome decoding problem with parameters (K, k, w) and feasible $\mathbb{N}$-linear constraint $(L, \mathbf{v})$ is defined as follows:

- *(Problem generation) Sample a matrix $H \leftarrow_r \{0,1\}^{k \times K}$ and a vector $x \leftarrow_r \{x \in \{0,1\}^K : L \cdot x = \mathbf{v}\}$. Set $y \leftarrow H \cdot x \bmod 2$. Output (H, y).*
- *(Goal) Given (H, y), find $x \in \{0,1\}^K$ such that*
  - *$H \cdot x = y \bmod 2$ (the $\mathbb{F}_2$-linear constraint), and*
  - *$L \cdot x = \mathbf{v}$ over $\mathbb{N}$ (the $\mathbb{N}$-linear constraint).*

Examples. Setting $c = 1$, $L = (1, \cdots, 1)$, and $\mathbf{v} = w$ corresponds to the constraint "x has Hamming weight w", and is the standard syndrome decoding problem. A common variant in the literature [AFS03,FGS07,MDCYA11,BLPS11,HOSS18,BCGI18,BCG+19b,BCG+19a,BCG+20,YWL+20, WYKW21,RS21,CRR21,BCG+22] is the *regular* syndrome decoding problem, where x is instead required to be a concatenation of w unit vectors, each of length K/w. We recover this variant by setting $c = w$, $\mathbf{v} = (1, \cdots, 1)^\intercal$, and defining L as the matrix with rows $L_i = (0 \cdots 0, 1 \cdots 1, 0 \cdots 0)$, where the band of ones is from $(i-1) \cdot K/w + 1$ to $i \cdot K/w$. Eventually, the d-split syndrome decoding problem from [FJR22b], where the vector x is divided into d blocks of weight w/d, is also easily seen to fit in this framework.

2.2 Honest-Verifier Zero-Knowledge Arguments of Knowledge

Given a two-party interactive protocols between PPT algorithms A with input a and B with input b where only B gets an output, we introduce two random variables: $\langle A(a), B(b)\rangle$ denotes the output of the protocol, and $\textsc{View}(A(a), B(b))$ denotes the transcript of the protocol.

Definition. A honest-verifier zero-knowledge argument of knowledge with soundness error ε for a NP language $\mathcal{L} = \{x \in \{0,1\}^* \ : \ \exists w, (x,w) \in \mathcal{R}_\mathcal{L} \wedge |w| = \mathsf{poly}(|x|)\}$ with relation $\mathcal{R}_\mathcal{L}$ is a two-party interactive protocol between a prover P and a verifier V which satisfies the following properties:

- **Perfect Completeness:** for every $(x,w) \in \mathcal{R}_\mathcal{L}$, the verifier always accept the interaction with a honest prover: $\Pr[\langle \mathsf{P}(x,w), \mathsf{V}(x)\rangle = 1] = 1$.
- **ε-Soundness:** [BG93] for every PPT algorithm $\tilde{\mathsf{P}}$ such that $\Pr[\langle \tilde{\mathsf{P}}(x), \mathsf{V}(x)\rangle = 1] = \tilde{\varepsilon} > \varepsilon$, there exists an *extractor* algorithm $\mathcal{E}$ which, given rewindable black-box access to $\tilde{\mathsf{P}}$, outputs a valid witness w' for x in time $\mathsf{poly}(\lambda, (\tilde{\varepsilon} - \varepsilon)^{-1})$ with probability at least $1/2$.
- **Honest-Verifier Zero-Knowledge (HVZK):** an argument of knowledge is (computationally, statistically, perfectly) HVZK if there exists a PPT simulator Sim such that for every $(x,w) \in \mathcal{R}_\mathcal{L}$, $\mathsf{Sim}(x) \equiv \textsc{View}(\mathsf{P}(x,w), \mathsf{V}(x))$, where $\equiv$ denotes computational, statistical, or perfect indistinguishability between the distributions.

Gap-HVZK. A *gap* honest-verifier zero-knowledge argument of knowledge [CKY09] with gap $\mathcal{L}'$, where $\mathcal{L}' \supseteq \mathcal{L}$ is an NP language with relation $\mathcal{R}_{\mathcal{L}'}$, is defined as a honest-verifier zero-knowledge argument of knowledge, with the following relaxation of ε-soundness: the extractor $\mathcal{E}$ is only guaranteed to output a witness w' such that $(x, w') \in \mathcal{L}'$. Concretely, in our setting, the witness is a valid solution to the syndrome decoding problem, and the language $\mathcal{L}'$ contains all strings which are *sufficiently close* (in a well-defined sense) to a valid solution. This is similar in spirit to the notion of *soundness slack* often used in the context of lattice-based zero-knowledge proof, where the honest witness is a vector with small entries, and the extracted vector can have significantly larger entries.

2.3 The MPC-in-the-Head Paradigm

The MPC-in-the-head paradigm was introduced in the seminal work of [IKOS07]. It provides a compiler which, given an n-party secure computation protocol for computing a function f' in the honest-but-curious model, produces a honest-verifier zero-knowledge argument of knowledge of x such that $f(x) = y$, for some public value y, where f' is a function related to f. In our context, the focus is on zero-knowledge for syndrome decoding problems, for which, for example, a typical choice of f would include a hardcoded description of the matrix H and from a vector x, f would output $f(x) = (H \cdot x, \mathsf{HW}(x))$.

At a high level (and specializing to MPC in the head with all-but-one additive secret sharing – the original compiler is more general), the compiler proceeds by letting the prover additively share the witness x into $(x_1, \cdots, x_n)$ among n virtual parties $(P_1, \cdots, P_n)$, run in his head an MPC protocol for securely computing $f'(x_1, \cdots, x_n) = f(\sum_i x_i)$ (where the sum is over some appropriate ring), and commit to the views of all parties. Then, the verifier queries a random size-$(n-1)$ subset of all views, which the prover opens. The verifier checks that these views are consistent and that the output is correct – for example, equal to (y, w) (when proving knowledge of x such that $H \cdot x = y$ and $\mathsf{HW}(x) = w$). She accepts if all checks succeeded. Soundness follows from the fact that the MPC protocol is correct, hence if the prover does not know a valid x, one of the views must be inconsistent with the output being correct (the soundness error is therefore $1/n$). Honest-verifier zero-knowledge follows from the fact that the MPC protocol is secure against passive corruption of $n-1$ parties (and the fact that $n-1$ shares of x leak no information about x).

3 Technical Overview

In this section, we provide a detailed overview of our zero-knowledge proof, and highlight the technical challenges in constructing an analyzing the proof.

3.1 Our Template Zero-Knowledge Proof

We start with the construction of a zero-knowledge proof of knowledge of a solution to an instance of the syndrome decoding problem, using the MPC-in-the-head paradigm. More generally, our protocol handles naturally any *syndrome decoding under* $\mathbb{N}$*-linear constraints* problem for some $\mathbb{N}$-linear constraint $(L, \mathbf{v})$, see Definition 2. To this end, we construct an n-party protocol Π where the parties have shares of a solution $x \in \{0,1\}^K$ to the syndrome decoding problem, and securely output $H \cdot x \bmod 2$ and $L \cdot x$ over $\mathbb{N}$. Given the output of the MPC protocol, the verifier checks (1) that the execution (in the prover's head) was carried out honestly (by checking a random subset of $n-1$ views of the parties) and (2) that the two outputs are equal to y and $\mathbf{v}$ respectively.

The high level intuition of our approach is the following: in MPC protocols, it is typically the case that linear operations are extremely cheap (or even considered as "free"), because they can be computed directly over secret values shared using a linear secret sharing scheme (such as additive sharing, or Shamir sharing), without communicating. In turn, we observe that several variants of the syndrome decoding problem reduce to finding a solution x that satisfy two types of linear constraints: one linear constraint over $\mathbb{F}_2$ (typically, checking that $H \cdot x = y$ given a syndrome decoding instance (H, y)) and one linear constraint over $\mathbb{N}$ (*e.g.* checking that $\langle x, \mathbf{1} \rangle = w$, *i.e.* that the Hamming weight of x is w). Now, an appropriate choice of linear secret sharing scheme can make any one of these two constraints *for free* in Π: if x is additively shared over $\mathbb{F}_2$, then verifying $H \cdot x = y$ is for free, while if x is additively shared over a large enough integer ring $\mathcal{R} = \mathbb{Z}_T$ (such that no overflow occurs when computing $L \cdot x$ over $\mathbb{N}$ for any $x \in \{0,1\}^K$), then verifying $L \cdot x = \mathbf{v}$ is for free.

Share conversion. By the above observation, the only missing ingredient to construct Π is a *share conversion* mechanism: a protocol where the parties start with $\mathbb{F}_2$-shares $[\![x]\!]_2$ of x, and securely convert them to $\mathcal{R}$-shares $[\![x]\!]_T$ of x. Our next observation is that for any integer ring $\mathbb{Z}_T$, this can be done easily using appropriate *preprocessing material.* Consider the case of a single bit $a \in \{0,1\}$; the parties initially have $\mathbb{F}_2$-shares $[\![a]\!]_2$ of a. Suppose now that the parties receive the $([\![b]\!]_2, [\![b]\!]_T)$ for a random $b \in \{0,1\}$ from a trusted dealer. The parties can locally compute $[\![a \oplus b]\!]_2$ and open the bit $c = a \oplus b$ by broadcasting their shares. Now, since $a = c \oplus b = c + b - 2b$ over $\mathbb{N}$, only two cases may arise:

Case 1: $c = 1$. Then $a = 1 - b$ and so $[\![a]\!]_T = [\![1-b]\!]_T$.
Case 2: $c = 0$. Then $a = b$ and so $[\![a]\!]_T = [\![b]\!]_T$

Therefore, the parties can compute $[\![a]\!]_T$ as $c \cdot [\![1-b]\!]_T + (1-c) \cdot [\![b]\!]_T$. This extends directly to an integral solution vector x. Hence, in the protocol Π, prior to the execution, a trusted dealer samples a random vector $r \leftarrow_r \{0,1\}^K$ and distribute $([\![r]\!]_2, [\![r]\!]_T)$ to the parties, where T is such that no overflow can occur when computing $L \cdot x \bmod T$ (in order to simulate $\mathbb{N}$-linear operations). A similar technique was used previously, in a different context, in [RW19, EGK$^+$20].

The MPC protocol. Building on this observation, we introduce an MPC protocol in the preprocessing model, where the trusted dealer picks a random bitstring r, and distributes $([\![r]\!]_2, [\![r]\!]_T)$ to the parties. On input additive shares of the witness x over $\mathbb{F}_2$, the parties can open $z = r + x$. Using the above observation, all parties can reconstruct shares $[\![x]\!]_T$. Then, any linear equation on x over either $\mathbb{F}_2$ or $\mathbb{Z}_T$ can be verified by opening an appropriate linear combination of the $\mathbb{F}_2$-shares or of the $\mathbb{Z}_T$ shares (this last step does not add any communication when compiling the protocol into a zero-knowledge proof).

Handling the preprocessing material. At a high level, there are two standard approaches to handle preprocessing material using MPC-in-the-head. The first approach was introduced in [KKW18]. It uses a natural cut-and-choose strategy: the prover plays the role of the trusted dealer, and executes many instances of the preprocessing, committing to all of them. Afterwards, the verifier asks for openings of a subset of all preprocessings, and checks that all opened strings have been honestly constructed. Eventually, the MPC-in-the-head compiler is applied to the protocol, using the unopened committed instances of the preprocessing phases. This approach is very generic, but induces a large overhead, both computationally and communication-wise. The second approach is tailored to specific types of preprocessing material, such as Beaver triples. It is inspired by the classical sacrificing technique which allows to check the correctness of a batch of Beaver triples, while sacrificing only a few triples. It was used in works such as Banquet [BdK+21], or more recently in [FJR22a].

Unfortunately, the first approach induces a large overhead, and the second one is tailored to specific types of preprocessing material. In our context, the structure of the preprocessing material makes it unsuitable. Fortunately, we show that, in our setting, the preprocessing material can be handled *essentially for free.*

Our technique works as follows: we let the prover compute (and commit to) the preprocessing material ($\llbracket \mathbf{r} \rrbracket_2, \llbracket \mathbf{r} \rrbracket_T$) himself, but require that the coordinates of $\mathbf{r}$ are *shuffled using a uniformly random permutation* (chosen by the verifier) before being used in the protocol. Crucially, as we show in our analysis, the verifier *never* needs to check that the preprocessing phase was correctly executed (which would induce some overhead): instead, we demonstrate that a malicious prover (who does not know a valid witness) cannot find *any* (possibly incorrect) preprocessing material that allows him to pass the verification *with the randomly shuffled material* with high probability.

Fundamentally, the intuition is the following: it is easy for the malicious prover to know values x, x' such that $H \cdot x = y \bmod 2$ and $L \cdot x' = \mathbf{v} \bmod T$. To pass the verification test in the protocol, a malicious prover must therefore fine-tune malicious preprocessing strings $(\mathbf{s}, \mathbf{t})$ such that the value $\mathbf{z} \odot (\mathbf{1} - \mathbf{t}) + (\mathbf{1} - \mathbf{z}) \odot \mathbf{t}$, computed from $\mathbf{z} = \mathbf{s} \oplus x$ for some x such that $H \cdot x = y \bmod 2$, is equal to a value x' such that $L \cdot x' = \mathbf{v} \bmod T$ (recall that in the honest protocol, the prover should use $\mathbf{s} = \mathbf{t} = \mathbf{r}$). But doing so requires a careful choice of the entries (s_i, t_i): intuitively, the prover needs $s_i = t_i$ whenever $x_i = x'_i$, and $s_i = 1 - t_i$ otherwise. However, when the coordinates of $(\mathbf{s}, \mathbf{t})$ are randomly shuffled, this is not the case with high probability. While the high-level intuition is clear, we note that formalizing it requires particularly delicate combinatorial arguments.

Full description of the MPC protocol. Let (H, y) be an instance of the $\mathbb{N}$-linear syndrome decoding problem with parameters (K, k, w) and feasible $\mathbb{N}$-linear constraint $(L, \mathbf{v})$. Let $x \in \{0,1\}^K$ denote a solution for this instance. We construct an n-party protocol Π in the preprocessing model, where the parties inputs are additive shares of x over $\mathbb{F}_2$. The protocol Π securely computes $H \cdot x \bmod 2$ and $L \cdot x$ in the honest-but-curious setting, with corruption of up to $n - 1$ parties. Let $\mathsf{par} \leftarrow (K, k, w, c, H, L)$. The protocol Π_{par} is represented on Figure 1.

Parameters: The protocol Π operates with n parties, denoted $(P_1, \cdots, P_n)$. (K, k, w, c) are four integers with $K > k > w$ and $k > c$. $H \in \{0,1\}^{k \times K}$ and $L \in \mathbb{N}^{c \times K}$ are public matrices. Let $\mathsf{par} \leftarrow (K, k, w, c, H, L)$, and let $T \leftarrow \|L \cdot \mathbf{1}\|_1$. We view $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$ as forming additive shares $\llbracket x \rrbracket_2$ over $\mathbb{F}_2$ of a vector $x \in \{0,1\}^K$.
Inputs: Each party P_i has input $\mathbf{x}_i \in \{0,1\}^K$.
Preprocessing: The trusted dealer samples $\mathbf{r} \leftarrow_r \{0,1\}^K$. He computes $\llbracket \mathbf{r} \rrbracket_2 = (\mathbf{s}_1, \cdots, \mathbf{s}_n) \leftarrow_r \mathsf{Share}_2(\mathbf{r})$ and $\llbracket \mathbf{r} \rrbracket_T = (\mathbf{t}_1, \cdots, \mathbf{t}_n) \leftarrow_r \mathsf{Share}_T(\mathbf{r})$, viewing bits as elements of the integer ring $\mathbb{Z}_T$ in the natural way. It distributes $(\mathbf{s}_i, \mathbf{t}_i)$ to each party P_i.
Online Phase: The protocol proceeds in broadcast rounds.

- The parties compute $\llbracket \mathbf{y}' \rrbracket_2 = H \cdot \llbracket x \rrbracket_2$ and $\llbracket \mathbf{z} \rrbracket_2 = \llbracket \mathbf{r} \rrbracket_2 + \llbracket x \rrbracket_2$. All parties open $\mathbf{y}'$ and $\mathbf{z}$.
- The parties compute $\llbracket \mathbf{v}' \rrbracket_T \leftarrow L \cdot (\mathbf{z} \odot \llbracket \mathbf{1} - \mathbf{r} \rrbracket_T + (\mathbf{1} - \mathbf{z}) \odot \llbracket \mathbf{r} \rrbracket_T)$, viewing $\mathbf{z}$ as a vector over $\mathbb{Z}_T$ in the natural way.
- All parties open $\mathbf{v}'$.

Output. The parties output $(\mathbf{y}', \mathbf{v}')$.

Fig. 1. Protocol Π_{par} for securely computing $H \cdot x \bmod 2$ and $L \cdot x$ in the honest-but-curious up to $n - 1$ corruptions.

A template zero-knowledge proof. Building upon the above, we describe on Figure 2 a template zero-knowledge proof. Looking ahead, our final zero-knowledge proof does (1) instantiate this template for a carefully chosen flavor of syndrome decoding with $\mathbb{N}$-linear constraints, and (2) introduce many optimizations to the proof, building both upon existing optimizations from previous works, and new optimizations tailored to our setting.

Parameters. (K, k, w, c) are four integers with $K > k > w$ and $k > c$. $H \in \{0,1\}^{k\times K}$ and $L \in \mathbb{N}^{c\times K}$ are public matrices. $y \in \{0,1\}^k$ and $\mathbf{v} \in \{0,1\}^c$ are public vectors. Let $\mathsf{par} \leftarrow (K, k, w, c, H, L)$, and let $T \leftarrow \|\mathbf{v}\|_1$. Let Commit be a non-interactive commitment scheme.
Inputs. The prover and the verifier have common input par and $(y, \mathbf{v})$, which jointly form an instance of the $\mathbb{N}$-linear syndrome decoding problem. The prover additionally holds a witness $x \in \{0,1\}^K$ which is a solution of the instance: $H \cdot x = y \bmod 2$ and $L \cdot x = \mathbf{v} (= \mathbf{v} \bmod T)$.
Witness Sharing. The prover samples $(\mathbf{x}_1, \cdots, \mathbf{x}_n) \leftarrow_r \mathsf{Share}_2(x)$. Each share $\mathbf{x}_i$ is the input of the virtual party P_i.
Round 1. The prover runs the trusted dealer of Π_{par} and obtains $((\mathbf{s}_1, \cdots, \mathbf{s}_n), (\mathbf{t}_1, \cdots, \mathbf{t}_n)) = ([\![\mathbf{r}]\!]_2, [\![\mathbf{r}]\!]_T)$. He computes and sends $c_i \leftarrow_r \mathsf{Commit}(\mathbf{x}_i, \mathbf{s}_i, \mathbf{t}_i)$ for $i = 1$ to n to the verifier.
Round 2. The verifier picks a uniformly random permutation $\pi \leftarrow_r S_K$ and sends it to the prover.
Round 3. The prover runs the online phase of Π_{par} where the parties $(P_1, \cdots, P_n)$ have inputs $(\mathbf{x}_1, \cdots, \mathbf{x}_n)$, using the shuffled preprocessing material $([\![\pi(\mathbf{r})]\!]_2, [\![\pi(\mathbf{r})]\!]_T)$. For each party P_i, let $\mathsf{msg}_i = (\mathbf{y}'_i, \mathbf{z}_i, \mathbf{v}'_i)$ denote the list of all messages sent by P_i during the execution. The prover sends $(\mathsf{msg}_1, \cdots, \mathsf{msg}_n)$ to the verifier.
Round 4. The verifier chooses a challenge $d \in [n]$ and sends it to the prover.
Round 5 The prover opens all commitments c_j for $j \neq d$ to the verifier.
Verification. The verifier checks:

- that all commitments were opened correctly;
- that the output of Π_{par} with transcript $(\mathsf{msg}_1, \cdots, \mathsf{msg}_n)$ is equal to $(y, \mathbf{v})$;
- that the messages msg_j sent by P_j are consistent with $(\mathbf{x}_j, \mathbf{s}_j, \mathbf{t}_j)$.

The verifier accepts if and only if all checks succeed.

Fig. 2. Template 5-round zero-knowledge proof for $\mathbb{N}$-linear syndrome decoding using MPC-in-the-head with the protocol Π_{par}

3.2 Concrete Instantiation for Regular Syndrome Decoding

With the template construction in mind, we can now discuss our concrete choice of syndrome decoding problem with $\mathbb{N}$-linear constraints. Our target is the *regular syndrome decoding problem*, where the linear constraint states that the witness x should be a concatenation of w unit vectors (see Section 2.1). The rationale behind this choice stems from the communication complexity of the template zero-knowledge proof from Figure 2. Intuitively, the communication is dominated by the cost of transmitting the vectors over the ring $\mathbb{Z}_T$ (*i.e.* the $\mathbf{t}_i$ vectors): sending each such vector requires $K \cdot \log T$ bits. Looking ahead, even with proper optimizations, the zero-knowledge proof cannot be competitive with state-of-the-art constructions communication-wise whenever the value of $T = \|L \cdot \mathbf{1}\|_1$ is large.

Typically, for the standard syndrome decoding problem, we have $T = K$, hence the communication involves a $K \cdot \log K$ term, and the overhead is too large (when choosing concrete parameters, K is typically in the thousands, hence $K \log K$ is of the order of a few kilobytes, which becomes a few dozen kilobytes after parallel repetitions). On the other hand, *regular* syndrome decoding appears to minimize this cost: the value of T is only K/w. Hence, by choosing the weight appropriately, we can reduce the value of T.

The regular syndrome decoding problem is also far from new: it was introduced in [AFS03] as the assumption underlying the security of a candidate for the SHA-3 competition, and was subsequently studied in numerous works, including [FGS07, MDCYA11, BLPS11], and more recently in [HOSS18]. The hardness of the regular syndrome decoding problem is also the core assumption underlying many recent works in secure computation with silent preprocessing, see *e.g.* [BCGI18, BCG$^+$19b, BCG$^+$19a, BCG$^+$20, YWL$^+$20, WYKW21, RS21, CRR21, BCG$^+$22] and references therein. It is therefore a well-established assumption.

In the following, we focus on the regular syndrome decoding problem as our primary instantiation of the template. Looking ahead, we seek to minimize the value of $T = K/w$. Concretely, as we show in Section 4.1, a standard chinese remainder theorem trick allows to work over the ring $\mathbb{Z}_{T/2}$ instead of $\mathbb{Z}_T$, as long as $\gcd(T/2, 2) = 1$ (*i.e.* $T/2$ is odd; intuitively, this is because the "mod 2 part" of the equation $L \cdot x = \mathbf{v} \bmod T$ can be obtained at no cost from the $\mathbb{F}_2$-sharing of x, hence it only remains to get $L \cdot x \bmod T/2$ and use the CRT to reconstruct $L \cdot x \bmod T$). The smallest possible value of $T/2$ satisfying the above constraint is $T/2 = 3$, implying $T = K/w = 6$. We therefore set $w = K/6$, which is the smallest value of w that sets the bitsize of the vectors $\mathbf{t}_i$ to its minimal value of $K \cdot \log(T/2) = K \cdot \log 3$.

3.3 Combinatorial Analysis

Our discussion so far hinged upon the assumption that when the preprocessing material is randomly shuffled by the verifier, a cheating prover has very low success probability. A core technical contribution of our work is to provide a bound on this success probability. We define (informally) a *combinatorial bound* to be a quantity p that bounds the probability of a cheating prover to find preprocessing material that causes the verifier to accept the interaction.

Definition 3 (Combinatorial bound – informal). *A real* $\mathsf{p} \in (0,1)$ *is a* combinatorial bound *for the template zero-knowledge proof if for every incorrect witness* x*, and every pair* $(\mathbf{s},\mathbf{t})$*, the probability, over the random choice of a permutation* π*, that* x *satisfies the following equations:*

- $x' = \mathbf{z} \odot (\mathbf{1} - \pi(\mathbf{t})) + (\mathbf{1} - \mathbf{z}) \odot \pi(\mathbf{t})$ *with* $\mathbf{z} = \pi(\mathbf{s}) \oplus x$
- $H \cdot x = y \bmod 2$*,* $L \cdot x = \mathbf{v} \bmod 2$*, and* $L \cdot x' = \mathbf{v} \bmod T/2$

is upper-bounded by p.

Note that the last two equations stem from the use of the gcd trick, where the "mod 2 part" of the equation $L \cdot x = \mathbf{v} \bmod T$ is verified directly on the original shares of x modulo 2, and the remaining equation is checked modulo $T/2$ (assuming that $\gcd(2, T/2) = 1$). Proving a tight combinatorial bound turns out to be a highly non-trivial task. In this section, we overview the key technical challenges one faces, and outline our solution.

A balls-and-bins analysis. A key difficulty in the analysis is that we must handle arbitrary choices of the strings $(\mathbf{s},\mathbf{t})$ chosen by the prover, but also arbitrary (invalid) witnesses x. In our concrete instantiation, we use the regular syndrome decoding problem, and always enforce $T = K/w = 6$ (this is the choice which maximizes efficiency). Therefore, we focus on this setting in our analysis. In this case, the setting becomes: assume that we are given an incorrect witness x, which is a concatenation of w length-T blocks $x^1, \cdots, x^w$. The equation $L \cdot x = \mathbf{v} \bmod 2$ translates to the condition that each block x^j has odd Hamming weight; since $T = 6$, we have $\mathsf{HW}(x^j) \in \{1,3,5\}$ for $j = 1$ to w.

Let us now fix a position $i \leq K$. The pair $(s_{\pi(i)}, t_{\pi(i)}) \in \mathbb{F}_2 \times \mathbb{F}_3$ "transforms" x_i into x'_i as follows: $x'_i = (x_i \oplus s_{\pi(i)}) \cdot (1 - 2t_{\pi(i)}) + t_{\pi(i)}$. In fact, the six elements of $\mathbb{F}_2 \times \mathbb{F}_3$ fall in three categories, depending on their effect on x_i:

- (Identity) $x'_i = x_i$. This happens whenever $s_{\pi(i)} = t_{\pi(i)}$.
- (Flip) $x'_i = 1 \oplus x_i$. This happens whenever $t_{\pi(i)} \in \{0,1\} \wedge s_{\pi(i)} \neq t_{\pi(i)}$.
- (Constant 2) $x'_i = 2$. This happens whenever $t_{\pi(i)} = 2$.

Therefore, the prover choice of $(\mathbf{s},\mathbf{t})$ boils down to choosing a sequence of $(\mathsf{copy}, \mathsf{flip}, \mathsf{const2})$ operators, which are randomly shuffled, and then applied to each bit of the witness x. We formulate the experiment as a balls-into-bins experiment: the witness x is seen as a sequence of K bins, where the i-th bin is labeled by x_i. The prover chooses K balls, where each ball represents an operator (we call type-A, type-B, and type-C the $\mathsf{copy}, \mathsf{flip}$, and $\mathsf{const2}$ operators respectively). Then, the balls are randomly thrown into the bins (with exactly one ball per bin), and the label of each bin is changed according to the operator of the ball it receives. The prover wins if, in the end, the sum of the labels in each block of bins is 1 modulo 3 (corresponding to checking that each block of x' has Hamming weight equal to 1 modulo 3).

Our analysis distinguishes two situations, depending on the balls chosen by the prover: either at least 60% of the balls are of the same type (we say that this type *dominates* the balls – the choice of the threshold is somewhat arbitrary), or the types are *well-spread* (no type appears more than 60% of the time). Intuitively, these cases correspond to two different 'failure modes':

- **Dominant Scenario.** Here, the prover's best choice is to pick x 'very close' to a valid witness (say, with a single incorrect block), and to set almost all balls to be type-A balls (type-A is what a honest prover would pick). Then, a few type-B balls are inserted, and the prover hopes that the permutation puts the type-B balls exactly within the incorrect blocks of x (hence correcting them). Alternatively, the prover could also pick x to be close to an 'anti-valid' witness (*i.e.* a valid witness with all its bits flipped) and set almost all balls to be type-B balls, to the same effect. In any case, bounding this scenario is done by bounding the probability that the incorrect blocks of x receive balls of the dominant types.

- **Well-Spread Scenario.** In the well-spread scenario, each bin receive a ball taken randomly from the initial set of balls. Since it is well-spread, this implies that the label of each bin is mapped to an element of $\{0, 1, 2\}$, with a well-spread probability mass on each of the options. For the prover to win, sufficiently many of the labels must be correctly set (so that all blocks have labels summing to 1 mod 3). If the random variables for each label were independent, a Chernoff bound would show that this happens with very low probability. While the labels are not independent from each other, a little bit of work allows to reduce the problem to bounding a hypergeometric distribution, for which strong Chernoff-style bounds exist (see Lemma 9).

To bound the dominant scenario, we use a (slightly involved) counting argument, enumerating the total number of winning configurations for the prover, for each choice of (1) the number of incorrect blocks in x (denoted ℓ), and (2) the number of balls of the dominant type (denoted θ), and divide it by the total number of configurations. For each choice of (ℓ, θ), this provides an explicit (albeit complex) formula for the bound. We conjecture that the best choice of ℓ, θ is to set $\ell = 1$ and $\theta = K - 1$ (*i.e.*, using a witness with a single incorrect block). Though we do not have a proof of this statement, we can still compute a bound explicitly by minimizing the formula over all possible choices of ℓ and θ. When picking concrete parameters, we use a Python script to compute the bound explicitly, given in Appendix A of the Supplementary Material (we note that the output of the script confirmed the conjecture for all parameters we tried).

In contrast, in the well-spread scenario, our analysis bounds p using a Chernoff-style bound for hypergeometric distribution, which provides directly an explicitly and simple formula for computing p in this case. Due to the exponential decay of the bound, we observe that the well-spread scenario is in fact never advantageous for a malicious prover: the best strategy is always to set $(\mathbf{s}, \mathbf{t})$ so as to be in the dominant scenario.

Allowing almost-regular witnesses. A careful reader might have noticed an apparent issue in our previous analysis: assume that a cheating prover uses an *antiregular* witness x (*i.e.*, a vector such that $x \oplus \mathbf{1}$ is regular), and only type-B balls (*i.e.* the pairs (s_i, t_i) are such that $s_i = 1 - t_i$). Then it passes the verifier checks exacty as an honest prover would: the antiregular vector x still has blocks of odds Hamming weight, and for any choice of π, x' is now equal to $\mathbf{1} \oplus x$: that is, a regular vector. Concretely, this means that our zero-knowledge proof is not a proof of knowledge of a regular solution, but rather a proof of knowledge of either a regular *or an antiregular* solution. Nevertheless, when building a signature scheme, this is not an issue: it simply implies that unforgeability relies instead on the hardness of finding a regular or antiregular solution to an RSD instance. But it is a folklore observation that this variant of RSD does in fact reduce to the standard RSD problem, with only a factor 2 loss in the success probability, hence this does not harm security.

In fact, we push this approach even further. The bound p which we obtain by the previous analysis is essentially tight, but remains relatively high for our purpose. Concretely, fixing a value of $K \approx 1500$ (this is roughly to the range of our parameter choices), we get $\mathsf{p} \approx 1/250$. This bound is met when the prover uses a witness which is regular almost everywhere, with at most one exceptional block, where it has Hamming weight 3 or 5 (or the antiregular version of that). In this case, the prover builds $(\mathbf{s}, \mathbf{t})$ honestly, except on a single position (s_i, t_i), where he sets $s_i = 1 - t_i$. Then, with probability 1/250, the permutation aligns i with the unique faulty block (there are 250 blocks in total), and the (s_i, t_i) pair "corrects" the faulty block, passing the verifier checks. Even though a 1/250 bound is not too bad, in our context it largely dominates the soundness error of the proof. This stems from the fact that our protocol has extremely low computational costs, hence we can freely set the number n of virtual parties much higher than in previous works, e.g. $n = 1024$ or $n = 2048$, while still achieving comparable computational costs. In this high-n setting, the hope is to achieve a soundness error close to the best possible value of $1/n$, in order to minimize the number of parallel repetitions (hence reducing communication).

To get around this limitation, we choose to *allow almost-regular witnesses* (or almost-antiregular witnesses). Concretely, we relax the soundness of our zero-knowledge proof to guarantee only that a successful cheating prover must at least know an almost-regular (or almost-antiregular) witness, *i.e.*, a witness whose blocks all have weight 1 except one, which might have weight 1, 3, or 5. This form of zero-knowledge proof with a gap between the language of honest witnesses and the language of extracted witnesses is not uncommon in the literature. In particular, it is similar in spirit to the notion of *soundness slack* in some lattice-based zero-knowledge proofs, where a witness is a vector with small entries, and an extracted witness can have much larger entries [BDLN16, CDXY16]. By using this relaxation, our bound p improves by (essentially) a quadratic factor: a cheating prover must now cheat on (at least) *two* positions (s_i, t_i), and hope that both align with the (at least) two incorrect blocks of x. Concretely, using $K \approx 1500$, our combinatorial analysis gives $\mathsf{p} \approx 3 \cdot 10^{-5}$ in this setting, which becomes a vanishing component of the soundness error (dominated by the $1/n$ term).

When building a signature scheme from this relaxed zero-knowledge proof, we use the Fiat-Shamir transform on a 5-round protocol, and must therefore adjust the number of repetitions to account for the attack of [KZ20]. For a bound of p as above, this severely harms efficiency. Following the strategy of [FJR22a], we avoid the problem by making p much smaller. Concretely, denoting τ_{ZK} the smallest integer such that $(1/n + \mathsf{p} \cdot (1 - 1/n))^{\tau_{\mathsf{ZK}}} \leq 2^{-\lambda}$, the optimal number of repetitions which one can hope for in the signature scheme is $\tau = \tau_{\mathsf{ZK}} + 1$. Therefore, denoting f the number of faulty blocks in the witness, we set f to be the smallest value such that the resulting bound p yields $\tau = \tau_{\mathsf{ZK}} + 1$, hence achieving the optimal number of repetitions. At this stage, the unforgeability of the signature now reduces to the hardness of finding either an almost-regular or an almost-antiregular solution to an RSD problem (with up to f faulty block), which seems quite exotic (though it remains in itself a plausible assumption). For the sake of relying only on the well-established RSD assumption, we set parameters such that, except with $2^{-\lambda}$ probability, a random RSD instance does not in fact have any almost-regular or almost-antiregular solution (with up to f faulty blocks) on top of the original solution. This implies that, for this choice of parameters, this "f-almost-RSD" assumption is in fact equivalent to the RSD assumption (with essentially no loss in the reduction).

Summing up. We first describe and construct an optimized zero-knowledge argument of knowledge, following the template outlined in this technical overview. We compile our new zero-knowledge proof into a signature using Fiat-Shamir. We use the combinatorial analysis to identify a bound p on the probability that the verifier picks a *bad permutation*, and formally prove that the zero-knowledge proof achieves ε-soundness, where $\varepsilon = (1/n + \mathsf{p} \cdot (1 - 1/n))$ (n being the number of parties in the MPC protocol). To achieve optimal efficiency for the signature scheme, we reduce p by allowing up to f faulty blocks in the witness, and select RSD parameters such that the underlying assumptions remains the standard RSD assumption despite this relaxation of the proof soundness. The full combinatorial analysis of our construction (the bound p) is a core technical contribution of our work, and is the focus of Section 6.

3.4 Cryptanalysis of RSD

We complement our analysis by providing an overview of the security of RSD. In particular, we give a precise picture of how RSD relates to the standard syndrome decoding assumption, depending on the parameters (K, k, w). Concretely, we define a "RSD uniqueness bound", analogous to the Gilbert-Varshamov (GV) bound for standard syndrome decoding, and show that (1) above the GV bound, RSD is always easier than SD; (2) below the RSD uniqueness bound, RSD becomes in fact *harder* than SD, and (3) in between the two bounds is a gray zone, where the hardness of the two problems is not directly comparable. Looking ahead, our choice of parameters lies inside this gray zone, and corresponds to a setting where a random RSD instance does not have additional f-almost-regular solutions with high probability, to guarantee a tight reduction to the standard RSD assumption even when allowing such relaxed solutions.

We also overview existing attacks on RSD, and in most cases revisit and improve them to exploit more carefully the structure of the RSD problem, obtaining significant speedups. Eventually, we design a new attack which outperforms all previous attacks. Our attack is not fully explicit: it requires an approximate birthday paradox search (*i.e.*, finding an almost-collision between items of two lists). For the sake of being conservative, when choosing concrete parameters, we assume that this approximate birthday paradox can be solved in time linear in the list size (it is far from clear how to perform such a fast approximate collision search, but it does not seem implausible that it can be done, hence we choose to stay on the safe side. We view finding such an algorithm as an interesting open problem). The details on our analysis of the RSD problem is the focus of Section 8.

4 Zero-Knowledge Proof for Regular Syndrome Decoding

4.1 Optimizations

We start from the template given in the Technical Overview (Section 3), and refine it using various optimizations. Some of these optimizations are standard, used *e.g.* in works such as [KKW18,BdK+21, FJR22a] (we present them as such when it is the case), and others are new, tailored optimizations.

Using a collision-resistant hash function. The "hash trick" is a standard approach to reduce the communication of public coin zero-knowledge proofs. It builds upon the following observation: in a zero-knowledge proof, the verification equation on a list of messages $(m_1, \cdots, m_\ell)$ often makes the messages *reverse samplable*: the verifier can use the equation to recover what the value of $(m_1, \cdots, m_\ell)$ should be. Whenever this is the case, the communication can be reduced by sending $h = H(m_1, \cdots, m_\ell)$ instead of $(m_1, \cdots, m_\ell)$, where H denotes collision-resistant hash function. The verification proceeds by reconstructing $(m_1, \cdots, m_\ell)$ and checking that $h = H(m_1, \cdots, m_\ell)$, and security follows from the collision resistance of H. As h can be as small as 2λ-bit long, this significantly reduces communication.

Using regular syndrome decoding in systematic form. Without loss of generality, we can set H to be in systematic form, *i.e.* setting $H = [H'|I_k]$, where I_k denotes the identity matrix over $\{0,1\}^{k\times k}$. This strategy was used in the recent code-based signature of [FJR22a]. Using H in systematic form, and writing x as $x = (x_1|x_2)$ where $x_1 \in \mathbb{F}_2^{K-k}, x_2 \in \mathbb{F}_2^k$, we have $Hx = H'x_1 + x_2 = y$. Since the instance (H, y) is public, this implies that the prover need not share x entirely over $\mathbb{F}_2$: it suffices for the prover to share x_1, and all parties can reconstruct $[\![x_2]\!]_2 \leftarrow y \oplus H' \cdot [\![x_1]\!]_2$. Additionally, the parties need not opening $\mathbf{z}$ entirely: denoting $\pi(\mathbf{r}) = (r_1|r_2)$, the parties can open instead $[\![z_1]\!]_2 = [\![x_1 \oplus r_1]\!]_2$ and define $z_2 = H'z_1 \oplus y$. This way, they can reconstruct the complete $\mathbf{z}$ as $\mathbf{z} = (z_1|z_2)$. The rest of the protocol proceeds as before. Following the above considerations, and to simplify notations, from now on we refer to the short vector of length $K-k$ in the small field (previously indicated with x_1) simply as x and to the long vector of size K in the large field as $\tilde{x}$ (*i.e.* $\tilde{x} = (x|H'x \oplus y)$).

Exploiting the regular structure of x. We further reduce the size of x using an optimization tailored to the RSD setting. Thanks to its regular structure, we can divide x into w blocks each of size $T = K/w$. But since each block has exactly one non-zero entry, given the first $T-1$ entries $(b_1, \cdots, b_{T-1})$ of any block, the last entry can be recomputed as $b_T = 1 \oplus \bigoplus_{i=1}^{T-1} b_i$. In the zero-knowledge proof, the prover does therefore only share $T-1$ out of the T bits in each block of x among the virtual parties. Similarly, the size of r_1 and z_1 are reduced by the same factor, since only $T-1$ bits of each block need to be masked.

Reducing the size of the messages. With the above optimizations, the equation $H\tilde{x} = H'x \oplus x_2 = y$ needs not be verified anymore: it now holds by construction, as $\tilde{x}$ is defined as $(x|H'x \oplus y)$. This removes the need to include $\mathbf{y}_i$ in the messages msg_i sent by each party P_i; this is in line with previous works, which also observed that linear operations are for free with proper optimizations. The message of each party becomes simply $\mathsf{msg}_i = (\mathbf{z}_i, \mathbf{v}'_i)$. Note that in this concrete instantiation the entries of $\mathbf{v}'_i$ are computed as $\langle \mathbf{1}, \tilde{\mathbf{x}}_\mathbf{i}{}^j \rangle$, where the vectors $\tilde{\mathbf{x}}_\mathbf{i}{}^j$ for $j = 1$ to w are the blocks of P_i's share of the vector $\mathbf{z} \odot (\mathbf{1} - \pi(\mathbf{t})) + (\mathbf{1} - \mathbf{z}) \odot \pi(\mathbf{t})$.

Using the Chinese remainder theorem. In the zero-knowledge proof, verifying any linear equation modulo 2 on the witness $\bar{\mathbf{x}}$ is for free communication-wise. Ultimately, the verifier wants to check that $\langle \bar{\mathbf{x}}^j, \mathbf{1} \rangle = 1 \bmod T$. Setting T to be equal to 2 modulo 4 guarantees that T is even, and $\gcd(T/2, 2) = 1$. Hence, it suffices to work over the integer ring $\mathbb{Z}_{T/2}$ instead of $\mathbb{Z}_T$, to let the verifier check the equation $\langle \bar{\mathbf{x}}^j, \mathbf{1} \rangle = 1 \bmod T/2$ for $j = 1$ to w. Indeed, by the Chinese remainder theorem, together with the relations $\langle \bar{\mathbf{x}}^j, \mathbf{1} \rangle = 1 \bmod 2$ (which can be checked for free), this ensures that $\langle \bar{\mathbf{x}}^j, \mathbf{1} \rangle = 1 \bmod T$ for $j = 1$ to w. This reduces the size of the $\mathbf{t}_i$ vectors from $K \cdot \log T$ to $K \cdot \log(T/2)$. As outlined in Section 3.2, we actually set $T = 6$ in our concrete instantiation, hence executing the protocol over the integer ring $\mathbb{Z}_3$, the smallest possible ring satisfying the coprimality constraint.

Compressing share with PRG. Another standard technique from [KKW18] uses a pseudorandom generator to compress all-but-one shares distributed during the input sharing and preprocessing phases. Indeed, writing $[\![\mathbf{x}]\!]_2 = (\mathbf{x_1}, \ldots, \mathbf{x_n})$, then $\mathbf{x_n} = \mathbf{x} - \bigoplus_{i=1}^{n-1} \mathbf{x_i} \bmod 2$. Denoting also $[\![\mathbf{r}]\!]_2 = (\mathbf{s_1}, \ldots, \mathbf{s_n})$ and $[\![\mathbf{r}]\!]_T = (\mathbf{t_1}, \ldots, \mathbf{t_n})$, it holds that $\sum_{i=1}^{n} \mathbf{t_i} = \bigoplus_{i=1}^{n} \mathbf{s_i} \bmod T$, which rewrites to $\mathbf{t}_n = \bigoplus_{i=1}^{n} \mathbf{s_i} - \sum_{i=1}^{n-1} \mathbf{t_i} \bmod T$.

We can compress the description of these shares by giving to each party P_i a $\lambda-$bit seed seed_i and letting each of them apply a pseudorandom generator to seed_i in order to obtain (pseudo)random shares $\mathbf{x_i}$, $\mathbf{s_i}$ and $\mathbf{t_i}$. All shares of $\mathbf{s}$ can be compressed this way (since $\mathbf{s}$ need just be a random vector), and all-but-one shares of $[\![\mathbf{x}]\!]_2$ and $\mathbf{t}$. The missing shares can be obtained by letting P_n receive an auxiliary string aux_n defined as:

$$\mathsf{aux}_n \leftarrow \left(\mathbf{x} - \bigoplus_{i=1}^{n-1} \mathbf{x_i} \bmod 2, \bigoplus_{i=1}^{n} \mathbf{s_i} - \sum_{i=1}^{n-1} \mathbf{t_i} \bmod T \right).$$

We refer to the information shared with each party as the *state* of the party. For each P_i for $1 \leq i \leq n-1$, we therefore have $\mathsf{state}_i = \mathsf{seed}_i$. The last party P_n has $\mathsf{state}_n = (\mathsf{seed}_n | \mathsf{aux}_n)$: in the online phase of the protocol each party seed_n can be used to randomly generate $\mathbf{s_n}$.

Generating the seeds from a puncturable pseudorandom function. To further reduce the overhead of communicating the seeds, we apply the standard optimization of generating all seeds from a puncturable PRF. A *puncturable pseudorandom function* (PPRF) is a PRF F such that given an input x, and a PRF key k, one can generate a *punctured* key, denoted $k\{x\}$, which allows evaluating F at every point except for x, and does not reveal any information about the value $F.\mathsf{Eval}(k, x)$. PPRFs have been introduced in [KPTZ13, BW13, BGI14]. The seminal GGM PPRF [GGM86] is a PPRF which has been used in multiple contexts to compress many seeds into a short seed, such that one can succinctly open all-but-one seeds. In particular, it was first used in the context of MPC-in-the-head signatures in [KKW18]. Concretely, we introduced a master seed seed^* from which we generate n minor seeds $\mathsf{seed}_1, \cdots, \mathsf{seed}_n$ as the leaves of a binary tree of depth $\log n$, where the two children of each nodes are computed using a length-doubling pseudorandom generators. This way, revealing all seeds except seed_j requires only sending the seeds on the nodes along the co-path from the root to the j-th leave, which reduces the communication from $\lambda \cdot (n-1)$ to $\lambda \cdot \log n$. Note that due to this optimization, when compiling the proof into a signature, collisions among seed^* for different signatures are likely to appear after $2^{\lambda/2}$ signatures. To avoid this issue, an additional random salt of length 2λ must be use, see Section 7.

Using deterministic commitments. As in [KKW18] and other previous works, we observe that all committed values are pseudorandom. Therefore, the commitment scheme does not have to be hiding: in the random oracle model, it suffices to instantiate $\mathsf{Commit}(x; r)$ deterministically as $H(x)$ for zero-knowledge to hold.

Final Zero Knowledge protocol. We represent on Figure 3 our final zero-knowledge proof of knowledge for a solution to the regular syndrome decoding problem, taking into account all the optimizations outlined above, except the use of deterministic commitments (using deterministic commitments requires using the ROM, which is otherwise not needed for the zero-knowledge proof. Looking ahead, we still use this optimization when compiling the proof to a signature using Fiat-Shamir, since we use the ROM at this stage anyway).

4.2 Security Analysis

We now turn to the security analysis of the protocol. A crucial component of our analysis is a *combinatorial bound*, which we introduce below. Before, we state some definition.

Definition 4 (f-Strongly invalid candidate witness). *We say that $x \in \mathbb{F}_2^K$ is a f-*weakly valid witness *if x is* almost *a regular vector (in the sense that it differs from a regular vector in at most f blocks), or almost an* antiregular *vector. Formally, let $(x^j)_{j \leq w}$ be the w length-K/w blocks of x. Assume that K/w is even. Then x is an f-weakly valid witness if*

Inputs: The prover and the verifier have a matrix $H \in \mathbb{F}_2^{k \times K} = [H'|I_k]$ and a vector $y \in \mathbb{F}_2^k$. The prover also knows a regular vector $\tilde{x} = (x|x_2) \in \mathbb{F}_2^K$ with Hamming weight $\mathsf{HW}(\tilde{x}) = w$ and such that $H\tilde{x} = y$.
Parameters and notations. We let n denote number of parties. We let x' denote the vector obtained by deleting the T-th bit in each block of x. We call "expanding" the action of recomputing x from x', *i.e.* adding a T-th bit at the end of each length-$(T-1)$ block, computed as the opposite of the XOR of all bits of the block.
Round 1 The prover emulates the preprocessing phase of Π as follows:

1. Chooses a random seed seed^*;
2. Uses seed^* as the root of a depth-$\log n$ full binary tree to produce the leaves $(\mathsf{seed}_i, \sigma_i)$ using a length-doubling PRG for each $i \in [n]$;
3. Use $(\mathsf{seed}_1, \cdots, \mathsf{seed}_{n-1})$ to create pseudorandom shares $(\mathbf{x'_1}, \cdots, \mathbf{x'_{n-1}})$ of x', as well as vectors $(\mathbf{s_i}, \mathbf{t_i}) \in \mathbb{F}_2^{(T-1)\cdot(K-k)/T} \times \mathbb{Z}_{T/2}^K$. Use seed_n to create $\mathbf{s_n}$ as well. Let $\mathbf{x_i}$ denote the vector obtained by "expanding" $\mathbf{x'_i}$ to $K-k$ bits;
4. Let $\mathbf{s'_i}$ denote the value obtained by "expanding" $\mathbf{s_i}$ to a $(K-k)$-bit vector, and let $s' \leftarrow \bigoplus_{i=1}^n \mathbf{s'_i}$. Set $s \leftarrow (s'|H' \cdot s' \oplus y)$. Define
$$\mathsf{aux}_n \leftarrow \left(x' \oplus \bigoplus_{i=1}^{n-1} \mathbf{x_i}, s' - \sum_{i=1}^{n-1} \mathbf{t_i} \bmod T/2 \right);$$
5. Sets $\mathsf{state}_i = \mathsf{seed}_i$ for $1 \leq i \leq n-1$ and $\mathsf{state}_n = \mathsf{seed}_n || \mathsf{aux}_n$;
6. For each $i \in [n]$ computes $\mathsf{com}_i := \mathsf{Commit}(\mathsf{state}_i, \sigma_i)$;
7. Computes $h := H(\mathsf{com}_1, \cdots, \mathsf{com}_n)$ and sends it to the verifier.

Round 2 The verifier chooses a permutation $\pi \in S_{K-k}$ and sends it to the prover.
Round 3 The prover:

1. Simulates the online phase of the n parties protocol Π using the pairs $(\pi(\mathbf{s_i}), \pi(\mathbf{t_i}))$ as the preprocessing material of the i-th party:
   - for each $i \in [n]$ compute $\mathbf{z'_i} = \mathbf{x'_i} \oplus \pi(\mathbf{s_i})$ getting $[\![z_1]\!]_2 = (\mathbf{z_1}, \cdots, \mathbf{z_n})$ by "expanding" the $\mathbf{z'_i}$'s;
   - Define $z_2 = H' \cdot z_1 \oplus y$ and $z = (z_1|z_2)$;
   - Set $[\![\bar{x}]\!]_{T/2} = (\bar{\mathbf{x_1}}, \ldots, \bar{\mathbf{x_n}})$ where $\bar{\mathbf{x_i}} = z + (1-2z) \odot \pi(\mathbf{t_i}) \bmod T/2$;
   - For each $i \in [n]$ compute:
     - $\bar{\mathbf{w}}_\mathbf{i}^\mathbf{j} = \mathsf{HW}(\bar{\mathbf{x_i}}^j) \bmod T/2$ for all the blocks $1 \leq j \leq w$;
     - $\mathsf{msg}_i = (\mathbf{z_i}, (\bar{\mathbf{w}}_\mathbf{i}^\mathbf{j})_{1 \leq j \leq w})$;
2. Compute $h' = H(\mathsf{msg}_1, \cdots, \mathsf{msg}_n)$;
3. Send z_1 and h' to the verifier. // sending z'_1 actually suffices

Round 4 The verifier chooses a challenge $d \in [n]$ and sends it to the prover.
Round 5 The prover sends $(\mathsf{state}_i, \sigma_i)_{i \neq d}$ and com_d.
Verification The verifier checks that everything is correct:

1. Recompute $\bar{\mathsf{com}}_j = \mathsf{Commit}(\mathsf{state}_j, \sigma_j)$ for $j \neq d$;
2. Recompute msg_i for all $i \neq d$ using state_i and z_1;
3. Recompute
$$\bar{\mathsf{msg}}_d = \left(z_1 - \sum_{i \neq d} \mathbf{z_i}, \left(1 - \sum_{i \neq d} \bar{\mathbf{w}}_\mathbf{i}^\mathbf{j} \right)_{1 \leq j \leq w} \right);$$
4. Check if $h = H(\bar{\mathsf{com}}_1, \cdots, \mathsf{com}_d, \cdots, \bar{\mathsf{com}}_n)$;
5. Check if $h' = H(\mathsf{msg}_1, \cdots, \bar{\mathsf{msg}}_d, \cdots, \mathsf{msg}_n)$.

Fig. 3. A five-round zero-knowledge proof of knowledge of a solution to the regular syndrome decoding problem

1. $\forall j \leq w$, $\mathsf{HW}(x^j) = 1 \bmod 2$, *and*
2. $|\{j : \mathsf{HW}(x^j) \neq 1\}| \leq f$ or $|\{j : \mathsf{HW}((\mathbf{1} \oplus x)^j) \neq 1\}| \leq f$,

where $\mathbf{1} \oplus x$ *is the vector obtained by flipping all bits of* x*. If* x *is not an* f*-weakly valid witness, we say that* x *is an* f-strongly invalid *candidate witness.*

Below, set $T \leftarrow K/w$. We assume for simplicity that the parameters are such that w divides K, and that $T = 2 \bmod 4$. Note that this ensures that a block x^j of the candidate witness x has Hamming weight 1 if and only if $\sum_{i=1}^{K/w} x_i^j = 1 \bmod T/2$ and $\sum_{i=1}^{K/w} x_i^j = 1 \bmod 2$.

Definition 5 (Combinatorial Bound). *Given a vector* $u \in \mathbb{N}^K$ *divided into* w *length-*K/w *blocks* u^j*, we denote* $\mathsf{Succ}(u)$ *the event that* $\sum_{i=1}^{K/w} u_i^j = 1 \bmod T/2$ *for all* $j \leq w$*. Then, a* combinatorial bound *for the zero-knowledge proof of Figure 3 with parameters* (K, w) *is a real* $\mathsf{p} = \mathsf{p}(K, w, f) \in (0, 1)$ *such that for* any f*-strongly invalid candidate witness* $x \in \mathbb{F}_2^K$ *satisfying* $\forall j \leq w$, $\mathsf{HW}(x^j) = 1 \bmod 2$ *(*i.e. *x still satisfies condition* 1 *of Definition 4), and for* any *pair of vectors* $(s, t) \in \mathbb{F}_2^K \times \mathbb{Z}_{T/2}^K$,

$$\Pr[\pi \leftarrow_r \mathsf{Perm}_K, x' \leftarrow \pi(t) + (x \oplus \pi(s)) \odot (\mathbf{1} - 2\pi(t)) \ : \ \mathsf{Succ}(x')] \leq \mathsf{p}(K, w, f),$$

where Perm_K *denotes the set of all permutations of* $\{1, \cdots, K\}$*.*

Informally, the combinatorial bound p is a bound on the probability that a malicious prover passes the verification *without* guessing the subset of views requested by the verifier. The formal notion relax what we mean by a malicious prover, by requesting that they use a witness *sufficiently far* from a honest regular witness. Looking ahead, this feature allows us to obtain much smaller combinatorial bounds for concrete choices of parameters. When building a signature scheme from the zero-knowledge proof, we further prove that finding a “relaxed” witness is at least as hard as solving the standard regular syndrome decoding problem, hence justifying that this relaxation does not harm security.

Theorem 6. *Let* Commit *be a non-interactive commitment scheme, and* H *be collision-resistant hash function. Let* p *be a combinatorial bound for the protocol of Figure 3. The protocol given on Figure 3 is a* gap *honest-verifier zero-knowledge argument of knowledge for the relation* $\mathcal{R}$ *such that* $((H, y), x) \in \mathcal{R}$ *if* $H \cdot x = y \bmod 2$ *and* x *is a regular vector of weight* w*. The* gap *relation* $\mathcal{R}'$ *is such that* $((H, y), x) \in \mathcal{R}'$ *if* $H \cdot x = y \bmod 2$ *and* x *is an* f*-weakly valid witness. The soundness error of the proof is at most* $\varepsilon = \mathsf{p} + 1/n - \mathsf{p}/n$*.*

The completeness of the protocol naturally derives from its definition. In Section 5, we prove the honest-verifier zero-knowledge and soundness properties.

4.3 Communication

The expected communication of the zero-knowledge argument amounts to:

$$4\lambda + \tau \cdot \left(\lambda(\log n + 1) + \left(\frac{2n-1}{n} \right) \frac{T-1}{T} (K - k) + \left(\frac{n-1}{n} \right) K \log_2 T \right) \text{ bits,}$$

where we assume that hashes are 2λ bits long, and commitments are λ bits long, and where τ denotes the number of parallel repetitions of the proof.

5 Proof of Theorem 6

In this section, we prove Theorem 6.

5.1 Honest-verifier zero-knowledge (HVZK)

HVZK follows from semi-honest security of the n-parties protocol Π. Given a simulator Sim_Π for Π, we can construct an honest-verifier zero-knowledge simulator for our protocol as follows:

- Run Sim_Π using the permutation π (which can be computed from the random coins of the adversary V against Π) to simulate the views of parties $P_{i \neq d}$ in order to obtain $\mathsf{state}_{i \neq d}$ and z. From these informations the simulator computes all the message msg_i, and so $h' = H(\mathsf{msg}_1, \cdots, \mathsf{msg}_n)$, as an honest prover would;

- For all $i \neq d$ chooses uniform σ_i and computes $\mathsf{com}_i = \mathsf{Commit}(\mathsf{state}_i, \sigma_i)$ as an honest prover would. Moreover, computes com_d as a commitment to a 0-string. Hence the simulator can compute $h = H(\mathsf{com}_1, \cdots, \mathsf{com}_n)$;
- The simulator outputs h, h', z and $(\mathsf{state}_i, \sigma_i)_{i \neq d}$, msg_d.

A straightforward hybrid argument shows that transcripts output by the simulator are computationally indistinguishable from transcripts of real executions of the protocol with an honest verifier.

5.2 Proof of Soundness

Let $\tilde{\mathsf{P}}$ be a prover which manages to generate an accepting proof with probability $\tilde{\varepsilon} > \varepsilon$. We exhibit an *extractor* which finds a witness x such that $H \cdot x = y$, where x is guaranteed to be a *weakly valid* witness (see Definition 4). Let R denote the randomness used by $\tilde{\mathsf{P}}$ to generate the commitment h of the first round, and by r a possible realization of R. Let $\mathsf{Succ}_{\tilde{\mathsf{P}}}$ denote the event that $\tilde{\mathsf{P}}$ succeeds in convincing V. By hypothesis

$$\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}] = \tilde{\varepsilon} > \varepsilon = \mathsf{p} + \frac{1}{n} - \frac{\mathsf{p}}{n}.$$

Let us fix an arbitrary value $\alpha \in \{0, 1\}$ such that $(1 - \alpha)\tilde{\varepsilon} > \varepsilon$, which exists since $\tilde{\varepsilon} > \varepsilon$. We say that a realization r of the prover randomness for the first flow is *good* if it holds that

$$\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}} | R = r] \geq (1 - \alpha)\tilde{\varepsilon}.$$

Furthermore, by the Splitting Lemma (see e.g. [FJR22a]), we have $\Pr[R \text{ good} | \mathsf{Succ}_{\tilde{\mathsf{P}}}] \geq \alpha$. Assume now that T_0 is the transcript of a successfull execution of the zero-knowledge proof with $\tilde{\mathsf{P}}$. Let r denote the random coin used by $\tilde{\mathsf{P}}$ in the first round, and let d_0 denote the fourth-round message of the verifier. If r is *good*, then

$$\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}} | R = r] \geq (1 - \alpha)\tilde{\varepsilon} > \varepsilon > \frac{1}{n},$$

which implies that there necessarily exists a second successful transcript T_1 with a different fourth-round message $d_1 \neq d_0$. As we will demonstrate afterwards, given (T_0, T_1), it is possible to extract a unique well-defined triplet (x, s, t) (where x is a candidate witness, and (s, t) is the preprocessing material used by $\tilde{\mathsf{P}}$) consistent with both transcripts.

Consistency of (T_0, T_1). Let (π_0, d_0) and (π_1, d_1) be the verifier challenges in the successful transcripts T_0 and T_1 respectively, with $d_0 \neq d_1$. Let us denote $(\mathsf{state}'_{i \neq d_0}, \sigma'_{i \neq d_0}, \mathsf{com}_{d_0})$ and $(\mathsf{state}_{i \neq d_1}, \sigma_{i \neq d_1}, \mathsf{com}_{d_1})$ the rest of the transcripts T_0 and T_1 respectively. Suppose that $\exists i \in [n] \backslash \{d_0, d_1\}$ such that $(\mathsf{state}_i, \sigma_i) \neq (\mathsf{state}'_i, \sigma'_i)$. Then there are two possibilities:

- The committed values are different:

$$\mathsf{com}_i = \mathsf{Commit}(\mathsf{state}_i, \sigma_i) \neq \mathsf{Commit}(\mathsf{state}'_i, \sigma'_i) = \mathsf{com}'_i.$$

  But since both transcripts from such states are supposed to be accepted, this implies that in particular, we have $h = H(\mathsf{com}_1, \cdots, \mathsf{com}_n)$ and $h = H(\mathsf{com}'_1, \cdots, \mathsf{com}'_n)$ which contradicts the collision resistance of H.
- The committed values are equal:

$$\mathsf{com}_i = \mathsf{Commit}(\mathsf{state}_i, \sigma_i) = \mathsf{Commit}(\mathsf{state}'_i, \sigma'_i) = \mathsf{com}'_i.$$

  This directly contradicts the binding property of Commit.

Therefore, the states and the randomness are necessarily mutually consistent (that is $\mathsf{state}'_{i \neq d_0, d_1} = \mathsf{state}_{i \neq d_0, d_1}$ and $\sigma'_{i \neq d_0, d_1} = \sigma_{i \neq d_0, d_1}$). Since $d_0 \neq d_1$, they jointly define a unique tuple $(\mathsf{state}_i, \sigma_i)_{i \in [n]}$, from which we can recompute $[\![x]\!]_2 = (\mathbf{x_1}, \ldots, \mathbf{x}_n)$ and $[\![s]\!]_2 = (\mathbf{s_1}, \ldots, \mathbf{s_n})$, $[\![t]\!]_p = (\mathbf{t_1}, \ldots, \mathbf{t_n})$.

The witness $\boldsymbol{x}$ is a valid witness. Now we show that if x is a strongly invalid witness, then $\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R = r] \leq \varepsilon$, contradicting our assumption that r is good. Let us denote $\mathsf{BadPerm}$ the event (defined over the random choice of a permutation π, and for the fixed value of (x, s, t)) that in each block of x', the entries of the blocks sum to 1 modulo $T/2$ (that is, the event $\mathsf{Succ}(x')$), where $x' = \pi(t) + (x \oplus \pi(s)) \odot (\mathbf{1} - 2\pi(t))$. By construction, the extraction procedure guarantees that the extracted candidate witness x has blocks of odd Hamming weight. Therefore, by definition of the combinatorial bound (Definition 5), we have $\Pr[\mathsf{BadPerm}] \leq \mathsf{p}$. Now,

$$\begin{aligned}\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R = r] &= \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}} \ \wedge \ \mathsf{BadPerm}|R = r] + \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}} \ \wedge \ \neg\mathsf{BadPerm}|R = r] \\ &\leq \mathsf{p} + (1 - \mathsf{p}) \cdot \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R = r \ \wedge \ \neg\mathsf{BadPerm}].\end{aligned}$$

We now bound $\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R = r \ \wedge \ \neg\mathsf{BadPerm}]$. Assume for the sake of contradiction that $\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R = r \ \wedge \ \neg\mathsf{BadPerm}] > 1/n$. As before, this implies that given a successful transcript T'_0 with fourth round d'_0, there necessarily exists a second successful transcript T'_1 with the same first three rounds, and a fourth round $d'_1 \neq d'_0$.

By the same argument as above, T'_0 and T'_1 are necessarily consistent, and uniquely define a tuple $(\mathsf{state}'_i, \sigma'_i)_{i\in[n]}$. Furthermore, since we condition on $R = r$, meaning that the first flow h' is the same as the first flow h in T_0, T_1, it must holds that $(\mathsf{state}'_i, \sigma'_i)_{i\in[n]} = (\mathsf{state}_i, \sigma_i)_{i\in[n]}$, the states and random coins uniquely defined by (T_0, T_1) (if this is not the case, we obtain either a contradiction to the collision-resistance of H or to the binding of Commit, as already shown).

Now, given T'_0, reconstruct the messages: $\mathsf{msg}_i = (\mathbf{z}_i, (\bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{i}})_{j\leq w})$ is computed as $\mathbf{z}_i \leftarrow \mathbf{x}_i \oplus \pi(\mathbf{s}_i)$ and $\bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{i}} = \langle 1, \bar{\mathbf{x}}_{\mathbf{i}}{}^j\rangle$. In these equations, $\mathbf{x}_i, \mathbf{s}_i$ are stretched from seed_i, and $\bar{\mathbf{x}}_{\mathbf{i}}$ is computed as $z + (1 - 2z) \odot \pi(\mathbf{t}_i)$, with $z = (z_1|H'z_1 \oplus y)$ and $\mathbf{t}_i$ stretched from seed_i (or computed from aux_n if $i = n$). Define $\mathbf{z}_{d'_0} \leftarrow z_1 - \sum_{i\neq d'_0} \mathbf{z}_i$ and $\bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{d'_0}} \leftarrow 1 - \sum_{i\neq d'_0} \bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{i}}$ for $j = 1$ to w. The remaining tuple, $\mathsf{m\bar{s}g}_{d'_0}$, is computed as $(\mathbf{z}_{d'_0}, (\bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{d'_0}})_{j\leq w})$.

Because T'_0 and T'_1 are consistent, all messages msg_i for $i \notin \{d'_0, d'_1\}$ reconstructed from T'_1 must be identical to those reconstructed from T'_0. However, the value $\mathsf{m\bar{s}g}_{d'_0}$ is necessarily distinct from the value $\mathsf{msg}_{d'_0}$ reconstructed from T'_1. Indeed, if this was not the case, since $\bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{d'_0}} = 1 - \sum_{i\neq d'_0} \bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{i}}$, this would imply that the values $\bar{\mathbf{w}}^{\mathbf{j}}_{\mathbf{i}})_{i\leq n}$ form additive shares of 1 for $j = 1$ to w, implying that the value $\bar{x} = \bigoplus_i \bar{\mathbf{x}}_{\mathbf{i}}$ has blocks of Hamming weights all equal to 1 modulo $T/2$. But since $\bar{x} = \pi(t) + (x \oplus \pi(s)) \odot (\mathbf{1} - 2\pi(t))$ where (x, s, t) are constructed from $(\mathsf{state}'_i, \sigma'_i)_{i\in[n]} = (\mathsf{state}_i, \sigma_i)_{i\in[n]}$, this implies that we would get a contradiction to $\neg\mathsf{BadPerm}$.

Hence, we get $\mathsf{msg}_{d'_0} \neq \mathsf{m\bar{s}g}_{d'_0}$. But since T'_0, T'_1 have the same first three rounds, and in particular the same hash $h' = H(\mathsf{msg}_1, \cdots, \mathsf{m\bar{s}g}_{d'_0}, \cdots, \mathsf{msg}_n) = H(\mathsf{msg}_1, \cdots, \mathsf{msg}_{d'_0}, \cdots, \mathsf{m\bar{s}g}_{d'_1}, \cdots, \mathsf{msg}_n)$, we reached a contradiction to the collision-resistance of H. Hence, assuming the collision-resistance of H, it necessarily holds that $\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R = r \ \wedge \ \neg\mathsf{BadPerm}] \leq 1/n$. Finishing the proof:

$$\begin{aligned}\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R = r] &\leq \mathsf{p} + (1 - \mathsf{p}) \cdot \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}|R = r \ \wedge \ \neg\mathsf{BadPerm}] \\ &\leq \mathsf{p} + (1 - \mathsf{p}) \cdot \frac{1}{n} = \varepsilon,\end{aligned}$$

contradicting our assumption that r is good. Therefore, x cannot be a strongly invalid witness.

The extractor. Equipped with the above analysis, we describe an extractor $\mathcal{E}$ which is given rewindable black-box access to a prover $\tilde{\mathsf{P}}$. Define $N \leftarrow \ln(2)/((1 - \alpha)\tilde{\varepsilon} - \varepsilon)$. $\mathcal{E}$ works as follows:

- Run $\tilde{\mathsf{P}}$ and simulate a honest verifier V to get a transcript T_0. Restart until T_0 is a successful transcript.
- Do N times:
  - Run $\tilde{\mathsf{P}}$ with a honest V and the same randomness as in T_0 to get a transcript T_1.
  - If T_1 is a successful transcript with $d_0 \neq d_1$, extract the tuple (x, s, t). If x is a weakly valid witness, output x.

The end of the proof is perfectly identical to the analysis in [FJR22b, Appendix F]: given that $\mathcal{E}$ found a first successful transcript T_0, we have

$$\begin{aligned}\Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}^{T_1} \wedge d_1 \neq d_0 | R \text{ good}] &= \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}^{T_1} | R \text{ good}] - \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}^{T_1} \wedge d_1 = d_0 | R \text{ good}] \\ &\geq (1-\alpha)\tilde{\varepsilon} - 1/n \geq (1-\alpha)\tilde{\varepsilon} - \varepsilon,\end{aligned}$$

hence by definition of N, $\mathcal{E}$ gets a second successful transcript with probability at least 1/2. From there, the analysis of the expected number of calls $\mathbb{E}[\mathsf{call}]$ of $\mathcal{E}$ to $\tilde{\mathsf{P}}$ is identical to [FJR22b, Appendix F] (indeed, our extractor is identical, and the zero-knowledge proof has a similar structure):

$$\begin{aligned}&\mathbb{E}[\mathsf{call}] \leq 1 + (1 - \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}]) \cdot \mathbb{E}[\mathsf{call}] + \Pr[\mathsf{Succ}_{\tilde{\mathsf{P}}}] \cdot (N + (1-\alpha/2) \cdot \mathbb{E}[\mathsf{call}]) \\ \implies &\mathbb{E}[\mathsf{call}] \leq \frac{2}{\alpha\tilde{\varepsilon}} \cdot \left(1 + \tilde{\varepsilon} \cdot \frac{\ln(2)}{(1-\alpha)\tilde{\varepsilon} - \varepsilon}\right),\end{aligned}$$

which gives an expected number of calls $\mathsf{poly}(\lambda, (\tilde{\varepsilon}-\varepsilon)^{-1})$ by setting $\alpha \leftarrow (1-\varepsilon/\tilde{\varepsilon})/2$ (corresponding to $(1-\alpha)\tilde{\varepsilon} = (\varepsilon + \tilde{\varepsilon})/2$). This concludes the proof.

6 Combinatorial Analysis of the Construction

The soundness error of our new proof system depends on a combinatorial bound p, which bounds the probability that a cheating prover with an incorrect witness x finds preprocessing material (s, t) such that the verifier test with $x, \pi(s), \pi(t)$ passes, over the choice of the random permutation π. In this section, we provide an explicit formula for computing a tight combinatorial bound p in the setting where $T = K/w = 6$ (corresponding to $\mathbb{Z}_{T/2}$ being the smallest ring whose order is coprime with 2; this is the choice that minimizes the communication of our proof). In this setting, a valid witness is a concatenation of w blocks of length 6, each block being a unit vector.

6.1 A Formula for Estimating p

We consider an adversary using an *f-strongly invalid witness*, see Definition 4, and choosing possibly preprocessing material $(\llbracket s \rrbracket_2, \llbracket t \rrbracket_3)$ (*i.e.* s might differ from t). The verifier picks and sends a uniformly random permutation $\pi : [K] \mapsto [K]$. After opening $z = x \oplus \pi(s)$, we compute $\llbracket x' \rrbracket_3$ as $\llbracket \pi(t) + z - 2z \odot \pi(t) \rrbracket_3$. The following lemma provides an explicit formula for estimating p:

Lemma 7. *Assume that x is an f-strongly invalid witness satisfying $\forall j \leq w$, $\mathsf{HW}(x^j) = 1 \bmod 2$. It holds that*

$$\mathsf{p} \leq 1 - \min\left\{\min_{\substack{0.6K \leq \theta < K \\ f < \ell < K/6 - f}} \left(\frac{F(j, K, \ell, \theta)}{\binom{K}{\theta}}\right) \ ; \ 1 - 0.96^K/(2K) \ ; \ (1 - e^{-0.2 \cdot K/6}) \cdot \left(1 - \frac{1}{\binom{K/6}{K/60}}\right)\right\},$$

where

$$F(j, K, \ell, \theta) := \sum_{j=1}^{K/6} (-1)^{j+1} \cdot \sum_{i=0}^{j} 6^i \cdot \binom{\ell}{i} \cdot \binom{K/6 - \ell}{j - i} \cdot \binom{K - 6 \cdot j}{\theta - 6 \cdot j + i}.$$

Some intuition about the bound. In the above formula, the first term of min corresponds to the setting where the preprocessing material (s, t) is *mostly honest* (*i.e.* $s_i = t_i$ on most positions) or *mostly dishonest* (*i.e.* $s_i \neq t_i$ on most positions), and the second term to the setting where (s, t) is *well spreads* (it has many positions with $s_i \neq t_i$ and many with $s_i = t_i$).

In the latter case, passing the verifier test requires being very lucky with the permutation π: since the honest and dishonest positions in (s, t) are randomly shuffled, they have a very high probability to not align well with the invalid witness x. We formalize this intuition by showing that the success event in this case is dominated by an event which a follows a hypergeometric distribution, and by applying standard concentration bounds for hypergeometric distributions.

The former case is more involved. Here, we bound the success probability for every fixed number ℓ of *incorrect blocks* in the witness x (*i.e.* blocks whose Hamming weight is not 1), using counting arguments and the inclusion-exclusion principle. Then, we minimize over all possible choices of ℓ, excluding only the cases $\ell \leq f$ and $\ell \geq K/6 - f$: these cases correspond to the witness x being an *f-weakly valid witness* as per Definition 4.

We note that in fact, the same analysis provides a bound on the probability of using any witness other than a strictly regular or a strictly antiregular vector. However, this bound is too weak for our purpose. Intuitively, the cheating prover can use the following strategy: he uses a witness x which is valid everywhere, except on a single block. Then, it chooses (s, t) to be a honest preprocessing material except on a single position i^*. With an appropriate choice of the value on the incorrect position, the cheating prover passs the verifier test whenever the permutation π aligns i^* with the faulty block of x. However, this event happens with probability $6/K$. For standard choices of parameters, this quantity is in the range $[250, 350]$, leading to a failure bound p in the range of $1/300$. This is too high for our purpose since our aim is to obtain small signatures, which requires designing a "one-shot" zero-knowledge proof with low soundness error.

A conjecture. In fact, we conjecture that this naive strategy is the best possible. That is, if the cheating prover holds an f-strongly invalid witness, the best he can do is (1) using a witness with exactly two incorrect blocks, and (2) choosing (s, t) to be a honest preprocessing material except on two positions (i_0, i_1). Then, the prover wins if and only if the permutation aligns exactly i_0, i_1 each with one of the two incorrect blocks. In this case, the winning probability of the prover is bounded by $\binom{w}{2}^{-1}$. We state this conjecture below:

Conjecture 8. If x is an f-strongly invalid witness satisfying $\forall j \leq w$, $\mathsf{HW}(x^j) = 1 \bmod 2$, then

$$\mathsf{p} \leq \binom{w}{f}^{-1}.$$

We do not have a proof of the conjecture. Nevertheless, for any concrete choice of the parameter K, it is not too hard to compute the concrete bound: one can run a small program computing explicitly the formula of Lemma 7. When doing so, we observe that we reach a bound which is very close to the bound of the conjecture. For example, computing the formula with $f = 2$ and $K = 1284$ yields a value of p approximately equal to $1/22880$, only slightly looser than the bound of $1/22898$ predicted by the conjecture. We provide the Python program used to estimate the bound in Appendix A of the Supplementary Material. Note that naively computing the double sum of products of binomials for all values of ℓ in $[f + 1, K/6 - f - 1]$ would yield a very slow runtimes (a few days over a laptop for values of K in the thousands). However, carefully storing some of the binomials and reusing them when appropriate makes the computation significantly faster. Below, we proceed with the proof of Lemma 7.

Notations. In the following, we let Fail denote the event, defined over the random sampling of π, that x' does not pass the verifier check; *i.e.*, that $\mathsf{Succ}(x') = 0$. In other words, the event Fail is raised if there is $i \in [1, K/6]$ such that $\sum_{j=6i-5}^{6i} x'_j \neq 1 \bmod 3$. We also define useful notations to discuss the type of preprocessing material (s, t) which a cheating prover can use. Observe that a pair $(s_i, t_i) \in \mathbb{F}_2 \times \mathbb{F}_3$ can take six possible values. We categorize the pairs (s_i, t_i) in the three following types:

Type A. $(s_i, t_i) \in \{(0, 0), (1, 1)\}$, which we call the copy type,
Type B. $(s_i, t_i) \in \{(0, 1), (1, 0)\}$, which we call the flip type,
Type C. $(s_i, t_i) \in \{(0, 2), (1, 2)\}$, which we call the $\mathsf{const2}$ type.

We explain the names $\mathsf{copy}, \mathsf{flip}, \mathsf{const2}$. It is helpful to understand these categories by viewing a pair (s_i, t_i) as the following *operator* which transforms a bit $u \in \mathbb{F}_2$ into $u' \in \mathbb{F}_3$:

$$f_{s_i, t_i} : u \to t_i + (u \oplus s_i) - 2 \cdot (u \oplus s_i) \cdot t_i \bmod 3.$$

It is easy to check that the six possible pairs (s_i, t_i) correspond only to three possible distinct operators. A type-A pair corresponds to the copy operator, which transforms $u \in \mathbb{F}_2$ into $u \in \mathbb{F}_3$. A type-B pair corresponds to the flip operator, which transforms $u \in \mathbb{F}_2$ into $1 - u \in \mathbb{F}_3$. Eventually, a type-C pair corresponds to the $\mathsf{const2}$ operator, which maps any $u \in \mathbb{F}_2$ to the constant $2 \in \mathbb{F}_3$.

Our analysis distinguishes two complementary scenarios for the vectors (s, t) (whose shares form the preprocessing material): either a type is *dominant* among the vectors components (*i.e.*, a significant proportion of all pairs (s_i, t_i) are of the same type), or the types are *well-spread.* The cutoff between these two scenarios does not matter much, but for the sake of concreteness, we say that (s, t) is in the *dominant* scenario if there is a type (either A, B, or C) such that more than 60% of all pairs (s_i, t_i) are of this type, and that (s, t) is in the *well-spread* scenario otherwise. We analyze each scenario separately.

Concentration inequality. We use a variant of the Chernoff inequality for hypergeometric distributions:

Lemma 9. *Suppose we have an urn with N balls, P of which are black. We randomly draw n balls from the urn (without replacement). Let $H(N, P, n)$ denote the number of black balls in the sample, and let $q \leftarrow P/N$. Then for any $t \in [0, 1-q]$, we have*

$$\Pr[H(N, P, n) \geq (q + t) \cdot n] \leq \exp\left(-n \cdot D_{\mathsf{KL}}(q + t||q)\right),$$

where $D_{\mathsf{KL}}(u||v) = u \cdot \ln(u/v) + (1-u) \cdot \ln((1-u)/(1-v))$ denotes the Kullback-Leibler divergence (or relative entropy).

6.2 The Dominant Scenario

We assume in this section that more than 60% of all pairs (s_i, t_i) are of the same type.

Case 1: type A is dominant. We first assume that the dominant type is A: at least 60% of all pairs are of the copy type. Let $\theta \geq 0.6 \cdot K$ denote the number of type-A pairs in $(\llbracket s \rrbracket, \llbracket t \rrbracket_3)$, *i.e.*, the number of coordinates $i \in [K]$ such that $s_i = t_i$.

Lemma 10. *Assume that x is a strongly invalid witness satisfying $\forall j \leq w$, $\mathsf{HW}(x^j) = 1 \bmod 2$, and that the number of type-A pairs in (s, t) is $\theta \geq 0.6K$. It holds that*

$$\Pr_{\pi}[\mathsf{Fail}] \geq \frac{\min_{f<\ell<K/6-f}\left(\sum_{j=1}^{K/6}(-1)^{j+1} \cdot \sum_{i=0}^{j} 6^i \cdot \binom{\ell}{i} \cdot \binom{K/6-\ell}{j-i} \cdot \binom{K-6\cdot j}{\theta-6\cdot j+i}\right)}{\binom{K}{\theta}}.$$

To prove Lemma 10, we again start by setting up some useful notations. Recall that x is divided into $K/6$ blocks x^j of length 6 (with $j \leq K/6$), where $x^j = (x_{6j-5}, \cdots, x_{6j})$. Because of the condition that x satisfies $\forall j \leq w$, $\mathsf{HW}(x^j) = 1 \bmod 2$ (which is guaranteed by construction in the proof), each x^j has Hamming weight either 1, 3, or 5. For every block x^j, we say that x^j is a *honest block* if $\mathsf{HW}(x^j) = 1$, and that it is a *dishonest block* otherwise (observe that if all the x^j are honest blocks, then x is a valid witness).

Proof. To bound the probability of the event Fail, we reformulate it as a balls-and-bins game. Each bin corresponds to an entry of the vector x, and each ball to a pair (s_i, t_i) from the preprocessing material. For the proof of Lemma 10, we mostly distinguish two types of balls: type-A balls and non-type-A balls. Then, the experiment becomes: we randomly throw K balls into K bins, where θ balls are type-A balls, and $K - \theta$ balls are non-type-A balls (the value of θ is adversarially chosen, but satisfies $\theta \geq 0.6 \cdot K$), such that each ball ends up in exactly one bin. Each bin i is labelled with the corresponding bit x_i of x. The bins are divided into $K/6$ blocks $B_1, \cdots, B_{K/6}$ of six bins. When a ball falls into a bin i, it changes its label x_i into a new label x'_i according to the operator: the label remains the same for type-A balls (copy type), it is "flipped" for type-B balls (flip type), and it is replaced by 2 for type-C balls ($\mathsf{const2}$ type).

We say that a block of bins B_j is honest if its bins are labelled with a honest block x_j of the witness x; otherwise, it is a dishonest block of bins. Let ℓ denote the number of honest blocks of bins. For simplicity, we assume that the honest blocks are B_1 to B_ℓ, and the remaining blocks are the dishonest blocks; this is without loss of generality.

Then, we define the following events $E_j = E_j^{(\ell)}$ for $j = 1$ to $K/6$ (we omit the superscript ℓ for readability): for $j = 1$ to ℓ, E_j is the event that exactly five type-A balls end up in the block B_j, and for $j = \ell + 1$ to $K/6$, E_j is the event that exactly six type-A balls end up in the block B_j. The rationale behind the choice of the E_j is the following: (1) if any E_j happens, then the event Fail is triggered, and (2) when the proportion of type-A balls is large, it is likely that one such event happens.

Claim.

$$\Pr[\mathsf{Fail}] \geq \Pr\left[\bigcup_{j=1}^{K/6} E_j\right].$$

Proof. We show that $\cup_{j=1}^{K/6} E_j \subset \mathsf{Fail}$. Fix any $j \leq K/6$. Assume first that $j \leq \ell$, then E_j is the event that exactly five type-A balls fell into B_j. As B_j is a honest block, it has five bins labeled with 0, and one bin labeled with 1. There are two possibilities:

- Either the five type-A balls fell in the five 0-labeled bins. Then, the labels of these five bins remains 0, and the label of the remaining 1-labeled bin is either flipped to 0 (if the remaining ball is a type-B ball) or set to 2 (if the remaining ball is a type-C balls). Therefore, the Hamming weight of the new labels $\mathsf{HW}(x'_j)$ is either 0 or 2: in both cases, it is not equal to 1 modulo 3, hence it fails the verifier check.
- Otherwise, four 0-labeled bins and the 1-labeled bin have their label unchanged, and the remaining 0-labeled bin is either set to 1 (type-B ball) or to 2 (type-C ball). In this case, $\mathsf{HW}(x'_j) \in \{2, 3\}$, hence $\mathsf{HW}(x'_j) \neq 1 \bmod 3$.

Hence, whenever E_j happens, the second verifier check fails, which proves that $E_j \subset \mathsf{Fail}$ for $j = 1$ to ℓ. Assume now that $j > \ell$: E_j is the event that exactly 6 type-A balls fell into B_j, and B_j is a dishonest block (*i.e.*, $\mathsf{HW}(x_j) \in \{3, 5\}$). Since the six type-A balls leave the labels of all six bins unchanged, the Hamming weight of B_j's labels remains 3 or 5, and neither values are equal to 1 modulo 3, causing the second verifier check to fail again. This proves that $E_j \subset \mathsf{Fail}$ for $j = \ell + 1$ to K as well, which concludes the claim. □

It remains to bound $\Pr\left[\cup_{j=1}^{K/6} E_j\right]$. We proceed through standard combinatorial arguments: we consider the space of all possible configurations of the experiment, without distinguishing type-B and type-C balls (*i.e.*, we count configurations by only looking at whether each bin contains a type-A or a non-type-A ball). There are exactly $\binom{K}{\theta}$ possible configurations; we now count the number of configurations where one of the events E_j happen. By the inclusion-exclusion principle:

$$\left|\bigcup_{j=1}^{K/6} E_j\right| = \sum_{j=1}^{K/6} (-1)^{j+1} \sum_{S \subset [1, K/6],\ |S| = j} \left|\bigcap_{k \in S} E_k\right|.$$

To bound the terms $|\cap_{k\in S} E_k|$, we must distinguish between the E_k with $k \leq \ell$ and those with $k > \ell$. Fix an integer $i \in [0, j]$. Among all possible subsets $S \subset [1, K/6]$ of size $|S| = j$, there are exactly $\binom{\ell}{i} \cdot \binom{K/6-\ell}{j-i}$ subsets S such that $|S \cap [1, \ell]| = i$. Fix any such subset $S = \{k_1, \cdots, k_i, k_{i+1}, \cdots, k_j\}$, and consider the set of configurations $|E_{k_1} \cap \cdots \cap E_{k_j}|$. Picking a configuration in this set amounts to

- filling all the (dishonest) blocks of bins $B_{k_{i+1}} \cdots B_{k_j}$ with type-A balls (using $6(j - i)$ type-A balls),
- choosing one bin in each (honest) block $B_{k_1} \cdots B_{k_i}$ which does *not* receive a type-A ball,
- filling all remaining bins in $B_{k_1} \cdots B_{k_i}$ with type-A balls (using $5i$ type-A balls),
- choosing a configuration of the $\theta - 6(j - i) - 5i = \theta - 6j + i$ remaining type-A balls among the $K - 6j$ remaining bins.

There are 6^i ways to pick one bin in each of the i blocks $B_{k_1} \cdots B_{k_i}$, and $\binom{K-6j}{\theta-6j+i}$ ways to pick a configuration of the remaining type-A balls among the remaining bins. Therefore,

$$|E_{k_1} \cap \cdots \cap E_{k_j}| = 6^i \cdot \binom{K - 6j}{\theta - 6j + i}.$$

Hence, by summing over all subsets S of size j with i elements below ℓ, for $i = 0$ to j, we have

$$\sum_{S \subset [1, K/6],\ |S| = j} \left|\bigcap_{k \in S} E_k\right| = \sum_{i=0}^{j} \binom{\ell}{i} \cdot \binom{K/6 - \ell}{j - i} \cdot 6^i \cdot \binom{K - 6j}{\theta - 6j + i},$$

and therefore

$$\left|\bigcup_{j=1}^{K/6} E_j\right| = \sum_{j=1}^{K/6}(-1)^{j+1}\sum_{i=0}^{j}\binom{\ell}{i}\cdot\binom{K/6-\ell}{j-i}\cdot 6^i\cdot\binom{K-6j}{\theta-6j+i}.$$

Now, by the previous claim, $\Pr[\mathsf{Fail}] \geq \Pr[\cup_{j=1}^{K/6}E_j] = |\cup_{j=1}^{K/6}E_j|/\binom{K}{\theta}$. Since the number ℓ of honest blocks of bins is chosen by the adversary, we minimize the failure probability over all possible values of ℓ, excluding only the cases $\ell \geq K/6 - f$ and $\ell \leq f$, since x is assumed to be an f-strongly invalid witness (Definition 4). This concludes the proof of Lemma 10. □

Case 2: type B is dominant. We now deal with the case where at least 60% of all pairs are of the flip type. This case is handled immediately through a simple reduction to Case 1. First, observe that by definition, a vector x is an f-strongly invalid witness if and only if $\mathbf{1}\oplus x$ is: this follows from Definition 4, and from the fact that $\mathsf{HW}(x^j) = \mathsf{HW}((\mathbf{1}\oplus x)^j)$ for any j since the block length is even. Now, fix any f-strongly invalid witness x. Observe that the probability $\Pr_\pi[\mathsf{Fail}]$ remains identical if we (1) replace x by $\mathbf{1}\oplus x$ (*i.e.* we flip all the bits of x) and (2) replace all type B balls by type A balls and vice-versa (leaving all type C balls untouched). In other words, flipping all bits of x and exchanging the roles of the copy and flip operators leaves the experiment unchanged (the behavior of $\mathsf{const2}$ operators being unaffected by this change). But this brings us back to the setting where type A is dominant, with an f-strongly invalid witness $\mathbf{1}\oplus x$, and the bound of Lemma 10 applies.

Case 3: type C is dominant. It remains to deal with the case where at least 60% of all pairs are of the $\mathsf{const2}$ type. As the fraction of type C balls approaches 1, it is easy to see that $\Pr[\mathsf{Fail}]$ approaches 1 as well, since whenever six type C balls fall into a block of six bins, the event Fail is raised (as $6\cdot 2 = 0 \neq 1 \bmod 3$). Therefore, without loss of generality, we assume that the fraction of type C balls is exactly 0.6. We will show that in this case,

$$\Pr_\pi[\mathsf{Fail}] \geq 1 - \frac{0.96^K}{2K}.$$

To simplify the analysis, we first consider a different distribution where all bins receive a type A, B, or C ball with the same respective probabilities. We call this setting the *simplified setting* and denote by $\mathsf{Fail}^{\mathsf{s}}, \mathsf{Win}^{\mathsf{s}}$ the events that the adversary fails or wins in this setting. Now, fix a block b of six bins, and let hw denote the Hamming weight of the labels of the block. For $i = 4, 5, 6$, denote iC the event that exactly i type C balls fall into the block, and let p denote the probability that a type A ball falls into a given bin, conditioned on the event that this bin does not receive a type C ball (then $1-p$ is the probability that a type B ball falls into the bin, under the same condition). We let $\mathsf{Win}^{\mathsf{s}}_b$ denote the event that the adversary wins for block b (*i.e.* after placing balls in the bins, the labels sum to 1 mod 3). We bound the probability that the adversary wins conditioned on iC balls falling into the block b. First, $\Pr[\mathsf{Win}^{\mathsf{s}}_b \mid 6C] = 0$ (since $6\cdot 2 \neq 1 \bmod 3$). Second,

$$\Pr[\mathsf{Win}^{\mathsf{s}}_b \mid 5C] = (1-\mathsf{hw}/6)p + (1-p)\cdot\mathsf{hw}/6 = (1-\mathsf{hw}/3)\cdot p + \mathsf{hw}/6.$$

Above, $1-\mathsf{hw}/6$ is the probability that the non-type C bin is a 0-labeled bin: observe that the adversary wins either if a type A ball falls into a 0-labeled bin, or if a type B ball falls into a 1-labeled bin (since $5\cdot 2 + 0 = 10 = 1 \bmod 3$. Eventually,

$$\Pr[\mathsf{Win}^{\mathsf{s}}_b \mid 4C] = (1-\mathsf{hw}/6)(1-\mathsf{hw}/5)\cdot(1-p)^2 + \mathsf{hw}(6-\mathsf{hw})\cdot p(1-p)/15 + \mathsf{hw}(\mathsf{hw}-1)\cdot p^2/30.$$

Above, $(1-\mathsf{hw}/6)(1-\mathsf{hw}/5)\cdot(1-p)^2$ is the probability that the two non-type C balls are type B balls and they both fall into 0-labeled bins, $\mathsf{hw}(6-\mathsf{hw})\cdot p(1-p)/15$ is the probability that the two non-type C balls are a type A and a type B ball and they fall into a 1-labeled and a 0-labeled bin respectively, and $\mathsf{hw}(\mathsf{hw}-1)\cdot p^2/30$ is the probability that the two non-type C balls are type A balls and they both fall into 1-labeled bins. Since $4\cdot 2 + 2\cdot 1$ is the only way to get 1 mod 3 with 4 type C balls, this covers all winning events for the adversary. Eventually,

$$\begin{aligned}\Pr[\mathsf{Fail}_b^{\mathsf{s}}] &\geq \Pr[\mathsf{Fail}_b^{\mathsf{s}} \wedge 4C] + \Pr[\mathsf{Fail}_b^{\mathsf{s}} \wedge 5C] + \Pr[\mathsf{Fail}_b^{\mathsf{s}} \wedge 6C] \\ &= \Pr[\mathsf{Fail}_b^{\mathsf{s}} \mid 4C] \cdot \Pr[4C] + \Pr[\mathsf{Fail}^{\mathsf{s}} \mid 5C] \cdot \Pr[5C] + \Pr[\mathsf{Fail}_b^{\mathsf{s}} \mid 6C] \cdot \Pr[6C] \\ &= \Pr[\mathsf{Fail}_b^{\mathsf{s}} \mid 4C] \cdot \binom{6}{4} \cdot 0.6^4 \cdot 0.4^2 + \Pr[\mathsf{Fail}_b^{\mathsf{s}} \mid 5C] \cdot 6 \cdot 0.6^5 \cdot 0.4 + \Pr[\mathsf{Fail}_b^{\mathsf{s}} \mid 6C] \cdot 0.6^6.\end{aligned}$$

Therefore, for each possible value of the Hamming weight $\mathsf{hw} \in \{0, \cdots, 6\}$ of the block of bins, plugging the formulas for $\Pr[\mathsf{Fail}_b^{\mathsf{s}} \mid iC] = 1 - \Pr[\mathsf{Win}_b^{\mathsf{s}} \mid iC]$ for $i = 4, 5, 6$ above yields a degree-2 polynomial Q_{hw} in p. Let us denote $(\alpha_{\mathsf{hw}}, \beta_{\mathsf{hw}}, \gamma_{\mathsf{hw}})$ the coefficients of this polynomial: $Q_{\mathsf{hw}}(p) = \alpha_{\mathsf{hw}} \cdot p^2 + \beta_{\mathsf{hw}}, \cdot p + \gamma_{\mathsf{hw}}$. For each value of $\mathsf{hw} \in \{0, \cdots, 6\}$, we minimize $Q_{\mathsf{hw}}(p)$ for $p \in [0, 1]$:

- $Q_0(p)$ is minimized at $p = 0$, and $Q_0(0) = 729/3125$,
- $Q_1(p)$ is minimized at $p = 0$, and $Q_0(0) = 4779/15625$,
- $Q_2(p)$ is minimized at $p = 0$, and $Q_0(0) = 5589/15625$,
- $Q_3(p)$ is minimized at $p = 0$ or 1, and $Q_0(0) = 5832/15625$.

Eventually, $Q_4(p)$, $Q_5(p)$, and $Q_6(p)$ are minimized at $p = 1$ and the minima are the same as $Q_2(p)$, $Q_1(p)$, and $Q_0(p)$ respectively (this stems from the fact that the roles of A balls and B balls are symmetrical when 0-labeled bins are exchanged with 1-labeled bins, which changes hw to $6 - \mathsf{hw}$).

Overall, the global minimum over (hw, p) is reached at $Q_0(0) = Q_6(1) = 729/3125$. This yields the following bound:

$$\Pr[\mathsf{Fail}_b^{\mathsf{s}}] \geq 729/3125 = 0.23328.$$

Then, across all K/6 blocks of bins, we get the bound

$$\Pr[\mathsf{Fail}^{\mathsf{s}}] \geq 1 - (1 - 0.23328)^{K/6} \leq 1 - 0.77^{K/6} \leq 0.96^K.$$

Recall that this bound holds in the simplified setting where each bin is assigned a ball whose type is *independently* sampled from $\{A, B, C\}$ with probabilities $p_A = 0.4 \cdot p$, $p_B = 0.4 \cdot (1 - p)$, and $p_C = 0.6$ (that is, we use a Bernoulli-style distribution). Now, let K_A, K_B, K_C denote the random variables that count the number of bins with type A, B, and C balls respectively. Observe that

$$\begin{aligned}\Pr[\mathsf{Win}^{\mathsf{s}}] &\geq \Pr[\mathsf{Win}^{\mathsf{s}} \wedge (K_A = 0.4pK \wedge K_C = 0.6K)] \\ &= \Pr[\mathsf{Win}^{\mathsf{s}} \mid K_A = 0.4pK, K_C = 0.6K] \cdot \Pr[K_A = 0.4pK \wedge K_C = 0.6K] \\ &\geq \Pr[\mathsf{Win}^{\mathsf{s}} \mid K_A = 0.4pK, K_C = 0.6K] \cdot \Pr[K_A = 0.4pK] \cdot \Pr[K_C = 0.6K].\end{aligned}$$

Furthermore, the probability $\Pr[\mathsf{Win}^{\mathsf{s}} \mid K_A = 0.4pK, K_C = 0.6K]$ corresponds exactly to the event that the adversary wins in the original setting, where a fixed pool of balls (with $0.4pK$ type A, $0.4(1 - p)K$ type B, and $0.6K$ type C balls) is randomly permuted and placed into the bins. That is,

$$\Pr[\mathsf{Win}^{\mathsf{s}} \mid K_A = 0.4pK, K_C = 0.6K] = \Pr[\mathsf{Win}^{\mathsf{s}}],$$

Hence

$$\Pr[\mathsf{Win}] \leq \frac{\Pr[\mathsf{Win}^{\mathsf{s}}]}{\Pr[K_A = 0.4pK] \cdot \Pr[K_C = 0.6K]} \leq \frac{0.77^{K/6}}{\Pr[K_A = 0.4pK] \cdot \Pr[K_C = 0.6K]}.$$

To finish the proof, it remains to lower bound $\Pr[K_A = 0.4pK]$ and $\Pr[K_C = 0.6K]$. Since K_A and K_C are both binomial random variables of respective means $0.4pK$ and $0.6K$, This can be done in a fairly straightforward way: let K_X be any binomial random variable in $\{0, \cdots, K\}$ with mean μ. Then

$$\begin{aligned}\Pr[K_X = \mu] &= \binom{K}{\mu} \cdot (\mu/K)^{\mu} \cdot (1 - \mu/K)^{K-\mu} \\ &= \binom{K}{\mu} \cdot e^{-h(\mu/K) \cdot K},\end{aligned}$$

where $h : x \to -x \ln x - (1-x)\ln(1-x)$ is the binary entropy function. Furthermore, by a standard application of the Stirling inequality[5],

$$\Pr[K_X = \mu] = \binom{K}{\mu} \cdot e^{-h(\mu/K)\cdot K} \geq \sqrt{\frac{K}{8\mu(K-\mu)}} \geq \frac{1}{\sqrt{2K}},$$

where the last inequalities is obtained by setting $\mu = K/2$, since it minimizes the expression. Plugging this in our bound on $\Pr[\mathsf{Win}]$, we get

$$\Pr[\mathsf{Win}] \leq \frac{0.96^K}{2K},$$

which concludes the analysis of the dominant scenario.

6.3 The Well-Spread Scenario

We again use the same balls-and-bins formalism as in the previous Section. We assume in this section that the preprocessing material $([\![s]\!]_2, [\![t]\!]_3)$ contains strictly less than 60% of pairs (s_i, t_i) of the most represented type. We show that if this is the case, no matter the witness x used by the prover, the event Fail is extremely likely to happen. The core intuition is the following: fix any block B of six bins, and fix any choice of type-A, type-B, and type-C balls for the first five bins. Let y denote the sum modulo 3 of the new labels in these five bins. If the types of balls are sufficiently well-spread, then the distribution of possible new labels for the leftover bin is also well-spread accross the set $\{0, 1, 2\}$. For example, in the most extreme case where the unused balls have an equal proportion of each of the three types, after a ball is randomly picked and thrown in the leftover bin, the new label of the bin is uniformly random over $\{0, 1, 2\}$, independently of its original label. Therefore, in the well-spread scenario, the new label of the leftover bin has a noticeable probability of being distinct from $1 - y \bmod 3$, in which case a failure event is raised.

The above intuition shows that each block is likely to cause a failure in this scenarios. While the events of a block causing a failure are not independent across different blocks, when the total number of balls and bins is large enough, their mutual influence appears very limited, and we expect the event that *at least one* of the blocks causes a failure to quickly become overwhelming. Below, we prove that this intuition is indeed correct.

Lemma 11. *Assume that x is a strongly invalid witness satisfying $\forall j \leq w$, $\mathsf{HW}(x^j) = 1 \bmod 2$. If the preprocessing material $([\![s]\!]_2, [\![t]\!]_3)$ contains strictly less than 60% of pairs (s_i, t_i) of the most represented type, it holds that*

$$\Pr_{\pi}[\mathsf{Fail}] \geq (1 - e^{-0.2\cdot K/6}) \cdot \left(1 - 1/\binom{K/6}{K/60}\right).$$

Proof. Fix one bin in each block (the *selected bins*). Randomly throwing the balls into all bins is equivalent to the following experiment: we first pick a uniformly random size-$K/6$ subset of all balls (the *selected balls*). We throw the remaining balls randomly among the remaining bins, and the selected balls randomly among the selected bins. We bound the probability of a failure event by proving two claims: (1) with high probability, the three types of balls (A, B, and C) remain relatively *well-spread* among the randomly selected balls, and (2) when the selected balls are well-spread, a failure event is extremely likely to happen, for any setting of the remaining balls into the remaining bins.

Claim. Let S be a uniformly random size-$K/6$ subset of $[1, K]$. Let p denote the probability, over the random choice of S, that the set $\{(s_i, t_i) \ : \ i \in S\}$ contains more than 90% of pairs of any given type. Then

$$p < \exp(-0.2 \cdot K/6).$$

Proof. Let θ/K denote the proportion of the most frequent type among all pairs (s_i, t_i); by assumption, $\theta/K < 0.6$. We view as "black balls" the balls of the most frequent type (in case of ties, the exact choice does not matter). The claim follows by applying the Chernoff inequality for the hypergeometric distribution given in Theorem 9, using $q = 0.6$ and $t = 0.3$. □

[5] This formulation of Stirling's inequality can be found for example in [Gal68].

Claim. Fix a set S of $K/6$ selected balls such that the set $\{(s_i, t_i) \ : \ i \in S\}$ does not contain more than 90% of pairs of any given type. Fix $K/6$ selected bins (one in each block). Fix an arbitrary repartition of the remaining balls into the remaining bins. Then the probability, taken over the random assignment of the $K/6$ selected balls to the $K/6$ selected bins, that no failure event happens is bounded by

$$\Pr[\text{no failure}] \leq \frac{1}{\binom{K/6}{K/60}}.$$

Proof. Let $1/3 \leq \theta/K < 0.9$ denote the proportion of the most common type among all selected balls. Let us call "black balls" the balls of this type, and "white balls" the remaining balls. For any block, and any assignment of non-selected balls among the five non-selected bins of this block, there exactly one type of ball (among A, B, and C) such that assigning this ball to the selected bin of the block does not lead to a failure: if y is the sum (modulo 3) of the new labels in the non-selected bins, this is the type which changes the label of the selected bin to $1 - y \bmod 3$. Imagine now that we paint each selected bin as follows: if the type that does not cause a failure in this block is that of the black balls, we color it in black; otherwise, we color it in white.

Then, two cases can happen: either the number of black balls among the selected balls is not equal to the number of black selected bins. In this case, no assignment can put a ball of the right color in all selected bins, and $\Pr[\text{no failure}] = 0$. Otherwise, assume that exactly $\theta/6$ selected bins are black. To avoid a failure event, it is necessary that the $\theta/6$ balls that end up in the $\theta/6$ black selected bins are exactly all the black balls. There are $\binom{K/6}{\theta/6}$ ways to choose which balls will end up in the black bins among the $K/6$ selected balls, and only one non-loosing configuration, hence

$$\Pr[\text{no failure}] \leq \frac{1}{\binom{K/6}{\theta/6}}.$$

From here, the claim follows using the fact that $\binom{K/6}{\theta/6} \geq \binom{K/6}{0.9 \cdot K/6} = \binom{K/6}{0.1 \cdot K/6}$ since $0.1 < \theta/K < 0.9$. □

Equipped with the two claims, the proof of Lemma 11 follows almost immediately: fix $K/6$ selected bins, one per block, and let S denote the indices of the balls that end up in these bins. We say thay S is well-spread if no type represents more than 90% of the types of the balls in S. Then,

$$\begin{aligned}\Pr[\mathsf{Fail}] &\geq \Pr[S \text{ is well-spread}] \cdot \Pr[\mathsf{Fail} \mid S \text{ is well-spread}] \\ &\geq (1 - e^{-0.2K/6}) \cdot \left(1 - \frac{1}{\binom{K/6}{K/60}}\right),\end{aligned}$$

which concludes the proof. □

This concludes the proof of Lemma 7.

7 A Signature scheme from Regular Syndrome Decoding

A signature scheme is composed of three algorithms $(\mathsf{KeyGen}, \mathsf{Sign}, \mathsf{Verify})$. KeyGen, starting with a security parameter λ, returns a key pair $(\mathsf{pk}, \mathsf{sk})$ where pk and sk are respectively the public key and the private key. The algorithm Sign on an input a message m and the secret key sk, gives a signature σ. Verify with input a message m, a public key pk and a signature σ, returns 0 or 1 depending on whether the signature σ is verified for m under pk or not. The security property for a signature scheme is the existential unforgeability against chosen message attacks: given a public key pk and an oracle access to $\mathsf{Sign}(\mathsf{sk}, \cdot)$ it is hard to obtain a pair (s, m) such that m was not queried to the signing oracle and $\mathsf{Verify}(\mathsf{pk}, s, m) = 1$.

In this section, we turn our 5-round protocol into a signature scheme using the Fiat-Shamir transform. The switch from an interactive protocol to a non-interactive protocol is done by calculating the two challenges π and d (corresponding respectively to the challenges chosen by the verifier in rounds 2 and 4 of our 5-round protocol) as follows:

$$h_1 = H(m, \mathsf{salt}, h), \qquad \pi \leftarrow \mathsf{PRG}(h_1), \qquad h_2 = H(m, \mathsf{salt}, h, h^{'}), \qquad d \leftarrow \mathsf{PRG}(h_2)$$

where m is the input message, H is an hash function and h and h' are the Round 1 and Round 3 hash commitments merged for the τ repetitions. As in previous works, we use a salt $\mathsf{salt} \in \{0,1\}^{2\lambda}$ to avoid $2^{\lambda/2}$-query attack resulting from collisions between seeds. We also take into account the forgery attack presented by Kales and Zaverucha [KZ20] against the signature schemes obtained by applying the Fiat-Shamir transform to 5-round protocols. Adapting this attack to our context yields a forgery cost of

$$\mathrm{cost}_{\mathrm{forge}} = \min_{\tau_1,\tau_2:\tau_1+\tau_2=\tau} \left\{ \frac{1}{\sum_{i=\tau_1}^{\tau} \binom{\tau}{i} \mathsf{p}^i (1-\mathsf{p})^{\tau-i}} + n^{\tau_2} \right\} \tag{1}$$

7.1 Description of the Signature Scheme

In our signature scheme, the key generation algorithm randomly samples a syndrome decoding instance (H, y) with solution x. We describe it on Figure 4.

Inputs: A security parameter λ.

1. Randomly chooses a $\mathsf{seed} \leftarrow \{0,1\}^{\lambda}$;
2. Using a pseudorandom generator with seed to obtain a regular vector $x \in \mathbb{F}_2^K$ with $\mathsf{HW}(x) = w$ and a matrix H;
3. Compute $y = Hx$;
4. Set $\mathsf{pk} = (H, y)$ and $\mathsf{sk} = (H, y, x)$.

Fig. 4. Key generation algorithm of the signature scheme

For a secret key $\mathsf{sk} = (H, y, x)$ and a message $m \in \{0,1\}^*$, the signing algorithm is described on Figure 5. Given a public key $\mathsf{pk} = (H, y)$, a message $m \in \{0,1\}^*$ and a signature σ, the verification algorithm is described in Figure 6.

Theorem 12. *Suppose the* PRG *used is* $(t, \epsilon_{\mathsf{PRG}})$*-secure and any adversary running in time* t *has at most an advantage* ϵ_{SD} *against the underlying d-split syndrome decoding problem. Model the hash functions* H_0, H_1, H_2 *as random oracles with output of length* 2λ*-bit. Then chosen-message adversary against the signature scheme depicted in Figure 5, running in time* t*, making* q_s *signing queries, and making* q_0, q_1, q_2 *queries, respectively, to the random oracles, succeeds in outputting a valid forgery with probability*

$$\Pr[\mathrm{Forge}] \leq \frac{(q_0 + \tau n_s)^2}{2 \cdot 2^{2\lambda}} + \frac{q_s\,(q_s + q_0 + q_1 + q_2)}{2^{2\lambda}} + q_s \cdot \tau \cdot \epsilon_{\mathsf{PRG}} + \epsilon_{\mathsf{SD}} + \Pr[X + Y = \tau] \tag{2}$$

where $\epsilon = \mathsf{p} + \frac{1}{n} - \frac{\mathsf{p}}{n}$*, with* p *given by Lemma 7,* $X = \max_{\alpha \in Q_1}\{X_\alpha\}$ *and* $Y = \max_{\beta \in Q_2}\{Y_\beta\}$ *with* $X_\alpha \sim \mathsf{Binomial}(\tau, \mathsf{p})$ *and* $Y_\beta \sim \mathsf{Binomial}\left(\tau - X, \frac{1}{n}\right)$ *where* Q_1 *and* Q_2 *are sets of all queries to oracles* H_1 *and* H_2*.*

The proof of Theorem 12 follows directly from the standard analysis of Fiat-Shamir-based signatures from 5-round identification protocol. It is identical to the proof of Theorem 5 in [FJR22a], and we omit it here.

7.2 Parameters Selection Process

In this section, we explain how to select parameters for the zero-knowledge argument system of Section 4.1 and the signature scheme of Section 7. Let f be the number of faulty blocks (of Hamming weight 3 or 5) allowed in the witness extracted from a cheating prover. Looking ahead, f is chosen as the smallest value that minimizes τ, the number of repetitions of the underlying zero-knowledge argument, which has a strong impact on the size of the signature. Given a candidate value f, our selection of the parameters (K, k, w) proceeds as outlined below. We remind the reader that we always enforce $w = K/6$ to get a blocksize 6, in order to work over the smallest possible field $\mathbb{F}_3$ in the zero-knowledge proof. We also set the target bit-security to $\lambda = 128$.

```
Inputs: A secret key $\mathsf{sk} = (H, y, x)$ and a message $m \in \{0,1\}^*$.
Sample a random $\mathsf{salt} \in \{0,1\}^{2\lambda}$.
Phase 1 For each iteration $e \in [\tau]$

  1. Choose a random seed $\mathsf{seed}_e \leftarrow \{0,1\}^\lambda$;
  2. Use $\mathsf{seed}_e$ and $\mathsf{salt}$ as input of a pseudorandom generator to produce $\mathsf{seed}_i^e$ for each $i \in [n]$;
  3. Compute $\mathsf{aux}_n^e$;
  4. Set $\mathsf{state}_i^e = \mathsf{seed}_i^e$ for $1 \leq i \leq n-1$ and $\mathsf{state}_n^e = \mathsf{seed}_n^e || \mathsf{aux}_n^e$;
  5. Use all the states to create, through a pseudorandom generator:
       – $\llbracket x^e \rrbracket_2 = (\mathbf{x}^{\mathbf{e}}_{\mathbf{1}}, \dots, \mathbf{x}^e_n)$;
       – $s = \llbracket r_1^e \rrbracket_2 = (\mathbf{s}^{\mathbf{e}}_{\mathbf{1}}, \dots, \mathbf{s}^{\mathbf{e}}_{\mathbf{n}})$;
       – $t = \llbracket r^e \rrbracket_q = (\mathbf{t}^{\mathbf{e}}_{\mathbf{1}}, \dots, \mathbf{t}^{\mathbf{e}}_{\mathbf{n}})$;
  6. For each $i \in [n]$ computes $\mathsf{com}_i^e := H_0(\mathsf{salt}, i, \mathsf{state}_i^e)$.

Phase 2

  1. Compute $h_1 = H_1(m, \mathsf{salt}, \mathsf{com}_1^1, \cdots, \mathsf{com}_n^1, \cdots, \mathsf{com}_\tau^1, \cdots, \mathsf{com}_n^\tau)$;
  2. Obtain $\pi^e_{\{e \in \tau\}} \in S_{K-k}$ via a pseudorandom generator using $h_1$.

Phase 3 For each iteration $e \in [\tau]$

  1. Each party $P_i$ computes $\mathbf{z}^{\mathbf{e}}_{\mathbf{i}} = \mathbf{x}^{\mathbf{e}}_{\mathbf{i}} \oplus \pi(\mathbf{s}^{\mathbf{e}}_{\mathbf{i}})$;
  2. The parties get $\llbracket z_1^e \rrbracket_2 = (\mathbf{z}^{\mathbf{e}}_{\mathbf{1}}, \cdots, \mathbf{z}^{\mathbf{e}}_{\mathbf{n}})$ and set $\llbracket z_2^e \rrbracket_2 = H' \llbracket z_1^e \rrbracket_2 \oplus y$ and so $z^e = (z_1^e | z_2^e)$;
  3. Obtain $\llbracket \bar{x^e} \rrbracket_q = (\bar{\mathbf{x}^{\mathbf{e}}_{\mathbf{1}}}, \dots, \bar{\mathbf{x}^{\mathbf{e}}_{\mathbf{n}}})$ where $\bar{\mathbf{x}^{\mathbf{e}}_{\mathbf{i}}} = z^e + (1 - 2z^e) * \pi(\mathbf{t}^{\mathbf{e}}_{\mathbf{i}})$;
  4. For each $j \in [n]$ compute:
       – $\bar{\mathbf{w}}^{\mathbf{j,e}}_{\mathbf{i}} = \langle 1, \bar{\mathbf{x}}_{\mathbf{i}}{}^{j,e} \rangle$ for all the blocks $1 \leq j \leq w$;
       – $\mathsf{msg}_i^e = \left( \mathbf{z}^{\mathbf{e}}_{\mathbf{i}}, \left\{ \bar{\mathbf{w}}^{\mathbf{j,e}}_{\mathbf{i}} \right\}_{1 \leq j \leq w} \right)$.

Phase 4

  1. Compute $h_2 = H_2(m, \mathsf{salt}, h_1, \mathsf{msg}_1^1, \cdots, \mathsf{msg}_n^1, \cdots, \mathsf{msg}_1^\tau, \cdots, \mathsf{msg}_n^\tau)$;
  2. Obtain $d^e_{\{e \in \tau\}} \in [n]$ via a pseudorandom generator using $h_2$.

Phase 5 Output the signature

       $\sigma = \mathsf{salt} | h_1 | h_2 | (\mathsf{state}^e_{i \neq d} | \mathsf{com}^e_{d^e})_{\{e \in \tau\}}$.
```

Fig. 5. Signing algorithm of the signature scheme

```
Inputs: A public key $\mathsf{pk} = (H, y)$, a message $m \in \{0,1\}^*$ and a signature $\sigma$.

  1. Split the signature as follows

       $\sigma = \mathsf{salt} | h_1 | h_2 | (\mathsf{state}^e_{i \neq d} | \mathsf{com}^e_{d^e})_{\{e \in \tau\}}$;

  2. Recompute $\pi^e_{\{e \in \tau\}} \in S_{K-k}$ via a pseudorandom generator using $h_1$;
  3. Recompute $d^e_{\{e \in \tau\}} \in [n]$ via a pseudorandom generator using $h_2$;
  4. For each iteration $e \in [\tau]$
       – For each $i \neq d$ recompute $\bar{\mathsf{com}}_i^e = H_0(\mathsf{salt}, i, \mathsf{state}_i^e)$;
       – Use all the states, except $\mathsf{state}^e_{d^e}$, to simulate the Phase 3 of the signing algorithm for all parties
         but the $d^e$–th, obtaining $\mathsf{msg}^e_{i \neq d^e}$;
       – Compute

         $\bar{\mathsf{msg}}^e_{d^e} = \left( z_1^e - \sum_{i \neq d} \mathbf{z}^{\mathbf{e}}_{\mathbf{i}}, \left\{ 1 - \sum_{i \neq d} \bar{\mathbf{w}}^{\mathbf{j,e}}_{\mathbf{i}} \right\}_{1 \leq j \leq w} \right)$;

  5. Check if $h_1 = H_1(m, \mathsf{salt}, \mathsf{com}_1^1, \cdots, \mathsf{com}_n^1, \cdots, \mathsf{com}_\tau^1, \cdots, \mathsf{com}_n^\tau)$;
  6. Check if $h_2 = H_2(m, \mathsf{salt}, h_1, \mathsf{msg}_1^1, \cdots, \mathsf{msg}_n^1, \cdots, \mathsf{msg}_1^\tau, \cdots, \mathsf{msg}_n^\tau)$;
  7. Output ACCEPT if both condition are satisfied.
```

Fig. 6. Verification algorithm of the signature scheme

Choosing k. As explained in Section 8.4, we set k such that even when allowing $f > 0$ faulty blocks in the zero-knowledge proof, the assumption underlying the unforgeability of the signature remains equivalent to the standard RSD assumption. Concretely, this is achieved by setting k to

$$k \leftarrow \left\lceil \log_2 \left(\sum_{i=0}^{f} 6^{w-i} \cdot \binom{w}{i} \cdot 26^i \right) \right\rceil + b \cdot \lambda,$$

with $b = 1$. We also consider a second choice of parameters, in which we set $b = 0$ in the above equation. This second choice of parameters corresponds to the f-almost-RSD uniqueness bound, the threshold where the number of almost-regular solution becomes close to 1. This setting should intuitively leads to the hardest instance of the almost-RSD problem. However, it does not reduce anymore to the standard RSD problem, since a random RSD instance might have irregular (but almost-regular) solutions. We use this alternative choice as a way to pick more aggressive parameters, under an exotic (albeit plausible) assumption.

Choosing K. Having chosen k (as a function of $w = K/6$), we turn our attention to K. Here, we use the attacks described in Sections 8.2 and 8.3 to select the smallest K such that, when setting k as above, we achieve λ bits of security against all attacks. We note that the approximate birthday paradox attack (Section 8.3) is always the most efficient attack, by a significant margin. Yet, it relies upon the assumption that approximate collisions can be found in linear time, and no such linear-time algorithm is known as of today. We view this optimistic evaluation of the attack efficiency as leading to a conservative choice of parameters.

Computing p. Equipped with a candidate instance (K, k, w) for a number f of faulty blocks, we use the formula of Lemma 7 to compute a bound p on the probability that a malicious prover can use an incorrect witness (with at least $f + 1$ faulty blocks) in the first part of the zero-knowledge proof. More precisely, since computing p exactly using the code given in Appendix A of the Supplementary Material takes a few hours of computation, we first set p using the value predicted by Conjecture 8 (which we found to match with all exact calculations we tried with the formula). Then, once we get a final choice of all parameters, we verify that the final bound p obtained was indeed correct, by running the explicit formula (hence running the code only once).

Computing τ. We compute the number of repetitions τ of the zero-knowledge argument, and of the signature scheme. This is where the parameter selection differs in each case:

Zero-knowledge argument. For the zero-knowledge argument, τ is computed as the smallest value such that $\varepsilon^\tau \leq 2^{-\lambda}$, where $\varepsilon = 1/n + \mathsf{p} \cdot (1 - 1/n)$, n being the number of parties. Here, there is no optimal choice of f. Instead, f is a tradeoff: choosing $f = 0$ guarantees that the zero-knowledge argument achieves standard soundness (with no gap) but makes ε higher. A larger f reduces p, hence ε, but introduces a gap in soundness. In any case, as soon as $\mathsf{p} \ll 1/n$, we have $\varepsilon \approx 1/n$. In practice, using $f = 1$ already leads to $\mathsf{p} < 5 \cdot 10^{-5}$, which is much smaller than any reasonable value of $1/n$ (since increasing n to such values would blow up computation). Hence, the only reasonable choices are $f = 0$ (for standard soundness) and $f = 1$ (for optimal efficiency).

Signature scheme. The signature scheme is obtained by compiling the zero-knowledge argument using Fiat-Shamir. Since we are compiling a 5-round zero-knowledge proof, the attack of Kales and Zaverucha [KZ20] applies, and we must choose τ according to Equation 1. This changes completely the optimal choice, since it is no longer true that any value of $\mathsf{p} \ll 1/n$ already leads to the smallest possible τ. In fact, by the convexity of Equation 1, the smallest possible τ one can hope for is $\tau_{\mathsf{ZK}} + 1$, where τ_{ZK} is the optimal value of τ for the zero-knowledge argument (*i.e.* the smallest value such that $\varepsilon^{\tau_{\mathsf{ZK}}} \leq 2^{-\lambda}$). Our strategy is therefore the following: we compute τ with Equation 1 for our candidate choice of f. Then, if $\tau > \tau_{\mathsf{ZK}} + 1$, we increase f by 1, and restart the entire procedure (choosing new parameters K, k, recomputing p, etc). After a few iterations, the procedure converges and yields the smallest number f of faulty blocks such that the resulting value of τ is minimal.

Choosing n. Eventually, it remains to choose the number of parties n. This choice is orthogonal to the other choices: a larger n always decreases communication (since it lowers the soundness error), but it increases computation (which scales linearly with n). To choose n, we use the same strategy as Banquet [BdK$^+$21]: we set n to a power of two, targeting a signing time comparable to that of previous works (on a standard laptop) for fairness of comparison. Then, we compute all parameters (K, k, w, f, τ), and reduce n to the smallest value which still achieves λ bits of security.

Runtime estimations. Eventually, it remains to estimate the runtime of the signature and verification algorithms of our signature scheme. Unfortunately, we do not yet have a full-fledged implementation of our signature scheme. We plan to write an optimized implementation of our new signature scheme in a future work. In the meantime, we use existing benchmark to conservatively estimate the runtime of our scheme. We consider the following implementation choices:

- The tree-based PRG is implemented with fixed-key AES. This is the standard and most efficient way to implement such PRGs over machines with hardware support for AES [AES01].
- The commitment scheme is implemented with fixed-key AES when committing to short values (λ bits), and with SHAKE when committing to larger values.
- The hash function is instantiated with SHAKE.

For fixed-key AES operations, the estimated runtime using hardware instructions is 1.3 cycles/byte [MSY21]. For SHAKE, the runtime strongly depends on a machine. However, according to the ECRYPT benchmarkings[6], on one core of a modern laptop, the cost of hashing long messages ranges from 5 to 8 cycles/byte (we used 8 cycles/byte in our estimations, to stay on the conservative side). Eventually, we also counted XOR operations (XORing two 64-bit machine words takes one cycle) and mod-3 operations. The latter are harder to estimate without a concrete implementation at hand. However, the contribution to the overall cost is relatively small: even estimating conservatively up to an order of magnitude of overhead compared to XOR operations has a minor impact on the overall runtime. We assumed an order of magnitude of overhead in our estimations, to remain on the conservative side. Eventually, when converting cycles to runtime, we assumed a 3.8 GHz processor, the same as in the previous work of [FJR22a], to facilitate comparison with their work (which is the most relevant to ours).

Of course, the above estimations ignore additional costs such as allocating or copying memory, and should therefore only be seen as a rough approximation of the timings that an optimized implementation could get. For comparison, in the Banquet signature scheme [BdK$^+$21], another candidate post-quantum signature scheme based on the MPC-in-the-head paradigm, 25% of the runtime of their optimized implementation was spent on allocating and copying memory, and 75% on the actual (arithmetic and cryptographic) operations.

Results. We considered two settings: a conservative setting, where the underlying assumption reduces to the standard RSD assumption, and an aggressive setting, where the parameters rely on the conjectured hardness of the f-almost-RSD assumption. All our numbers are reported on Table 1. We obtained the following parameters:

Conservative setting (standard RSD). We obtain an optimal choice of number f of faulty blocks equal to $f = 12$. Given this f, we set $K = 1842$, $k = 1017$, and $w = 307$. We targeted 128 bits of security against all known attacks, assuming conservatively that approximate birthday collisions can be found in linear time to estimate the cost of our most efficient attack. In this parameter range, the solution to the random RSD instance is the only 12-almost-regular solution except with probability 2^{-128}, hence 12-almost-RSD reduces to standard RSD. With these parameters, we considered three values of n. Each time, we first set n to a power of two, compute the optimal value of τ, and then reduce n to the smallest value that still works for this value of τ.

- Setting 1 – fast signature (rsd-f): $\tau = 18$, $n = 193$. In this setting, the signature size is 12.52 KB. The runtime estimated with our methodology described above is 2.7ms.
- Setting 2 – medium signature 1 (rsd-m1): $\tau = 13$, $n = 1723$. In this setting, the signature size is 9.69 KB. The runtime estimated with our methodology described above is 17ms.
- Setting 3 – medium signature 2 (rsd-m2): $\tau = 12$, $n = 3391$. In this setting, the signature size is 9.13 KB. The runtime estimated with our methodology described above is 31ms.
- Setting 4 – short signature 2 (rsd-s): $\tau = 11$, $n = 7644$. In this setting, the signature size is 8.55 KB. The runtime estimated with our methodology described above is 65ms.

[6] `https://bench.cr.yp.to/results-hash.html`

Aggressive setting (f-almost-RSD). In this setting, we set k at the f-almost-RSD uniqueness bound (the threshold above which the number of f-almost-regular solutions approaches 1). In this setting, there might be additional almost-regular solution beyond the regular solution x for a random RSD instance, hence f-almost-RSD does not reduce directly to the standard RSD assumption. We consider this assumption to be plausible but exotic, and investigate how relying on it improves the parameters. We view the conservative parameters as our main choice of parameters. The aggressive parameters yield noticeable improvements in signature size and runtime, which could motivate further cryptanalysis of this exotic variant. We provide four settings of parameters, comparable to our conservative settings, using the optimal value $f = 13$ and the same numbers n of parties as above. In this setting, we have $K = 1530$, $k = 757$, and $w = 255$.

8 Cryptanalysis of Regular Syndrome Decoding

In this section, we provide a thorough analysis of the regular syndrome decoding problem. While this problem has been used and analyzed in the past, we find our parameter setting to differ from the previous works. Specifically:

- In the work of [AFS03], which introduced the assumption, and its follow ups [FGS07, MDCYA11, BLPS11], the goal is to construct a collision-resistant hash function. Therefore, the parameters are chosen such that the mapping $x \to H \cdot x$ is highly non-injective. In other words, in their parameter setting, an instance (H, y) always have (usually exponentially) many solutions. This, in turns, open the avenue to powerful generalized birthday attacks (GBA), leading to rather large parameters. In contrast, in our setting, we only rely on the one-wayness of $x \to H \cdot x$ and do not require the mapping to be compressive.
- In the recent line of work on pseudorandom correlation generators [BCGI18, BCG$^+$19b, BCG$^+$19a, BCG$^+$20, YWL$^+$20, WYKW21, RS21, CRR21, BCG$^+$22], the regular syndrome decoding problem is typically used in a very particular setting: it relies on the *extremely low noise* setting, but allow the dimension K to be huge, typically of the order of 2^{20} to 2^{30}. Of course, this is the opposite of our setting, where K has a strong impact on efficiency (hence, the smaller K, the better), and using a high noise regime allows to reduce the block size, further increasing efficiency.

The work of [HOSS18] also relies on the regular syndrome decoding problem, and we found their parameter setting to be the closest to ours (though the paper handles a relatively large variety of parameter settings). A section of [HOSS18] is devoted to the analysis of RSD, and we used it as a starting point. Yet, we found this section to contain a few mistakes and imprecisions.

Below, we provide an in-depth analysis of the regular syndrome decoding assumption. Our analysis clarifies how the different parameter settings relate to the injectivity of the RSD problem, and their relation to the standard syndrome decoding problem. We also cover existing attack, explain how they adapt to the RSD setting, and design new attacks, tailored to the regular noise distribution, which outperform the attacks considered in [HOSS18].

8.1 Uniqueness Bound for Regular Syndrome Decoding

The Gilbert-Varshamov (GV) bound is the largest $\mathsf{d_{GV}}$ such that $\sum_{i=0}^{\mathsf{d_{GV}}-1} \binom{K}{i} \leq 2^k$. The GV bound provides a weight threshold for the injective setting of the syndrome decoding problem, when the constraint on the solution x is of the form $\mathsf{HW}(x) \leq w$: when $w \leq \mathsf{d_{GV}}$, the pair $(H, y = H \cdot x)$ uniquely determines x with high probability over the choice of a random sparse vector x. Of course, when we impose additional structure on the noise, we restrict the number of possible preimages, and the threshold changes. For example, if we impose an exact weight constraint $\mathsf{HW}(x) = w$ (the most standard setting for syndrome decoding), the uniqueness threshold is achieved when $\binom{K}{w} \leq 2^k$. Eventually, in the RSD setting, the uniqueness threshold is achieved when $(K/w)^w \leq 2^k$.

Expected number of solutions. When sampling a random instance $(H, y = H \cdot x)$ of the RSD problem, the expected number of solutions is given by

$$\begin{aligned}1 + \mathop{\mathbb{E}}_{H,x}\left[|\{x' \,:\, H \cdot x' = H \cdot x \,\wedge\, x' \text{ regular}\}|\right] &= 1 + \sum_{\substack{x' \neq x \\ x' \text{ regular}}} \Pr_{H,x}[H \cdot x' = H \cdot x] \\ &= 1 + \frac{1}{2^k} \cdot \left((K/w)^w - 1\right),\end{aligned}$$

where the last inequality follows from the standard fact that for any pair of vectors (x, x'), $\Pr_H[H \cdot x' = H \cdot x] = 1/2^k$. Therefore, when $((K/w)^w - 1)/2^k$ is very small, the solution x is unique with high probability; when it gets larger, the average number of solutions increases.

Intuition. In the case of the standard syndrome decoding problem, it is well-known that the hardness of the problem is maximized around the GV bound. Similarly, the natural intuition here is that the RSD problem is the hardest when $(K/w)^w \approx 2^k$. In slightly more details, the "highly injective" setting (where $(K/w)^w \ll 2^k$) cannot be harder than the threshold setting, since the adversary can always delete equations, transforming the instance (H, y) into a new instance (H', y') by removing rows of H and entries of y. As long as the adversary maintains $(K/w)^w < 2^k$, it remains likely that the solution to the new instance (H', y') is still unique. On the other hand, in the highly surjective setting (where $(K/w)^w \gg 2^k$), the number of solutions grows exponentially, and powerful attacks such as the generalized birthday attack (GBA) can be used to recover a solution.

Relation between SD and RSD depending on k and w. The GV bound and the RSD uniqueness bound allow to clarify the relation between the hardness of the standard syndrome decoding problem (SD) and the regular syndrome decoding problem (RSD). These two bounds delimit three areas depending on the relation between w and k (for a fixed value of K):

- When $2^k \geq \binom{K}{w}$, both the SD and RSD problem are injective with high probability. In this setting, the RSD problem is *easier* than the SD problem. The straightforward reduction works as follows: given an RSD instance (H, y), create a random SD instance (H', y') by shuffling the columns of H and the entries of y, and run the SD solver. By injectivity, the solution x' returned by the solver is the shuffled RSD solution x.
- When $2^k \leq (K/w)^w$, both the SD and RSD problem are surjective with high probability. In this setting, the RSD problem is *harder* than the SD problem. This is because a regular solution to a random SD instance is in particular a solution for the SD problem. More formally, when $2^k \ll (K/w)^w$, one can show that the distribution of random SD instances and RSD instances become both statistically close to uniform. Precisely, a simple calculation shows that their statistical distance to the uniform distribution is at most $\sqrt{\varepsilon}/2$, where $\varepsilon = 2^k/(K/w)^w$ for RSD, and $\varepsilon = 2^k/\binom{K}{w}$ for SD (see *e.g.* [DA19] for a proof of this statement in the case of SD; the adaptation to RSD is immediate). Therefore, a random SD instance is distributed in particular as a random RSD instance, and running an RSD solver returns a regular solution, which is in particular a valid solution to the SD instance.
- Eventually, when $(K/w)^w \leq 2^k \leq \binom{K}{w}$, the relation between SD and RSD becomes unclear, and their hardness is not directly comparable. In this setting, SD can be attacked efficiently using GBA, while RSD cannot; on the other hand, variants of information set decoding attacks (ISD) can be tuned to benefit from the regularity of the solution.

Furthermore, as we mentioned previously, each of SD and RSD is conjectured to be optimally hard on their respective boundaries. The above discussion is summarized on Figure 7. We also indicate where our parameter choice fits in this picture: as we discussed in Section 3, we set $w = K/6$, which is optimal efficiency-wise for our zero-knowledge proof. We choose k to guarantee that with overwhelming probability, the only f-weakly valid solution to RSD is a regular solution, to get security under the standard RSD assumption for our signature scheme. This setting corresponds to being "in the gray zone", not too far from the RSD uniqueness bound.

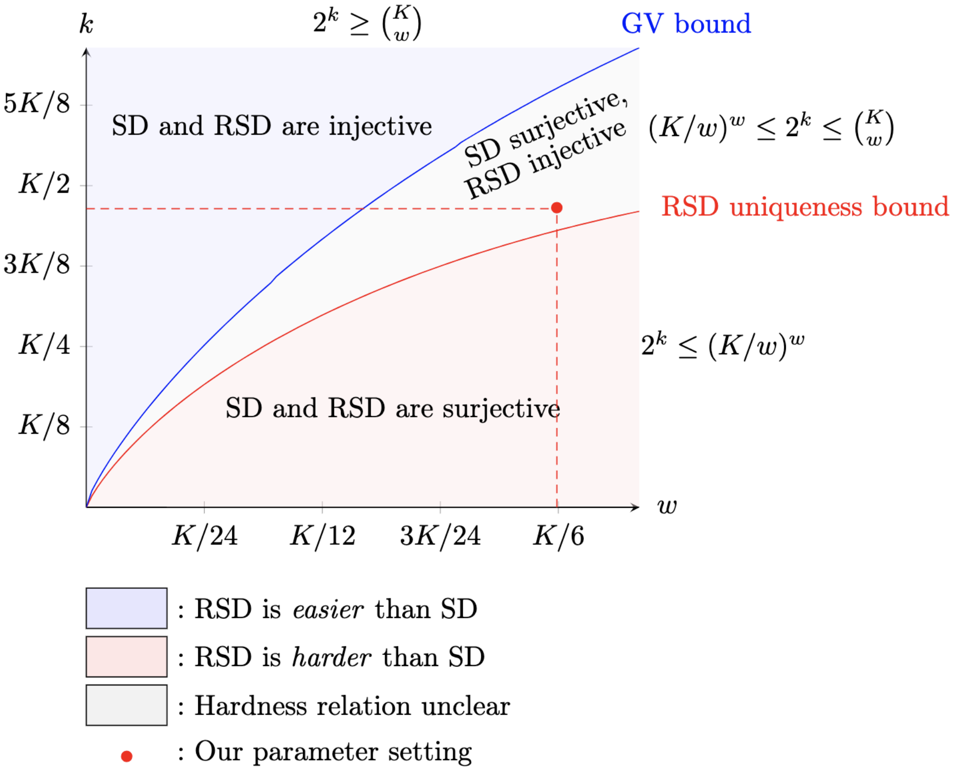

Fig. 7. Behaviour of SD and RSD when w and k vary, for a fixed value of K

8.2 Known Attacks against RSD

We now turn our attention to existing attacks against the regular syndrome decoding assumption. A detailed overview was provided in [HOSS18]. Their work covers three types of attacks: linearization attacks, generalized birthday attacks (GBA), and information set decoding (ISD). Unfortunately, it seems that most of their description confuses the RSD uniqueness bound with the GV bound. This might stem from the fact that, in their setting, they consider both the SD and the RSD problems. Furthermore, their attempt to cover both SD and RSD attacks in a single shot resulted in missing opportunities for optimizing the attack by exploiting the regular structure of the noise. In the next section, we revisit each these three attacks, and when they can be used in our setting, clarify their exact complexity, and tailor them to RSD.

Generalised Birthday Attack. GBA attacks refer to a family of algorithms based on Wagner's divide-and-conquer algorithm for solving generalized birthday problems. It was first applied to decoding problems by Coron and Joux in [CJ04], and subsequently improved in [MS09, BLN$^+$09, Kir11, NCB11]. GBA attacks can be straightforwardly adapted to RSD. However, they are tailored to the setting where the problem admits a large number of solutions. Since we focus in our instantiations on the setting where the average number of solutions is close to 1, GBA attacks do not apply to our choices of parameters, and we refrain from providing a full description of these attacks.

Linearization Attack. The linearization attack was introduced by Saarinen in [Saa07]. Using the regular structure of the noise, we can improve the attack by adding a preprocessing phase that reduces the size of the matrix H. The basic idea is to proceed by dividing the matrix into blocks of K/w columns of H. Take the first column of each block, and FOX it to all other columns, getting a new matrix H'. Additionally, XOR the first column of each block of H to y, getting a new syndrome y'. Let x be any regular solution to $H \cdot x = y$. Observe that

- it still holds that $H' \cdot x = y'$. This is because each block of x has weight 1, hence a single copy of each first column of a block is added to the result when computing $H' \cdot x$;

– the first column of each block of H' is now identically 0, and can therefore be deleted.

After performing this process, we get a new matrix $H' \in \mathbb{F}_2^{k \times K-w}$, and a new target solution $x' \in \mathbb{F}_2^{K-w}$ (which is x with the first entry of each block deleted – this still uniquely specifies x). The initial equality becomes $H'x' = y'$.

The goal now becomes finding a solution x' where each block (of size $K/w - 1$) has weight *at most* one (in deleting some of the entries, we may have eliminated some non-zero entries).

In the next step, we define two new matrix $H_2 \in \mathbb{F}_2^{k \times (K-w-k)}$ whose columns are randomly selected from columns of the parity check matrix H' and $H_1 \in \mathbb{F}_2^{k \times k}$ whose columns are the remaining ones in H'. We have $y' = H'x' = H_1x_1 + H_2x_2$ with $x_1 \in \mathbb{F}_2^k$ and $x_2 \in \mathbb{F}_2^{K-w-k}$. At this stage, whenever $x_2 = 0$, we get $y' = H_1x_1$, where H is a square matrix, which is invertible with high probability. Therefore, we compute the solution x_1, recover x', and check that we found the right solution; otherwise, we start again with a new random choice of H_1, H_2. The full description of the attack is given on Figure 8.

Inputs: A matrix $H \in \mathbb{F}_2^{k \times K}$ a syndrome $y \in \mathbb{F}_2^k$ and a value $w \in \mathbb{N}$.
Output: A regular vector $x \in \mathbb{F}_2^K$ with Hamming weight $\mathsf{HW}(x) = w$ such that $Hx = y$.

1. **Precomputing phase:**
   - In each block of K/w columns of H, delete the first column and XOR it to all remaining columns of the block. Denote the new matrix by H'.
   - XOR all deleted columns to the syndrome y, obtaining a new syndrome y'.

   The goal is now to find a vector x' of length $K - w$ with w blocks, each of Hamming weight 0 or 1, such that $H'x' = y'$.
2. **Permutation phase:**
   - Write $H' = [H_1|H_2]$ and $y' = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$ where $H_1 \in \mathbb{F}_2^{k \times k}$.
   - If $H_1^{-1}y'$ is a sparse vector, set $x_1 = H_1^{-1}y'$. Otherwise, shuffle the columns of H' and the entries of y', and start again the permutation phase.

Fig. 8. A variant of linearization attacks tailored to RSD

We compute the expected cost of the above attack:

$$\mathbb{E}[\mathsf{cost}] = \frac{\text{cost per iteration}}{\text{success proba per iteration}}.$$

The construction of the matrix H_2 requires choosing of $K-w-k$ columns out of the of total $K-w$ columns of the matrix H'. By picking the same number of columns of H_2 in each block, we must take $(K - w - k)/w = K/w - 1 - k/w$ columns from each block. The attack succeeds if all selected columns correspond to zero-entries in the solution vector x', that is, if $x_2 = 0$. We now compute the probability that this is the case.

By construction, each block of x' has Hamming weight 0 with probability w/K, and Hamming weight 1 otherwise. Hence, the expected number of blocks with Hamming weight 1 is $(1 - w/K) \cdot w$, and the attack succeeds if the $K/w - 1 - k/w$ columns selected in each of these blocks are a subset of the $K/w - 2$ 0-columns. Therefore, the probability of picking only 0 entries is well approximated by

$$P = \left[\binom{K/w - 2}{K/w - 1 - k/w} \Big/ \binom{K/w - 1}{K/w - 1 - k/w}\right]^{(1-w/K)\cdot w}.$$

In each iteration, the cost is dominated by the cost of solving a $k \times k$ random system of linear equations, which takes about k^3 arithmetic operations (for realistic values of k, up to a few hundreds), or $k^{2.7}$ (for larger value, using Strassen's algorithm). This yields a total expected cost $\mathbb{E}[\mathsf{cost}] = k^\omega/P$ (with ω an appropriate matrix multiplication exponent).

Information Set Decoding Attacks. Information Set Decoding (ISD) is a decoding technique to solve the syndrome decoding problem for linear codes. This type of algorithm was initially introduced in 1962 by Prange [Pra62] and subsequently improved, first through a polynomial improvement proposed by Lee and Leon [LB88, Leo88] and then by an exponential improvement by Stern [Ste88] followed by numerous other improvements [FS09, BLP11, MMT11, BJMM12, MO15].

The structure of ISD attacks is tailored to the structure of the target noise vector. Therefore, adapting ISD attacks to the RSD setting is not always straightforward. In particular, we observe that the most efficient and recent variants of ISD [BJMM12, MO15] make use of the representation technique. However, the representation technique is tailored to the standard syndrome decoding problem, and does not appear to bring any improvement when we consider *regular* noise: the representation technique would be well suited, for example, if different blocks can have (say) one of two possible weights. In this case, it could introduce additional equation capturing how the weights are distributed. However, in the context of RSD, all blocks have the same Hamming weight 1, and a direct application of the representation technique yields no improvement over a standard birthday algorithm. We view as an interesting question, and leave to future work, the possibility of finding an alternative, indirect way of using the representation technique to improve the attack.

However, it appears that pre-representation technique ISD variants can be adapted to the RSD setting. Furthermore, this adaptation significantly simplifies and improves the attack. At a high level, this is because knowing that the noise is regular significantly reduces the search space. More formally, we start from the Generalized ISD algorithm defined by Finiasz and Sendrier [FS09]. This is the same variant which was discussed in [HOSS18]; however, their description appears to confuse the GV bound with the RSD uniqueness bound. Furthermore, they did not provide a complete description of the attack and apparently missed a few natural optimizations to reduce the search space in the RSD setting (probably because they aimed for a unified description covering both SD and RSD).

Fix an RSD instance (H, y) with noise weight w. The algorithm is a two-step algorithm, with a first step consisting of a Gaussian elimination (which includes a permutation) resulting in a new instance with a new noise vector x', and a second step in which the noise vector is broken into two vectors $x' = x_1 || x_2$ (the purpose being to later use the birthday paradox to find a candidate solution x_1). The Finiasz-Sendrier algorithm proceeds by searching for two vectors of the same length as x_1 but with half weight and whose sum is x_2 (using a procedure called *submatrix matching*). In the following, we describe several improvements to this attack.

Our first observation relates to the choice partition of x_1 as a sum of two vectors of the same length, rather than executing the birthday search over vectors whose *concatenation* is x_1. This choice stems from the random permutation used in the first step: although the initial error x has a regular structure, when it is randomly permuted, it loses its regularity. This implies that if one wanted to write x_1 as a concatenation of two half-length vectors, it would not be guaranteed that they actually have half of the total Hamming weight each. This would induce an additional overhead, since one would have to consider all possible partitions of the weight w.

We observe that in fact, our setting allows to avoid this problem. Since we directly target the setting where the RSD matrix is given in systematic form $H = [H' | I_k]$, we do not need to use a random permutation of H; rather, we can directly worth with the initial regular noise vector x. This regular structure can be exploited: as the weight is perfectly balanced within each block of the error, we can rewrite it as a concatenation of two vectors $x = x_1 || x_2$ with weight $w/2$ each. This significantly reduces the length of the target vector x_1, which we search through a birthday paradox, leading to an important reduction of the total cost of the attack. We provide a full description of the attack on Figure 9

Let's now analyze the cost of this attack. The creation of the two lists $\bar{L}$ and $\tilde{L}$ requires, for all the $6^{\frac{r+q}{K/w}}$ possible vectors of length $\frac{r+q}{2}$ and weight $\frac{r+q}{2K/w}$, a matrix-vector multiplication $\bar{R} \cdot e_1, \tilde{R} \cdot e_2$. The total cost for this step is therefore about $(K/w)^{\frac{r+q}{2K/w}} \cdot q \cdot \frac{r+q}{2K/w}$. The search for a collision between the two lists takes linear time. Since both lists are of length $(K/w)^{\frac{r+q}{2K/w}}$, this brings to the total cost a value of $(K/w)^{\frac{r+q}{2K/w}} \cdot q \cdot \frac{r+q}{2K/w}$. The last step in which for each collision we check the weight by doing a matrix-vector multiplication costs $\frac{(K/w)^{r+q}}{2^q} \cdot (k-q) \cdot \frac{r+q}{K/w}$. So the total cost of the attack presented in 9 is roughly

$$\left[q \cdot \frac{r+q}{K/w} \cdot (K/w)^{\frac{r+q}{12}} \right] + \left[\frac{(K/w)^{\frac{r+q}{K/w}}}{2^q} \cdot (k-q) \cdot \frac{r+q}{K/w} \right] \tag{3}$$

To maximize the possibility of finding a collision between the two lists and at the same time the possibility that the check on the weight in the final step has a positive result, it is necessary that there

Inputs: A matrix $H \in \mathbb{F}_2^{k \times K}$ in standard form $H = [H'|I_k]$, a syndrome $y \in \mathbb{F}_2^k$ and a value $w \in \mathbb{N}$.
Output: A vector $x \in \mathbb{F}_2^K$ with Hamming weight $\mathsf{HW}(x) = w$ and such that $Hx = y$.
Parameters: $0 \le q \le k$ and $0 \le p \le r + q$ with $r = K - k$.
repeat

1. Rewriting the matrix

$$H = [H'|I_k] = \left[H' \left| \begin{array}{c|c} I_q & 0 \\ \hline 0 & I_{k-q} \end{array} \right. \right] = \left[\begin{array}{c|c|c} H_1 & I_q & 0 \\ \hline H_2 & 0 & I_{k-q} \end{array} \right] = \left[\begin{array}{c|c} R_1 & 0 \\ \hline R_2 & I_{k-q} \end{array} \right]$$

   where $R_1 = [H_1|I_q] \in \mathbb{F}_2^{q \times r+q}$ and $R_2 = [H_2|0] \in \mathbb{F}_2^{k-q \times r+q}$ and the vectors

$$y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$

   where $y_1 \in \mathbb{F}_2^q, y_2 \in \mathbb{F}_2^{k-q}, x_1 \in \mathbb{F}_2^{r+q}, x_2 \in \mathbb{F}_2^{k-q}$.
2. Computing submatrix problem, solving the reduced problem $R_1 x_1 = y_1$ where $\mathsf{HW}(x_1) = p = \frac{r+q}{K/w}$:
   - Rewrite $R_1 = \bar{R}||\tilde{R}$
   - For all the words $e_1 \in \mathbb{F}_2^{\frac{r+q}{2}}$ of weight $\frac{p}{2} = \frac{r+q}{2K/w}$, create the list $\bar{L} = \{\bar{R}e_1\}$;
   - For all the words $e_2 \in \mathbb{F}_2^{\frac{r+q}{2}}$ of weight $\frac{p}{2} = \frac{r+q}{2K/w}$, create the list $\tilde{L} = \{\tilde{R}e_2 + y_1\}$;
   - Search for a collision between $\bar{L}$ and $\tilde{L}$;
   - For each collision (e_1, e_2) if $\mathsf{HW}(y_2 + R_2(e_1||e_2)) = w - p = \frac{k-q}{K/w}$ then set $x_1 = e_1||e_2$.
3. Extend solution computing $x_2 = y_2 + R_2 x_1$

Fig. 9. An Information Set Decoding algorithm tailored to the RSD setting with matrix in systematic form

is no imbalance between the cost of step 2 and the cost of step 3. This can be ensured by requiring that the two parts of 3 balance each other. Hence, by imposing $q \cdot \frac{r+q}{K/w} \cdot (K/w)^{\frac{r+q}{2K/w}} = \frac{(K/w)^{\frac{r+q}{K/w}}}{2^q} \cdot (k-q) \cdot \frac{r+q}{K/w}$ we obtain

$$q\left(1 - \frac{\log_2(K/w)}{2K/w}\right) + \log_2\left(\frac{q}{k-q}\right) = (K-k)\left(\frac{\log_2(K/w)}{2K/w}\right) \tag{4}$$

Solving 4 for q and injecting the resulting value in 3 yields the lowest possible cost for the attack.

8.3 An Approximate Birthday Paradox Attack

In this section, we describe a new attack on regular syndrome decoding. Informally, our attack resemble the ISD variant described in the previous section, but applies the birthday paradox agorithm at a different place, removing the need for optimizing over the choice of a value q altogether. In exchange, the attack requires more than a standard birthday collision search: it requires an *approximate* collision search between two lists (where approximate means that the goal is two find two strings whose XOR is a sparse regular vector – that is, the target strings are identical up to differing in exactly one coordinate per block).

Approximate collision searches have been previously assumed to take linear time as well in the literature. We stress, however, that we do not actually know of a strict linear time algorithm (in the size of the lists) for solving our variant of the approximate collision search. In fact, we view finding such an efficient algorithm as an interesting open problem. However, for the sake of being conservative in our parameter choice, we *assume* that an approximate collision can be found in the (linear) time it takes to scan the lists. We note that the existence of a strictly linear time algorithm appear far from obvious, and our assumption might therefore be overly conservative.

The basic attack. We elaborate on the idea behind our attack below. Starting from the RSD instance (H, y) where $H \in \mathbb{F}_2^{k \cdot K}$ and $y \in \mathbb{F}_2^k$ with solution $x \in \mathbb{F}_2^K$ of weight w, we rewrite H and x as follows:

$$H = [H'|I_k] = \left[\bar{H}|\tilde{H}|I_k\right] \text{ and } x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \bar{x} \\ \tilde{x} \\ x_2 \end{bmatrix}$$

where $\bar{x}, \tilde{x} \in \mathbb{F}_2^{\frac{r}{2}}$, and $x_2 \in \mathbb{F}_2^k$. From there, the attack proceeds in two simple steps:

1. Search step of an "almost collision" between two lists: we have

   $$y = Hx = \left[\bar{H}|\tilde{H}|I_k\right] \begin{bmatrix} \bar{x} \\ \tilde{x} \\ x_2 \end{bmatrix} = \bar{H}\bar{x} + \tilde{H}\tilde{x} + x_2$$

   This implies that

   $$\left(\tilde{H}\tilde{x} + y\right) + \left(\bar{H}\bar{x}\right) = x_2 \qquad (5)$$

   Now, since x_2 is a sparse vector, the problem is reduced to finding two vectors $\bar{x}, \tilde{x}$ such that $\left(\tilde{H}\tilde{x} + y\right) + \left(\bar{H}\bar{x}\right)$ is sparse. We can therefore find $\tilde{x}, \bar{x}$ by storing all possible values of $\left(\tilde{H}\tilde{x} + y\right)$ and $\left(\bar{H}\bar{x}\right)$ in respective lists $\tilde{L}$ and $\bar{L}$, and searching for an almost-collision between the two lists.
2. Extend the partial solution obtained in the previous step: given $(\bar{x}, \tilde{x})$, set $x_1 = \begin{bmatrix} \bar{x} \\ \tilde{x} \end{bmatrix}$ and compute x_2 as $H'x_1 + y$.

Improving the attack by extending the matrix. We can improve the basic attack by letting the adversary append additional equation to the parity-check matrix, and bring it back to systematic form afterwards (this requires only row operations and does not change the structure of the solution). At a high-level, each additional equation encompasses the information that each block of the target noise vector has odd Hamming weight. Hence, letting h'_i denote the vector whose entries are all 0, except for a "band of ones" in the i-th block, the inner product between h' and x must be equal to 1. This lets the adversary add w additional rows to the matrix H before running the above attack. We describe the full attack on Figure 10

Inputs: A matrix $H \in \mathbb{F}_2^{(k+w)\times K}$ with $H = [H'|M]$, a syndrome $y \in \mathbb{F}_2^{k+w}$ and a value $w \in \mathbb{N}$. We assume that the matrix H was extended with w additional rows and that y was extended with ones, capturing the fact that each block of x has odd Hamming weight.
Output: A regular vector $x \in \mathbb{F}_2^K$ with Hamming weight $\mathsf{HW}(x) = w$ such that $Hx = y$.
Parameters: $r = K - k - w$

```
1. Write $H' = \bar{H}|\tilde{H}$;
2. Search for an almost collision
    – Using all regular words $\bar{x} \in \mathbb{F}_2^{\frac{r}{2}}$ of weight $\frac{r}{2K/w}$, create the list $\bar{L} = \{M^{-1} \cdot \bar{H}\bar{x}\}$;
    – Using all regular words $\tilde{x} \in \mathbb{F}_2^{\frac{r+q}{2}}$ of weight $\frac{r}{2K/w}$, create the list $\tilde{L} = \{M^{-1} \cdot (\tilde{H}\tilde{x} + y)\}$;
    – Search for an almost collision $(\bar{x}, \tilde{x})$ between $\bar{L}$ and $\tilde{L}$ and set $x_1 = \bar{x}|\tilde{x}$;
3. Extend the solution by computing $x_2 = y + H'x_1$.
```

Fig. 10. An approximate birthday paradox attack on the regular syndrome decoding problem

Cost of the attack. The second step of the attack, the creation of the lists $\bar{L}$ and $\tilde{L}$ of size $(K/w)^{\frac{r}{2K/w}}$ each, costs $2 \cdot (K/w)^{\frac{r}{2K/w}} \cdot \frac{r}{2K/w} \cdot k = (K/w)^{\frac{r}{2K/w}-1} \cdot r \cdot k$. We assume for the sake of being conservative that finding an almost-collison takes linear time, and bound the cost as

$$\mathsf{cost} \geq (K/w)^{\frac{r}{2K/w}-1} \cdot r \cdot k \qquad (6)$$

A note on optimizing security. In our parameter setting, the above attack significantly outperforms the ISD variant and the linearization attack described in the previous section. Furthermore, the larger the value of k, the more efficient the attack gets. Therefore, the optimal parameter choice is obtained by minimizing the value of k, while maintaining the constraint that the resulting RSD instance does not have too many solutions with high probability (otherwise, GBA attacks could apply). This is in line with the intuition that optimal parameter choices belong to the RSD uniqueness curve (see Figure 7).

8.4 From Regular Syndrome Decoding to Almost-Regular Syndrome Decoding

The security of our new signature scheme does not in fact reduce directly to the regular syndrome decoding problem, but rather to an *almost-regular* syndrome problem. This stems from the fact that our zero-knowledge proof of knowledge of Section 4 has a soundness gap: while an honest witness is assumed to be a standard regular vector, the soundness only guarantees that an f*-almost-regular* (or f-almost-antiregular) vector can be extracted from a malicious prover. Here, f-almost-regular means that the extracted vector x is divided in w blocks such that all blocks except f have Hamming weight 1, and the f remaining block can have any odd weight (in our setting, weight 1, 3, or 5, since we settle for a block size 6).

This variant of RSD, which we informally call f*-almost-RSD*, is in itself a plausible assumption, albeit a somewhat exotic one. In our main choice of parameters for the signature scheme, we aim to reduce security to the standard RSD assumption instead. To do so, we use a parameter setting where almost-RSD is in fact *equivalent* to RSD. Concretely, we pick the instance (K, k, w) with two constraints: first, we enforce $K/w = 6$, and second, we set

$$k \leftarrow \left\lceil \log_2 \left(\sum_{i=0}^{f} 6^{w-i} \cdot \binom{w}{i} \cdot 26^i \right) \right\rceil + \lambda.$$

This guarantees $\sum_{i=0}^{f} 6^{w-i} \cdot \binom{w}{i} \cdot 26^i \leq 2^{k-\lambda}$. Intuitively, without the $+\lambda$ term, the above would correspond to the f*-almost-RSD uniqueness bound*, *i.e.* the threshold above which there is on average a single almost-regular solution to a random RSD instance (the left hand side in the inequality corresponds to the total number of regular vectors with $w - f$ blocks of weight 1, and f blocks of weight 1, 3, or 5 – that is, the space of solutions with the right structure). Let (H, y) be a random RSD instance, where y is computed as $H \cdot x$ for a random regular vector x. With the λ term, the inequality guarantees that the probability that there exists any additional solution beyond x is at most $2^{-\lambda}$. In this regime, the almost-RSD problem becomes clearly equivalent to the standard RSD problem: except with probability $2^{-\lambda}$, any almost-regular solution to the problem has to be the only regular solution.

References

ABG$^+$19. N. Aragon, O. Blazy, P. Gaborit, A. Hauteville, and G. Zémor. Durandal: A rank metric based signature scheme. In *EUROCRYPT 2019, Part III*, *LNCS* 11478, pages 728–758. Springer, Heidelberg, May 2019.

AES01. Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce, November 2001.

AFS03. D. Augot, M. Finiasz, and N. Sendrier. A fast provably secure cryptographic hash function. Cryptology ePrint Archive, Report 2003/230, 2003. https://eprint.iacr.org/2003/230.

BCG$^+$19a. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, P. Rindal, and P. Scholl. Efficient two-round OT extension and silent non-interactive secure computation. In *ACM CCS 2019*, pages 291–308. ACM Press, November 2019.

BCG$^+$19b. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. Efficient pseudorandom correlation generators: Silent OT extension and more. In *CRYPTO 2019, Part III*, *LNCS* 11694, pages 489–518. Springer, Heidelberg, August 2019.

BCG$^+$20. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, and P. Scholl. Efficient pseudorandom correlation generators from ring-LPN. In *CRYPTO 2020, Part II*, *LNCS* 12171, pages 387–416. Springer, Heidelberg, August 2020.

BCG$^+$22. E. Boyle, G. Couteau, N. Gilboa, Y. Ishai, L. Kohl, N. Resch, and P. Scholl. Correlated pseudorandomness from expand-accumulate codes. In *CRYPTO*, 2022. to appear.

BCGI18. E. Boyle, G. Couteau, N. Gilboa, and Y. Ishai. Compressing vector OLE. In *ACM CCS 2018*, pages 896–912. ACM Press, October 2018.

BdK$^+$21. C. Baum, C. de Saint Guilhem, D. Kales, E. Orsini, P. Scholl, and G. Zaverucha. Banquet: Short and fast signatures from AES. LNCS, pages 266–297. Springer, Heidelberg, 2021.

BDLN16. C. Baum, I. Damgård, K. Larsen, and M. Nielsen. How to prove knowledge of small secrets. 2016. https://eprint.iacr.org/2016/538.

Beu20. W. Beullens. Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In *EUROCRYPT 2020, Part III*, *LNCS* 12107, pages 183–211. Springer, Heidelberg, May 2020.

BG93. M. Bellare and O. Goldreich. On defining proofs of knowledge. In *CRYPTO'92*, *LNCS* 740, pages 390–420. Springer, Heidelberg, August 1993.

BGI14. E. Boyle, S. Goldwasser, and I. Ivan. Functional signatures and pseudorandom functions. In *PKC 2014*, *LNCS* 8383, pages 501–519. Springer, Heidelberg, March 2014.

BGKM22. L. Bidoux, P. Gaborit, M. Kulkarni, and V. Mateu. Code-based signatures from new proofs of knowledge for the syndrome decoding problem. *arXiv preprint arXiv:2201.05403*, 2022.

BJMM12. A. Becker, A. Joux, A. May, and A. Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In *EUROCRYPT 2012*, *LNCS* 7237, pages 520–536. Springer, Heidelberg, April 2012.

BLN$^+$09. D. J. Bernstein, T. Lange, R. Niederhagen, C. Peters, and P. Schwabe. FSBday. In *INDOCRYPT 2009*, *LNCS* 5922, pages 18–38. Springer, Heidelberg, December 2009.

BLP11. D. J. Bernstein, T. Lange, and C. Peters. Smaller decoding exponents: Ball-collision decoding. In *CRYPTO 2011*, *LNCS* 6841, pages 743–760. Springer, Heidelberg, August 2011.

BLPS11. D. J. Bernstein, T. Lange, C. Peters, and P. Schwabe. Really fast syndrome-based hashing. In *AFRICACRYPT 11*, *LNCS* 6737, pages 134–152. Springer, Heidelberg, July 2011.

BW13. D. Boneh and B. Waters. Constrained pseudorandom functions and their applications. In *ASIACRYPT 2013, Part II*, *LNCS* 8270, pages 280–300. Springer, Heidelberg, December 2013.

CDXY16. R. Cramer, I. Damgard, C. Xing, and C. Yuan. Amortized complexity of zero-knowledge proofs revisited: Achieving linear soundness slack. 2016. `https://eprint.iacr.org/2016/681`.

CJ04. J.-S. Coron and A. Joux. Cryptanalysis of a provably secure cryptographic hash function. Cryptology ePrint Archive, Report 2004/013, 2004. `https://eprint.iacr.org/2004/013`.

CKY09. J. Camenisch, A. Kiayias, and M. Yung. On the portability of generalized schnorr proofs. 2009. `https://eprint.iacr.org/2009/050`.

CRR21. G. Couteau, P. Rindal, and S. Raghuraman. Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. LNCS, pages 502–534. Springer, Heidelberg, 2021.

DA19. T. Debris-Alazard. *Cryptographie fondée sur les codes: nouvelles approches pour constructions et preuves; contribution en cryptanalyse.* PhD thesis, Sorbonne université, 2019.

DST19. T. Debris-Alazard, N. Sendrier, and J.-P. Tillich. Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In *ASIACRYPT 2019, Part I*, *LNCS* 11921, pages 21–51. Springer, Heidelberg, December 2019.

EGK$^+$20. D. Escudero, S. Ghosh, M. Keller, R. Rachuri, and P. Scholl. Improved primitives for MPC over mixed arithmetic-binary circuits. In *CRYPTO 2020, Part II*, *LNCS* 12171, pages 823–852. Springer, Heidelberg, August 2020.

FGS07. M. Finiasz, P. Gaborit, and N. Sendrier. Improved fast syndrome based cryptographic hash functions. In *Proceedings of ECRYPT Hash Workshop*, page 155. Citeseer, 2007.

FJR21. T. Feneuil, A. Joux, and M. Rivain. Shared permutation for syndrome decoding: New zero-knowledge protocol and code-based signature. Cryptology ePrint Archive, Report 2021/1576, 2021. `https://eprint.iacr.org/2021/1576`.

FJR22a. T. Feneuil, A. Joux, and M. Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In *CRYPTO 2022*, 2022.

FJR22b. T. Feneuil, A. Joux, and M. Rivain. Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. *Cryptology ePrint Archive*, 2022.

FS87. A. Fiat and A. Shamir. How to prove yourself: Practical solutions to identification and signature problems. In *CRYPTO'86*, *LNCS* 263, pages 186–194. Springer, Heidelberg, August 1987.

FS09. M. Finiasz and N. Sendrier. Security bounds for the design of code-based cryptosystems. In *ASIACRYPT 2009*, *LNCS* 5912, pages 88–105. Springer, Heidelberg, December 2009.

Gal68. R. G. Gallager. *Information theory and reliable communication.* Springer, 1968.

GGM86. O. Goldreich, S. Goldwasser, and S. Micali. How to construct random functions. *Journal of the ACM*, 33(4):792–807, October 1986.

GPS21. S. Gueron, E. Persichetti, and P. Santini. Designing a practical code-based signature scheme from zero-knowledge proofs with trusted setup. Cryptology ePrint Archive, Report 2021/1020, 2021. `https://eprint.iacr.org/2021/1020`.

HOSS18. C. Hazay, E. Orsini, P. Scholl, and E. Soria-Vazquez. TinyKeys: A new approach to efficient multi-party computation. In *CRYPTO 2018, Part III*, *LNCS* 10993, pages 3–33. Springer, Heidelberg, August 2018.

IKOS07. Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. Zero-knowledge from secure multiparty computation. In *39th ACM STOC*, pages 21–30. ACM Press, June 2007.

Kir11. P. Kirchner. Improved generalized birthday attack. Cryptology ePrint Archive, Report 2011/377, 2011. `https://eprint.iacr.org/2011/377`.

KKW18. J. Katz, V. Kolesnikov, and X. Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In *ACM CCS 2018*, pages 525–537. ACM Press, October 2018.

KPTZ13. A. Kiayias, S. Papadopoulos, N. Triandopoulos, and T. Zacharias. Delegatable pseudorandom functions and applications. In *ACM CCS 2013*, pages 669–684. ACM Press, November 2013.

KZ20. D. Kales and G. Zaverucha. An attack on some signature schemes constructed from five-pass identification schemes. In *CANS 20*, *LNCS* 12579, pages 3–22. Springer, Heidelberg, December 2020.

LB88. P. J. Lee and E. F. Brickell. An observation on the security of mceliece's public-key cryptosystem. In *Workshop on the Theory and Application of of Cryptographic Techniques*, pages 275–280. Springer, 1988.

Leo88. J. S. Leon. A probabilistic algorithm for computing minimum weights of large error-correcting codes. *IEEE Transactions on Information Theory*, 34(5):1354–1359, 1988.

MDCYA11. M. Meziani, Ö. Dagdelen, P.-L. Cayrel, and S. M. E. Yousfi Alaoui. S-fsb: An improved variant of the fsb hash family. In *International Conference on Information Security and Assurance*, pages 132–145. Springer, 2011.

MMT11. A. May, A. Meurer, and E. Thomae. Decoding random linear codes in $\tilde{\mathcal{O}}(2^{0.054n})$. In *ASIACRYPT 2011*, *LNCS* 7073, pages 107–124. Springer, Heidelberg, December 2011.

MO15. A. May and I. Ozerov. On computing nearest neighbors with applications to decoding of binary linear codes. In *EUROCRYPT 2015, Part I*, *LNCS* 9056, pages 203–228. Springer, Heidelberg, April 2015.

MS09. L. Minder and A. Sinclair. The extended k-tree algorithm. In *20th SODA*, pages 586–595. ACM-SIAM, January 2009.

MSY21. J.-P. Münch, T. Schneider, and H. Yalame. VASA: Vector AES instructions for security applications. Cryptology ePrint Archive, Report 2021/1493, 2021. `https://eprint.iacr.org/2021/1493`.

NCB11. R. Niebuhr, P.-L. Cayrel, and J. Buchmann. Improving the efficiency of generalized birthday attacks against certain structured cryptosystems. In *WCC 2011-Workshop on coding and cryptography*, pages 163–172, 2011.

Pra62. E. Prange. The use of information sets in decoding cyclic codes. *IRE Transactions on Information Theory*, 8(5):5–9, 1962.

RS21. P. Rindal and P. Schoppmann. VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. LNCS, pages 901–930. Springer, Heidelberg, 2021.

RW19. D. Rotaru and T. Wood. MArBled circuits: Mixing arithmetic and Boolean circuits with active security. Cryptology ePrint Archive, Report 2019/207, 2019. `https://eprint.iacr.org/2019/207`.

Saa07. M.-J. O. Saarinen. Linearization attacks against syndrome based hashes. In *INDOCRYPT 2007*, *LNCS* 4859, pages 1–9. Springer, Heidelberg, December 2007.

Sho94. P. W. Shor. Algorithms for quantum computation: Discrete logarithms and factoring. In *35th FOCS*, pages 124–134. IEEE Computer Society Press, November 1994.

Ste88. J. Stern. A method for finding codewords of small weight. In *International Colloquium on Coding Theory and Applications*, pages 106–113. Springer, 1988.

Ste94. J. Stern. Designing identification schemes with keys of short size. In *CRYPTO'94*, *LNCS* 839, pages 164–173. Springer, Heidelberg, August 1994.

WYKW21. C. Weng, K. Yang, J. Katz, and X. Wang. Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. pages 1074–1091. IEEE Computer Society Press, 2021.

YWL$^+$20. K. Yang, C. Weng, X. Lan, J. Zhang, and X. Wang. Ferret: Fast extension for correlated OT with small communication. In *ACM CCS 20*, pages 1607–1626. ACM Press, November 2020.

Supplementary Material

A Python Script for Computing the Combinatorial Bound

We provide below a script computing the combinatorial bound, according to the formula given in Lemma 7. The script focuses on computing the first term in the minimum, since the other two terms have a closed-form formula and can be computed with a straightforward calculation.

```
import math
from scipy.misc import comb

def Func_u(N,H,x,i,j): # x corresponds to theta in the writeup, and H to ell
                                       in the writeup
  val = (6**j)*comb(H,j,exact=True)*comb(N/6-H,i-j,exact=True)*comb(N-6*i,x+j
                                       -6*i,exact=True)
  return val

def minproba(N,X,f): # f denotes the maximum number of faulty blocks allowed
                                       in the witness
  proba = 0
  u = 0
  val = [0 for k in range(N/6-1)]
  for i in range(1,N/6+1):
    for j in range(i+1):
      u = Func_u(N,1,X,i,j)
      val[0] += (-1)**(i+1)*u
      for H in range(1,N/6-1):
        if H != j-1:
          u = u*(H+1)*(N/6-H-i+j)/((N/6-H)*(H+1-j))
        else:
          u = Func_u(N,H+1,X,i,j)
        val[H] += (-1)**(i+1)*u

  for H in range(N/6-1):
    val[H] = math.exp(math.log(val[H]) - math.log(comb(N,X, exact=True)))

  return min(val[f-1:N/6-f])

if __name__ == '__main__':

  f = 12  # number of faulty blocks allowed
  K = 1776  # set the value of K here
  d = 710   # indicates the number of values of theta to try, from K-d to K-(
                                       f+1).
        # to validate a parameter set, all values from 0.6*K to K-(f+1) must
                                       be verified
        # we conjecture that the best value is always K-(f+1)

  for x in range(K-d, K-f):
    p = 1-minproba(K,x,f)
    if p==0.0:    #if the failure probability of the adversary, given by
                                       minproba, is too close to 1 and Python
                                        rounds it to 1.0
      print("pmin("+str(x)+") ~= 0")
    else:
      val = int(1/p)
      print("pmin("+str(x)+") = 1/"+str(val))
```