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 [23] 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 [35] 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 [34]. 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 [8, 9, 20, 21, 25]. Indeed, while some alternative approaches such as Wave [18] and Durandal [2] 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 [27] (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,

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

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 [5, 29], 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 [21].

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 [3] as the assumption underlying the FSB candidate to the NIST hash function competition, and was subsequently analyzed in [7,24,30], among others. It has also been used and analyzed in many recent works on secure computation, such as [10–14, 16, 26, 32, 36, 37].

Brief overview of our approach. While we use the MPC in the head paradigm, as in previous works [9,20,21,25], 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", 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 [25, 29], 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 [21] or compositions of random permutations [9, 20]. 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 5). 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 [21], 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.

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

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 [21] 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 [20, 21]. Our timings are estimated runtimes with the methodology given in Section 5.2.

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
[25] - 256	24.0 KB	0.11 KB	-	SD over $\mathbb{F}_{256}$
[25] - 256	19.8 KB	0.12 KB	-	SD over $\mathbb{F}_{1024}$
[20] (fast)	22.6 KB	0.09 KB	13	SD over $\mathbb{F}_2$
[20] (short)	16.0 KB	0.09 KB	62	SD over $\mathbb{F}_2$
[9] Sig1	23.7 KB	0.1 KB	-	SD over $\mathbb{F}_2$
[9] Sig2	20.6 KB	0.2 KB	-	(QC)SD over $\mathbb{F}_2$
[21] - Var1f	15.6 KB	0.09 KB	-	SD over $\mathbb{F}_2$
[21] - Var1s	10.9 KB	0.09 KB	-	SD over $\mathbb{F}_2$
[21] - Var2f	17.0 KB	0.09 KB	13	SD over $\mathbb{F}_2$
[21] - Var2s	11.8 KB	0.09 KB	64	SD over $\mathbb{F}_2$
[21] - Var3f	11.5 KB	0.14 KB	6	SD over $\mathbb{F}_{256}$
[21] - 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 5.2, 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.

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 [3, 7, 10–14, 16, 24, 26, 30, 32, 36, 37] 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 [22], 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:** [6] 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 [15] 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 [27]. 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 [19, 33].

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 [29]. 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 [5], or more recently in [21].

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 $[\![x]\!]_2$ 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 $[\![\mathbf{r}]\!]_2 = (\mathbf{s}_1, \cdots, \mathbf{s}_n) \leftarrow_r \mathsf{Share}_2(\mathbf{r})$ and $[\![\mathbf{r}]\!]_T = (\mathbf{t}_1, \cdots, \mathbf{t}_n) \leftarrow_r \mathsf{Share}_T(\mathbf{r})$, 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 $[\![\mathbf{y}']\!]_2 = H \cdot [\![x]\!]_2$ and $[\![\mathbf{z}]\!]_2 = [\![\mathbf{r}]\!]_2 + [\![x]\!]_2$. All parties open $\mathbf{y}'$ and $\mathbf{z}$.
- The parties compute $[\![\mathbf{v}']\!]_T \leftarrow L \cdot (\mathbf{z} \odot [\![\mathbf{1} - \mathbf{r}]\!]_T + (\mathbf{1} - \mathbf{z}) \odot [\![\mathbf{r}]\!]_T)$, 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.

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

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 $((\mathbf{s}_1, \cdots, \mathbf{s}_n), (\mathbf{t}_1, \cdots, \mathbf{t}_n)) = (\llbracket \mathbf{r} \rrbracket_2, \llbracket \mathbf{r} \rrbracket_T)$. 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 $(\llbracket \pi(\mathbf{r}) \rrbracket_2, \llbracket \pi(\mathbf{r}) \rrbracket_T)$. 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}

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 [3] as the assumption underlying the security of a candidate for the SHA-3 competition, and was subsequently studied in numerous works, including [7, 24, 30], and more recently in [26]. 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.* [10–14, 16, 32, 36, 37] 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 the full version).

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 the full version (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 [4, 17]. 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 [28]. For a bound of p as above, this severely harms efficiency. Following the strategy of [21], 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. Due to the page limitations, and although the combinatorial analysis of our construction (the bound p) is a core technical contribution of our work, we had to defer it to the

full version, to cover the description of the zero-knowledge proof of the signature scheme in the main body.

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). Due to space limitations, the details on our analysis of the RSD problem are deferred to the full version.

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 [5, 21, 29] (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 [21]. 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 [29] 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 $\mathbf{s_n}$.

Tree-based generation of the seeds. To further reduce the overhead of communicating the seeds, we apply the standard tree-based technique of [29] and generate the seeds as the leaves of a complete binary tree. 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 5.

Using deterministic commitments. As in [29] 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*

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

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 $1 \le i \le n-1$ 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:
     - $\bar{\mathbf{w}}_\mathbf{i}^\mathbf{j} = \mathsf{HW}(\bar{\mathbf{x}}_\mathbf{i}^{\,j}) \bmod T/2$ for all the blocks $1 \le j \le w$;
     - $\mathsf{msg}_i = (\mathbf{z_i}, (\bar{\mathbf{w}}_\mathbf{i}^\mathbf{j})_{1 \le j \le w})$;
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 $(\mathsf{state}_i, \sigma_i)_{i \ne d}$ 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 $j \ne d$;
2. Recompute msg_i for all $i \ne d$ using state_i and z_1;
3. Recompute

$$\bar{\mathsf{msg}}_d = \left(z_1 - \sum_{i \ne d} \mathbf{z_i}, \left(1 - \sum_{i \ne d} \bar{\mathbf{w}}_\mathbf{i}^\mathbf{j} \right)_{1 \le j \le w} \right);$$

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

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 the full version, 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 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:

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

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 [28] 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

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

5.1 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 7. *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 in the full version,* $X = \max_{\alpha \in Q_1}\{X_\alpha\}$ *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*.*

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 seed_e and salt as input of a pseudorandom generator to produce seed_i^e for each $i \in [n]$;
3. Compute 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:
   - $[\![x^e]\!]_2 = (\mathbf{x_1^e}, \ldots, \mathbf{x}_n^e)$;
   - $s = [\![r_1^e]\!]_2 = (\mathbf{s_1^e}, \ldots, \mathbf{s_n^e})$;
   - $t = [\![r^e]\!]_q = (\mathbf{t_1^e}, \ldots, \mathbf{t_n^e})$;
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 $\pi_{\{e \in \tau\}}^e \in S_{K-k}$ via a pseudorandom generator using h_1.

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

1. Each party P_i computes $\mathbf{z_i^e} = \mathbf{x_i^e} \oplus \pi(\mathbf{s_i^e})$;
2. The parties get $[\![z_1^e]\!]_2 = (\mathbf{z_1^e}, \cdots, \mathbf{z_n^e})$ and set $[\![z_2^e]\!]_2 = H'[\![z_1^e]\!]_2 \oplus y$ and so $z^e = (z_1^e | z_2^e)$;
3. Obtain $[\![\bar{x^e}]\!]_q = (\bar{\mathbf{x_1^e}}, \ldots, \bar{\mathbf{x_n^e}})$ where $\bar{\mathbf{x_i^e}} = z^e + (1 - 2z^e) * \pi(\mathbf{t_i^e})$;
4. For each $j \in [n]$ compute:
   - $\bar{\mathbf{w}}_\mathbf{i}^{\mathbf{j,e}} = \langle 1, \bar{\mathbf{x}}_\mathbf{i}^{j,e} \rangle$ for all the blocks $1 \leq j \leq w$;
   - $\mathsf{msg}_i^e = \left(\mathbf{z_i^e}, \left\{ \bar{\mathbf{w}}_\mathbf{i}^{\mathbf{j,e}} \right\}_{1 \leq j \leq w} \right)$.

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 $d_{\{e \in \tau\}}^e \in [n]$ via a pseudorandom generator using h_2.

Phase 5 Output the signature

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

Fig. 5. Signing algorithm of the signature scheme

The proof of Theorem 7 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 [21], and we omit it here.

5.2 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 5. Let f be the number of faulty blocks (of Hamming weight 3 or 5) allowed in the witness ex-

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

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 $\bar{\mathsf{com}}^e_i = H_0(\mathsf{salt}, i, \mathsf{state}^e_i)$;
   - 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
$$\bar{\mathsf{msg}}^e_{d^e} = \left(z^e_1 - \sum_{i\neq d} \mathbf{z}^{\mathbf{e}}_{\mathbf{i}}, \left\{ 1 - \sum_{i\neq d} \bar{\mathbf{w}}^{\mathbf{j},\mathbf{e}}_{\mathbf{i}} \right\}_{1\leq j\leq w} \right);$$
5. Check if $h_1 = H_1(m, \mathsf{salt}, \mathsf{com}^1_1, \cdots, \mathsf{com}^1_n, \cdots, \mathsf{com}^1_\tau, \cdots, \mathsf{com}^\tau_n)$;
6. Check if $h_2 = H_2(m, \mathsf{salt}, h_1, \mathsf{msg}^1_1, \cdots, \mathsf{msg}^1_n, \cdots, \mathsf{msg}^\tau_1, \cdots, \mathsf{msg}^\tau_n)$;
7. Output ACCEPT if both condition are satisfied.

Fig. 6. Verification algorithm of the signature scheme

tracted 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 block-size 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 $\lambda = 128$.

Choosing *k*. As explained in the full version, 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 the full version, 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 (see full version) 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 a lemma in the full version 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 the full version takes a few hours of computation, we first set p using the value predicted by a conjecture, which is in the full version (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 [28] 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 [5]: 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 [1].
- 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 [31]. For SHAKE, the runtime strongly depends on a machine. However, according to the ECRYPT benchmarkings[5], 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 [21], to facilitate comparison with their work (which is the most relevant to ours).

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

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 [5], 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.

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 $w = 255$.

Acknowledgement

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No 945332.

The first and second author acknowledge the support of the French Agence Nationale de la Recherche (ANR), under grant ANR-20-CE39-0001 (project SCENE). This work was also supported by the France 2030 ANR Project ANR-22-PECY-003 SecureCompute.

The third Author of this work has been supported by the European Union's H2020 Programme under grant agreement number ERC-669891.

References

1. Advanced Encryption Standard (AES). National Institute of Standards and Technology (NIST), FIPS PUB 197, U.S. Department of Commerce (Nov 2001)
2. Aragon, N., Blazy, O., Gaborit, P., Hauteville, A., Zémor, G.: Durandal: A rank metric based signature scheme. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 728–758. Springer, Heidelberg (May 2019)
3. Augot, D., Finiasz, M., Sendrier, N.: A fast provably secure cryptographic hash function. Cryptology ePrint Archive, Report 2003/230 (2003), `https://eprint.iacr.org/2003/230`
4. Baum, C., Damgård, I., Larsen, K., Nielsen, M.: How to prove knowledge of small secrets (2016), `https://eprint.iacr.org/2016/538`
5. Baum, C., de Saint Guilhem, C., Kales, D., Orsini, E., Scholl, P., Zaverucha, G.: Banquet: Short and fast signatures from AES. pp. 266–297. LNCS, Springer, Heidelberg (2021)
6. Bellare, M., Goldreich, O.: On defining proofs of knowledge. In: Brickell, E.F. (ed.) CRYPTO'92. LNCS, vol. 740, pp. 390–420. Springer, Heidelberg (Aug 1993)
7. Bernstein, D.J., Lange, T., Peters, C., Schwabe, P.: Really fast syndrome-based hashing. In: Nitaj, A., Pointcheval, D. (eds.) AFRICACRYPT 11. LNCS, vol. 6737, pp. 134–152. Springer, Heidelberg (Jul 2011)
8. Beullens, W.: Sigma protocols for MQ, PKP and SIS, and Fishy signature schemes. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part III. LNCS, vol. 12107, pp. 183–211. Springer, Heidelberg (May 2020)
9. Bidoux, L., Gaborit, P., Kulkarni, M., Mateu, V.: Code-based signatures from new proofs of knowledge for the syndrome decoding problem. arXiv preprint arXiv:2201.05403 (2022)
10. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y.: Compressing vector OLE. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 896–912. ACM Press (Oct 2018)

11. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Resch, N., Scholl, P.: Correlated pseudorandomness from expand-accumulate codes. In: CRYPTO (2022), to appear
12. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Rindal, P., Scholl, P.: Efficient two-round OT extension and silent non-interactive secure computation. In: Cavallaro, L., Kinder, J., Wang, X., Katz, J. (eds.) ACM CCS 2019. pp. 291–308. ACM Press (Nov 2019)
13. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators: Silent OT extension and more. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part III. LNCS, vol. 11694, pp. 489–518. Springer, Heidelberg (Aug 2019)
14. Boyle, E., Couteau, G., Gilboa, N., Ishai, Y., Kohl, L., Scholl, P.: Efficient pseudorandom correlation generators from ring-LPN. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 387–416. Springer, Heidelberg (Aug 2020)
15. Camenisch, J., Kiayias, A., Yung, M.: On the portability of generalized schnorr proofs (2009), https://eprint.iacr.org/2009/050, https://eprint.iacr.org/2009/050
16. Couteau, G., Rindal, P., Raghuraman, S.: Silver: Silent VOLE and oblivious transfer from hardness of decoding structured LDPC codes. pp. 502–534. LNCS, Springer, Heidelberg (2021)
17. Cramer, R., Damgard, I., Xing, C., Yuan, C.: Amortized complexity of zero-knowledge proofs revisited: Achieving linear soundness slack (2016), https://eprint.iacr.org/2016/681, https://eprint.iacr.org/2016/681
18. Debris-Alazard, T., Sendrier, N., Tillich, J.P.: Wave: A new family of trapdoor one-way preimage sampleable functions based on codes. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 21–51. Springer, Heidelberg (Dec 2019)
19. Escudero, D., Ghosh, S., Keller, M., Rachuri, R., Scholl, P.: Improved primitives for MPC over mixed arithmetic-binary circuits. In: Micciancio, D., Ristenpart, T. (eds.) CRYPTO 2020, Part II. LNCS, vol. 12171, pp. 823–852. Springer, Heidelberg (Aug 2020)
20. Feneuil, T., Joux, A., Rivain, M.: 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
21. Feneuil, T., Joux, A., Rivain, M.: Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. In: CRYPTO 2022 (2022)
22. Feneuil, T., Joux, A., Rivain, M.: Syndrome decoding in the head: Shorter signatures from zero-knowledge proofs. Cryptology ePrint Archive (2022)
23. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Heidelberg (Aug 1987)
24. Finiasz, M., Gaborit, P., Sendrier, N.: Improved fast syndrome based cryptographic hash functions. In: Proceedings of ECRYPT Hash Workshop. vol. 2007, p. 155. Citeseer (2007)
25. Gueron, S., Persichetti, E., Santini, P.: 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
26. Hazay, C., Orsini, E., Scholl, P., Soria-Vazquez, E.: TinyKeys: A new approach to efficient multi-party computation. In: Shacham, H., Boldyreva, A. (eds.)

CRYPTO 2018, Part III. LNCS, vol. 10993, pp. 3–33. Springer, Heidelberg (Aug 2018)
27. Ishai, Y., Kushilevitz, E., Ostrovsky, R., Sahai, A.: Zero-knowledge from secure multiparty computation. In: Johnson, D.S., Feige, U. (eds.) 39th ACM STOC. pp. 21–30. ACM Press (Jun 2007)
28. Kales, D., Zaverucha, G.: An attack on some signature schemes constructed from five-pass identification schemes. In: Krenn, S., Shulman, H., Vaudenay, S. (eds.) CANS 20. LNCS, vol. 12579, pp. 3–22. Springer, Heidelberg (Dec 2020)
29. Katz, J., Kolesnikov, V., Wang, X.: Improved non-interactive zero knowledge with applications to post-quantum signatures. In: Lie, D., Mannan, M., Backes, M., Wang, X. (eds.) ACM CCS 2018. pp. 525–537. ACM Press (Oct 2018)
30. Meziani, M., Dagdelen, Ö., Cayrel, P.L., Yousfi Alaoui, S.M.E.: S-fsb: An improved variant of the fsb hash family. In: International Conference on Information Security and Assurance. pp. 132–145. Springer (2011)
31. Münch, J.P., Schneider, T., Yalame, H.: VASA: Vector AES instructions for security applications. Cryptology ePrint Archive, Report 2021/1493 (2021), `https://eprint.iacr.org/2021/1493`
32. Rindal, P., Schoppmann, P.: VOLE-PSI: Fast OPRF and circuit-PSI from vector-OLE. pp. 901–930. LNCS, Springer, Heidelberg (2021)
33. Rotaru, D., Wood, T.: MArBled circuits: Mixing arithmetic and Boolean circuits with active security. Cryptology ePrint Archive, Report 2019/207 (2019), `https://eprint.iacr.org/2019/207`
34. Shor, P.W.: Algorithms for quantum computation: Discrete logarithms and factoring. In: 35th FOCS. pp. 124–134. IEEE Computer Society Press (Nov 1994)
35. Stern, J.: Designing identification schemes with keys of short size. In: Desmedt, Y. (ed.) CRYPTO'94. LNCS, vol. 839, pp. 164–173. Springer, Heidelberg (Aug 1994)
36. Weng, C., Yang, K., Katz, J., Wang, X.: Wolverine: Fast, scalable, and communication-efficient zero-knowledge proofs for boolean and arithmetic circuits. pp. 1074–1091. IEEE Computer Society Press (2021)
37. Yang, K., Weng, C., Lan, X., Zhang, J., Wang, X.: Ferret: Fast extension for correlated OT with small communication. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 20. pp. 1607–1626. ACM Press (Nov 2020)