



MPCitH Signature from Restricted Syndrome Decoding

Michele Battagliola¹, Ethan Y. Chen², Hugo Sauerbier Couvée³, and Violetta Weger³

¹ Polytechnic University of Marche, Ancona, Italy
`battagliola.michele@proton.me`

² Duke University, Durham, USA
`ethan.chen@duke.edu`

³ Technical University of Munich, Munich, Germany
`{hugo.sauerbier-couvee, violetta.weger}@tum.de`

Abstract. CROSS is a code-based signature based on the Restricted Syndrome Decoding Problem (R-SDP) that is currently among the fourteen candidates in the NIST standardization process. While CROSS enjoys a very competitive verification time, its primary drawback is its significantly large signature size. In this work, we introduce a new Multi-Party Computation in the Head (MPCitH) protocol for the R-SDP with the primary goal of reducing the signature sizes of CROSS. To do so, we design a publicly verifiable secret sharing scheme tailored to restricted vectors and a new multiplicative-to-additive conversion for it. These new cryptographic gadgets may be of independent interest as they can serve as building blocks for future research in multi-party computation, such as a threshold version of CROSS.

Keywords: Digital signature, Restricted decoding, MPCitH, Secret Sharing

1 Introduction

The development of quantum computing poses an imminent threat to the public-key cryptographic algorithms that secure modern digital communication. In response to this challenge, NIST initiated its first call for the Post-Quantum Cryptography (PQC) standardization process in December 2016. Following the conclusion of this initial phase, a subsequent call was issued by NIST in September 2022, to diversify the landscape of post-quantum signatures. In the first round, 40 algorithms were accepted as complete and proper; in October 2024, NIST announced 14 algorithms that were promoted to the second round [5].

CROSS. One of these schemes is CROSS, which is obtained by converting, via the Fiat-Shamir transform, an interactive Zero Knowledge (ZK) proof for the Restricted Syndrome Decoding Problem (R-SDP) [7].

More specifically, CROSS is a five-round protocol where the signer wants to prove the knowledge of a secret vector \mathbf{e} whose entries are constrained in a set \mathbb{G} (hence the word *restricted*) such that \mathbf{e} is the error vector corresponding to a public syndrome $(\mathbf{s}, \mathbf{H}^\top)$. The particular structure of CROSS makes it very competitive in terms of speed, where it is among the fastest zero-knowledge based algorithms; however, this comes at the price of a very large signature size due to the large number of repetitions needed.

Multi-Party Computation in the Head. Proposed in 2007 by Ishai, Kushilevitz, Ostrovsky and Sahai [26], the Multi-Party Computation in the Head (MPCitH) technique has seen widespread success in the design of post-quantum digital signatures, with five out of fourteen candidates currently in the NIST competition adopting it [1, 2, 3, 6, 14], with a sixth one, FAEST, being based on the VOLE in the Head (VOLEitH) framework that is closely related to it [13]. In general, MPCitH signatures offer a good balance between verification time and signature size, thanks to various optimizations such as the hypercube technique and the Threshold Computation in the Head (TCitH) framework that allow an increase in soundness and thus a reduction in the required number of repetitions.

1.1 Our Contributions

In this work, we present a new MPCitH protocol for the R-SDP, with the primary goal of reducing the signature size of CROSS, targeting the instances where the short version of CROSS would be employed. Our approach comes at the cost of increased computational time, but it offers a valuable trade-off for applications where bandwidth or storage is of primary concern. In particular, we manage to reduce the signature size of CROSS by around 55 – 60% with respect to the original protocol, making it only slightly bigger when compared to the other MPCitH signatures and thus offering a valid alternative for the short version of CROSS.

As cornerstones of our construction, we introduce two new cryptographic gadgets that may be of independent interest for future research in multi-party computation:

1. A publicly verifiable secret sharing scheme for restricted vectors, which is a novel and essential building block for our protocol.
2. A new multiplicative-to-additive conversion protocol tailored for operations on shared restricted errors.

Beyond their application in the context of MPCitH, these tools may also be interesting for the purpose of true MPC protocols on shared restricted vectors, such as a threshold version of CROSS having a distributed key generation.

Recently, VOLEitH and TCitH versions of CROSS were proposed in [12] to reduce the signature size. Unlike the signature presented in [12], which utilizes a standard polynomial system of equations model typical of VOLEitH proofs, our approach takes a fundamentally different route by directly designing an MPC protocol for restricted syndrome evaluation.

1.2 Paper Organization

The remainder of this paper is organized as follows. In Section 2 we introduce the preliminary notions needed for the paper, with a focus on multi-round protocols and MPCitH. In Section 3 we describe the multi-party protocol that is at the foundation of our signature, as well as the secret sharing used and the multiplicative-to-additive conversion protocol. In Section 4 we describe the MPCitH based zero-knowledge protocol and prove its security, while in Section 5 we provide the full description of the signature obtained from it. In Section 6 we present some possible optimizations to our protocol, such as the hypercube technique and the Threshold Computation-in-the-Head (TCitH) framework. Finally in Section 7 we present our conclusions, discuss open problems, and suggest ideas for future work.

2 Preliminaries

In this section we introduce the notation and the various mathematical notions used through this paper. We also provide a brief introduction to the MPCitH framework.

2.1 Notation

Vectors are written as bold symbols, and the j^{th} component of the vector \mathbf{v} is denoted by v_j . Arithmetic operations on vectors, unless otherwise stated, are performed component-wise, and all logarithms are taken base-2. Component-wise multiplication of vectors is denoted by \star . Finite fields are generally denoted by \mathbb{F} , with \mathbb{F}_q denoting a finite field of size q . The symmetric group over a set of n elements, or the set of permutations over n elements, is denoted by S_n . We use the notation $a \leftarrow_{\$} A$, to mean that a was sampled uniformly at random from the set A .

2.2 Coding Theory

A *linear code* over \mathbb{F}_q with length n is a k -dimensional subspace of \mathbb{F}_q^n . The code can be represented by a *parity-check matrix* $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$ such that $\mathbf{c}\mathbf{H}^\top = 0$ for every codeword \mathbf{c} . The *syndrome* of a vector \mathbf{v} is $\mathbf{s} = \mathbf{v}\mathbf{H}^\top$. The *Hamming weight* of a vector \mathbf{v} , denoted $wt_H(\mathbf{v})$, is the number of non-zero entries in \mathbf{v} . Many cryptographic protocols in code-based cryptography are based on the Syndrome Decoding Problem (SDP) from coding theory, which is known to be NP-complete [10, 15].

Problem 1 (Decisional Syndrome Decoding Problem). Given a parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$, $t \in \mathbb{N}$, and a syndrome $\mathbf{s} \in \mathbb{F}_q^{n-k}$, decide whether there exists an error vector $\mathbf{e} \in \mathbb{F}_q^n$ such that $\mathbf{e}\mathbf{H}^\top = \mathbf{s}$ and $wt_H(\mathbf{e}) \leq t$.

The Restricted Syndrome Decoding Problem (R-SDP) [8, 9] is a variant of SDP that is also NP-complete. Instead of bounding the weight of the errors, we ask the entries of the error vector \mathbf{e} to be restricted to a cyclic subgroup \mathbb{G} of \mathbb{F}_q^* generated by g . Relying on R-SDP rather than SDP allows for more compact representations of the error vector \mathbf{e} as well as a simpler MPCitH protocol for proving knowledge of \mathbf{e} .

Problem 2 (Restricted Syndrome Decoding Problem). Given a parity-check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$, a syndrome $\mathbf{s} \in \mathbb{F}_q^{n-k}$, and a cyclic group $\mathbb{G} = \{g^i \mid i \in \{0, 1, \dots, z-1\}\}$ for some $g \in \mathbb{F}_q^*$ of order z , decide whether there exists an error vector $\mathbf{e} = (g^{x_1}, \dots, g^{x_n}) \in \mathbb{G}^n$ such that $\mathbf{e}\mathbf{H}^\top = \mathbf{s}$.

2.3 Cryptographic Tools

Pseudo Random Generator. Let $G : \{0, 1\}^* \rightarrow \{0, 1\}^*$ be a polynomial-time computable function, and let $\ell : \mathbb{N} \rightarrow \mathbb{N}$ be a polynomial-time computable function such that $\ell(k) > k$ for all k . G is a (t, ε) -secure *Pseudo Random Generator* (PRG) if it satisfies the following two conditions:

- For any k and any $x \in \{0, 1\}^k$, $G(x) \in \{0, 1\}^{\ell(k)}$.
- For any algorithm \mathcal{A} running in time t ,

$$|\Pr[\mathcal{A}(G(x)) = 1 \mid x \leftarrow \{0, 1\}^k] - \Pr[\mathcal{A}(x') = 1 \mid x' \leftarrow \{0, 1\}^{\ell(k)}]| < \varepsilon(k).$$

A *Tree PRG* [24] is a PRG that expands a master seed into N subseeds using a tree structure and allows subseeds to be revealed by revealing a small subset of labels in the tree. The root of the tree is labelled with the master seed. Then, for each node in the tree, its children are labelled with the output of a PRG applied to the label of the node. The subseeds are the labels of the N leaves of the tree. Given a small subset S of the subseeds, we can reveal all the subseeds except S by revealing the labels of the siblings on the paths from the root to the subseeds in S . The number of labels revealed is bounded by $|S| \log(N)$.

Commitment Scheme. A *commitment scheme* consists of a pair of algorithms (Com, Ver) .

- a commit algorithm Com which takes a message \mathbf{m} and randomness ρ as input, and outputs a commitment c .
- a verify algorithm Ver which takes a message \mathbf{m} , a commitment c , and randomness ρ as input, and outputs a bit $b = 1$ if c is correctly generated from \mathbf{m} and ρ , and outputs $b = 0$ otherwise.

A commitment scheme should have the following properties:

- **Correctness:** For any message \mathbf{m} and randomness ρ ,

$$\Pr[\text{Ver}(\mathbf{m}, \text{Com}(\mathbf{m}, \rho), \rho) = 1] = 1.$$

- **Hiding:** for any Probabilistic Polynomial Time (PPT) algorithm \mathcal{A} and any two messages m_1, m_2 the following is negligible:

$$\Pr[\mathcal{A}(c) = 1 \mid c \leftarrow \$ \text{Com}(m_1)] - \Pr[\mathcal{A}(c') = 1 \mid c' \leftarrow \$ \text{Com}(m_2)].$$

- **Binding:** For any PPT algorithm \mathcal{A} , the following probability is negligible:

$$\Pr[m_1 \neq m_2, \text{Ver}(m_1, c, \rho_1) = \text{Ver}(m_2, c, \rho_2) = 1 \mid (\{m_i, \rho_i\}_{i=1,2}, c) \leftarrow \$ \mathcal{A}(1^\lambda)].$$

For our purposes, we consider a commitment scheme that outputs commitments of size 2λ .

2.4 Five-round Protocols.

We provide here the relevant definition for five-round protocols.

Definition 1 (Binary relation). A binary relation is a finite set $R \subseteq X \times Y$, where $X, Y \subseteq \{0, 1\}^*$. Given $(x, y) \in R$, we say that y is a witness for the statement x . The set $L_R = \{x \in X \mid \exists y \in Y \text{ s.t. } (x, y) \in R\}$ is called the set of true statements for R , or its language.

Throughout this work we are interested in *hard relations*, i.e., relations for which no polynomial time algorithm is able to compute y from x .

Definition 2 (Interactive Proof). An interactive proof $(\mathcal{P}, \mathcal{V})$ for a binary relation $R \subseteq X \times Y$ is an interactive protocol between two PPT machines \mathcal{P} and \mathcal{V} . The prover \mathcal{P} takes as input a pair $(x, y) \in R$ while the verifier \mathcal{V} takes as input x . At the end of the protocol, \mathcal{V} either accepts (outputs 1) or rejects (outputs 0). We denote the output of the protocol with $(\mathcal{P}(y), \mathcal{V})(x)$. Furthermore, we say that a transcript, i.e., the set of all messages exchanged in a protocol execution, is accepting (rejecting) if \mathcal{V} accepts (rejects, respectively).

Definition 3 (Public-Coin). An interactive proof $(\mathcal{P}, \mathcal{V})$ is public-coin if all \mathcal{V} 's random choices are made public.

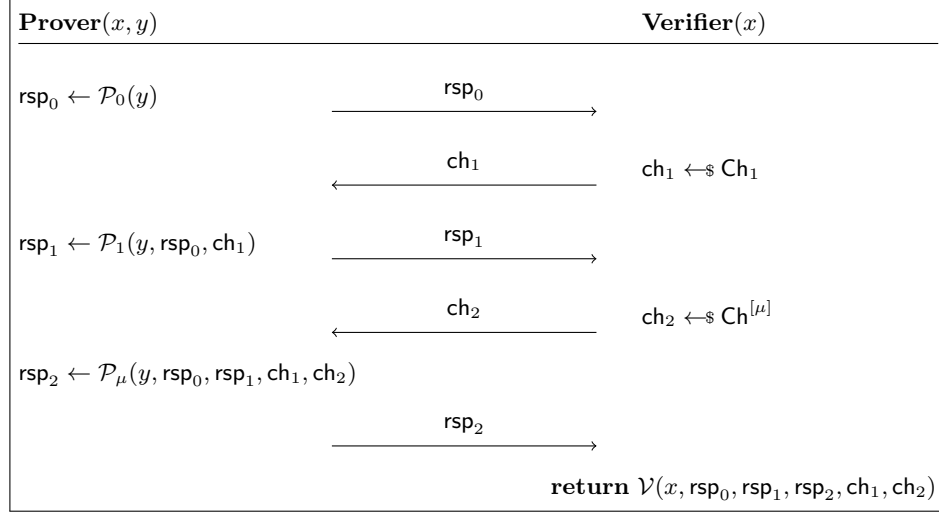
If an interactive proof is public-coin, the verifier needs to send to the prover only their random choices. For this reason, we call *challenges* the messages ch sent by the verifier and *challenge set* the set Ch from which the verifier's messages are sampled. We call *responses* the messages rsp sent by the prover. Since we are interested in 5-round protocols, the prover sends three messages $(\text{rsp}_i, i \in \{0, 1, 2\})$ and the verifier sends two intermediate challenges $(\text{ch}_i, i \in \{1, 2\})$. We depict such a protocol in Figure 1.

Commonly, an interactive proof is required to satisfy completeness and soundness, as per definitions below.

Definition 4 (Completeness). An interactive proof $(\mathcal{P}, \mathcal{V})$ for a binary relation $R \subseteq X \times Y$ is complete if, for every $(x, y) \in R$, we have:

$$\Pr[(\mathcal{P}(y), \mathcal{V})(x) = 0] \leq \gamma(x),$$

where the value $\gamma(x)$, called completeness error, is negligible (in $|x|$). If $\gamma(x) = 0$ for all $x \in L_R$, the protocol is said to be perfectly complete.

**Fig. 1.** Public-Coin $(2\mu + 1)$ -Round Interactive Proof

Definition 5 (Soundness). *An interactive proof $(\mathcal{P}, \mathcal{V})$ for a binary relation $R \subseteq X \times Y$ is sound if, for every $x \notin L_R$ and a (potentially-dishonest) prover \mathcal{P}^* , we have:*

$$\Pr[(\mathcal{P}^*, \mathcal{V})(x) = 1] \leq \sigma(x),$$

where the value $\sigma(x)$, called soundness error, is negligible (in $|x|$).

We note that an interactive proof which satisfies both the previous properties allows a prover \mathcal{P} to convince the verifier \mathcal{V} that a statement x is true. To prove \mathcal{P} 's knowledge of a witness y such that $(x, y) \in R$, the following stronger feature is required.

Definition 6 (Knowledge Soundness). *An interactive proof $(\mathcal{P}, \mathcal{V})$ for a binary relation $R \subseteq X \times Y$ is knowledge sound, with knowledge error κ , if there exists an algorithm \mathcal{E} that, given as input any $x \in X$ and rewindable oracle access to a (potentially-dishonest) prover \mathcal{P}^* , runs in an expected polynomial time (in $|x|$) and outputs a witness $y \in Y$ for x with probability:*

$$\Pr[(x, \mathcal{E}^{\mathcal{P}^*}(x)) \in R] \geq \frac{\varepsilon(x, \mathcal{P}^*) - \kappa(x)}{\text{poly}(|x|)},$$

where $\varepsilon(x, \mathcal{P}^*) = \Pr[(\mathcal{P}^*, \mathcal{V})(x) = 1]$. The algorithm \mathcal{E} is called knowledge extractor.

Definition 7 (Proof of Knowledge). *An interactive proof $(\mathcal{P}, \mathcal{V})$ for a binary relation $R \subseteq X \times Y$ which satisfies both completeness with completeness error ρ and knowledge soundness with knowledge error κ is a proof of knowledge if there exists a positive-definite polynomial p over the integers such that $1 - \rho(x) \geq \kappa(x) + \frac{1}{p(|x|)}$ for all $x \in X$.*

Definition 8 (Honest-Verifier Zero-Knowledge (HVZK)). *An interactive proof $(\mathcal{P}, \mathcal{V})$ for a binary relation R is computationally (resp., statistically/perfectly) Honest-Verifier Zero-Knowledge (HVZK) if there exists a PPT algorithm S (called simulator), such that for any $x \in L_R$, S produces a transcript which is computationally (resp., statistically/perfectly) indistinguishable from the distribution of the transcripts obtained through the interaction of \mathcal{P} and \mathcal{V} .*

To prove the HVZK property we prove a stronger version, namely *piecewise simulatability*, introduced in [28].

Definition 9 (Piecewise Simulatability). *A 5-round argument of knowledge is piecewise simulatable if there exist probabilistic-polynomial time algorithms $A = (A_1, A_2)$ and $B = (B_1, B_2)$, defined as follows:*

Simulator A:

- 1: $T_1 = (\text{rsp}_0, \text{ch}_1, \text{rsp}_1) \leftarrow_{\$} A_1(x)$
- 2: $\text{rsp}_2 \leftarrow_{\$} A_2(x, T_1, \text{ch}_2)$
- 3: $T \leftarrow (\text{rsp}_0, \text{ch}_1, \text{rsp}_1, \text{ch}_2, \text{rsp}_2)$

Simulator B:

- 1: $T_1 = \text{rsp}_0 \leftarrow_{\$} B_1(x)$
- 2: $(\text{rsp}_1, \text{ch}_2, \text{rsp}_2) \leftarrow_{\$} B_2(x, T_1, \text{ch}_1)$
- 3: $T' \leftarrow (\text{rsp}_0, \text{ch}_1, \text{rsp}_1, \text{ch}_2, \text{rsp}_2)$

where T and T' are distributed as the output of an honest execution of the protocol when ch_1 (resp., ch_2) is chosen uniformly at random from the challenge space.

Informally speaking, a protocol is piecewise simulatable if it is possible to simulate an honest execution by only choosing one challenge and receiving the remaining part from an external source: indeed, simulator A produces the first half of the transcript, receives the second challenge ch_2 and then completes the transcript, while B generates the first message, receives the first challenge ch_1 and then simulates the remaining part. This notion is particularly interesting since it allows us to upper bound the soundness of protocol (that is the probability of guessing at least one challenge) and since it can be used to attack protocols based on parallel repetition, as shown in [11, 28].

2.5 Multi-Party Computation

A secure Multi-Party Computation (MPC) protocol is an interactive protocol between N parties which allows the parties to jointly perform computation of an agreed upon function on their private inputs without revealing the private inputs. See [19, 30] for formal MPC security definitions.

Secret Sharing. Let $x \in \mathbb{F}_q$, and denote by $\llbracket x \rrbracket = (\llbracket x \rrbracket_1, \dots, \llbracket x \rrbracket_N) \in \mathbb{F}_q^N$ a *secret-sharing* of x among N parties, where party P_i receives share $\llbracket x \rrbracket_i$. A vector $\mathbf{v} \in \mathbb{F}_q^n$ is secret-shared component-wise, i.e., $\llbracket \mathbf{v} \rrbracket = (\llbracket v_1 \rrbracket, \dots, \llbracket v_n \rrbracket)$. An *additive* secret sharing for a secret x is obtained as follows: first the dealer randomly generates the first $N-1$ shares, then sets the N^{th} share as $\llbracket x \rrbracket_N = x - \sum_{i=1}^{N-1} \llbracket x \rrbracket_i$. To obtain a *multiplicative* secret sharing, the dealer checks that $\llbracket x \rrbracket_i \neq 0$ for all i and then sets the last share as $\llbracket x \rrbracket_N = x / \prod_{i=1}^{N-1} \llbracket x \rrbracket_i$.

Observe that additive secret sharings are linearly homomorphic, meaning that $c \cdot \llbracket a \rrbracket + \llbracket b \rrbracket = \llbracket c \cdot a + b \rrbracket$, and multiplicative secret sharings are multiplicatively homomorphic, meaning that $\llbracket a \rrbracket \cdot \llbracket b \rrbracket = \llbracket a \cdot b \rrbracket$.

To avoid confusion when both multiplicative and additive secret sharing are used in the same protocol, we use the symbol $\llbracket x \rrbracket$ to denote additive secret sharing, while we use $\langle\langle x \rangle\rangle$ for multiplicative secret sharing. In the contexts where the exact style of secret sharing is not relevant we use simply the symbol $\llbracket x \rrbracket$.

MPC-in-the-Head. The *MPC-in-the-Head* (MPCitH) paradigm [26] is a general way of building HVZK arguments of knowledge from MPC protocols. Let R be a hard relation and $x \in \{0, 1\}^*$ a public statement. Let f be a function that on input $w \in \{0, 1\}^*$ checks whether $(x, w) \in R$ or not. Let us consider a semi-honest secure N -party protocol that computes f . We can build an HVZK argument of knowledge in the following way:

- The prover generates an N -party secret sharing of $\llbracket w \rrbracket = (\llbracket w \rrbracket_1, \dots, \llbracket w \rrbracket_N)$.
- Then, the prover simulates “in its head” the N -party MPC protocol, where party P_i has input $\llbracket w \rrbracket_i$.
- For each party, the prover commits to the party’s view and sends the commitment to the verifier, alongside all broadcasts made in the MPC protocol and the output shares.
- The verifier then sends a challenge $\text{ch} \in \{1, \dots, N\}$.
- The prover reveals the $\llbracket w \rrbracket_i$ for all $i \neq \text{ch}$, as well as the randomness used for all the commitments except the one for party P_{ch} .
- The verifier checks that the $N - 1$ inputs shares are consistent with the commitments and broadcasts, while also checking that the output of the protocol is 1. If the checks are satisfied, then the verifier accepts; otherwise, it rejects.

By construction, a prover with knowledge of w can always convince the verifier to accept. Moreover, since the starting MPC protocol is secure against semi-honest adversary, the transcript does not reveal anything about the witness w . Finally, it is easy to see that the protocol is sound since from the response of two different ch, ch' it is possible to reconstruct the whole w .

Now, suppose that the MPC protocol has probability p of outputting 1 even if $(x, w) \notin R$, then, a dishonest prover without the knowledge of w can successfully cheat if it corrupts one party’s computation, and that party is challenged by the verifier, or if the MPC protocol incorrectly accepts. For N parties, the first case occurs with probability $1/N$. The second case occurs with probability p . Thus, the zero-knowledge protocol has a soundness error of

$$\frac{1}{N} + p \left(1 - \frac{1}{N}\right).$$

3 Multi-Party Computation Protocols

In this section we design a multi-party protocol for the Restricted Syndrome Decoding Problem (R-SDP). To do so, we first need a share conversion protocol that converts multiplicative shares of a secret into additive shares of the same secret.

3.1 Secret sharing for R-SDP

We first show a multiplicative and an additive secret sharing for R-SDP vector \mathbf{e} such that $\mathbf{e}\mathbf{H}^\top = \mathbf{s}$ for some public $(\mathbf{s}, \mathbf{H}^\top)$. In the following D is the dealer, while P_1, \dots, P_N are the participants of the secret sharing.

Multiplicative Secret Sharing The dealer D picks $N - 1$ random restricted vectors $\mathbf{e}_1, \dots, \mathbf{e}_{N-1}$ then sets $\mathbf{e}_N = \mathbf{e} / \prod_{i=1}^{N-1} \mathbf{e}_i$. Then, for all $i = 1, \dots, N$, D sends \mathbf{e}_i to P_i and publishes $\mathbf{s}_i = \mathbf{e}_i \mathbf{H}^\top$. Notice that the published $\mathbf{e}_i \mathbf{H}^\top$ are instances of R-SDP, so security follows trivially.

To verify the correctness of their share, party P_i checks that \mathbf{e}_i is restricted and then computes $\mathbf{e}_i \mathbf{H}^\top$ and checks the published data. However, notice that it is impossible to check that the shared secret is indeed a witness for $(\mathbf{s}, \mathbf{H}^\top)$ without reconstruction, since from the public data \mathbf{s}_i it is impossible to compute \mathbf{s} (except when $N = 2$).

Additive Secret Sharing The dealer D picks $N - 1$ random vector $\mathbf{e}_1, \dots, \mathbf{e}_{N-1}$ then sets $\mathbf{e}_N = \mathbf{e} - \sum_{i=1}^{N-1} \mathbf{e}_i$. Then D sends \mathbf{e}_i to P_i and publish $\mathbf{e}_i \mathbf{H}^\top$.

To verify the correctness of their share, party P_i simply computes $\mathbf{e}_i \mathbf{H}^\top$ and checks the published data. It can also be checked that $\sum_{i=1}^N \mathbf{e}_i \mathbf{H}^\top = \mathbf{s}$. However, notice that P_i has no guarantee that the reconstructed vector is indeed restricted, since \mathbf{e}_i is not required to be so. Moreover, notice that the published $\mathbf{e}_i \mathbf{H}^\top$ are not instances of R-SDP, so in principle there could be some leakage. We now show that it is not the case and that an adversary controlling $N - 1$ parties is not able to retrieve the secret value \mathbf{e} .

Proposition 1. *Let \mathcal{A} be an adversary for the additive secret sharing scheme controlling $N - 1$ parties. If \mathcal{A} is able to compute \mathbf{e} from the public data and the data of the controlled parties, then there exists an adversary \mathcal{A}' able to solve R-SDP.*

Proof. Let $(\mathbf{s}, \mathbf{H}^\top)$ be an instance of R-SDP. On input $(\mathbf{s}, \mathbf{H}^\top)$, \mathcal{A}' works as follows: it first picks $N - 1$ random vectors $\mathbf{e}_1, \dots, \mathbf{e}_{N-1}$, computes $\mathbf{s}' = \mathbf{s} - \sum_{i=1}^{N-1} \mathbf{e}_i \mathbf{H}^\top$, and then publishes all the $\mathbf{s}_i = \mathbf{e}_i \mathbf{H}^\top$ for $i = 1, \dots, N - 1$ and \mathbf{s}' . The adversary \mathcal{A} then chooses the $N - 1$ parties it aims to corrupt. If P_N is not among them, then \mathcal{A}' can answer with the corresponding $\mathbf{e}_1, \dots, \mathbf{e}_{N-1}$. In this case the simulation is perfect, since all the \mathbf{e}_i and \mathbf{s}_i are honestly generated. Otherwise \mathcal{A}' computes a vector \mathbf{e}'_N such that $\mathbf{e}'_N \mathbf{H}^\top = \mathbf{s}'$. This simulation is indistinguishable from the real one, since \mathcal{A} has only access to $N - 1$ parties

and all the verification checks are satisfied. Indeed, the only difference is that $\sum_{i=1}^{N-1} \mathbf{e}_i + \mathbf{e}'_N$ is not restricted. However, let \mathbf{e}_j be the error vector not queried by \mathcal{A} and \mathbf{s}_j be the corresponding public syndrome. Notice that the vectors $\{\mathbf{e}_i\}_{i=1, \dots, N-1, i \neq j}, \mathbf{e}'_N, \mathbf{e}'_j = \mathbf{e} - \sum_{i=1, i \neq j}^{N-1} \mathbf{e}_i - \mathbf{e}'_N$ form an additive secret sharing of \mathbf{e} such that $\mathbf{s}_j = \mathbf{e}'_j \mathbf{H}^\top$ by construction. Thus the simulation is indistinguishable from a real execution.

In both cases the simulation is indistinguishable from a real execution, thus eventually \mathcal{A} will output a restricted vector \mathbf{e}' . At this point \mathcal{A}' can use the same vector to solve the initial challenge. \square

The main feature of additive secret sharing is its homomorphic property, which allows verification of the syndrome equation with only the public data. However, as said before, there is no way to check whether the initial secret vector is restricted or not. On the other hand, in multiplicative secret sharing each party receives a restricted vector, so their (reconstructed) product is restricted as well. In the following we show a protocol that converts multiplicative secret shares to additive secret shares, allowing us to have the advantages of both secret sharing schemes in our MPC protocol.

3.2 Share conversion

Presented independently in [31] and [33], the following protocol converts multiplicative shares of a secret x into additive shares of the same secret in the MPC with preprocessing model. In addition to their share of x , each party is provided secret shares of random values r and t , where r is shared additively, t is shared multiplicatively, and $r = t \neq 0$.

1. The parties locally set $\langle\langle \alpha \rangle\rangle_i = \langle\langle x \rangle\rangle_i / \langle\langle t \rangle\rangle_i$.
2. The parties reconstruct α .
3. The parties locally compute $\llbracket x \rrbracket_i = \alpha \cdot \llbracket r \rrbracket_i$.

Correctness. Assuming $r = t$,

$$\sum_{i=1}^N \llbracket x \rrbracket_i = \sum_{i=1}^N \alpha \cdot \llbracket r \rrbracket_i = \alpha \cdot \sum_{i=1}^N \llbracket r \rrbracket_i = (x/t) \cdot r = x,$$

hence the protocol correctly computes an additive secret sharing of x .

Security. The only messages that the parties receive in the protocol are all the shares of $\llbracket \alpha \rrbracket$. A simulator can simply generate $\llbracket \alpha \rrbracket$ uniformly at random, since t is assumed to be uniformly distributed, and thus $\alpha = x/t$ is also uniformly distributed in the real world.

3.3 Protocol for R-SDP from share conversion

Given an instance $(\mathbf{s}, \mathbf{H}^\top)$ of R-SDP with restriction $\mathbb{G} = \{g^i\}$, this protocol based on the share conversion protocol described above checks that the error vector, represented by the exponents \mathbf{x} where $\mathbf{e} = g^{\mathbf{x}}$, is a valid solution. The secret \mathbf{x} is secret-shared additively to the N parties. Notice that if \mathbf{x} is shared additively, then $g^{\llbracket \mathbf{x} \rrbracket}$ is a multiplicative secret sharing of $\mathbf{e} = g^{\mathbf{x}}$. Additionally, random vectors \mathbf{r} and \mathbf{t} , where $\mathbf{r} = \mathbf{t}$, are shared additively and multiplicatively, respectively.

1. The parties locally set $\llbracket \alpha \rrbracket_i = g^{\llbracket \mathbf{x} \rrbracket_i} / \llbracket \mathbf{t} \rrbracket_i$.
2. The parties reconstruct α .
3. The parties locally set $\llbracket \mathbf{e} \rrbracket_i = \alpha \star \llbracket \mathbf{r} \rrbracket_i$.
4. The parties locally set $\llbracket \mathbf{v} \rrbracket_i = \llbracket \mathbf{e} \rrbracket_i \mathbf{H}^\top$.
5. The parties reconstruct \mathbf{v} , and the protocol accepts if $\mathbf{v} = \mathbf{s}$.

Correctness. As said before $g^{\llbracket \mathbf{x} \rrbracket}$ is a multiplicative secret sharing of \mathbf{e} , since

$$\prod_{i=1}^N g^{\llbracket \mathbf{x} \rrbracket_i} = g^{\sum_{i=1}^N \llbracket \mathbf{x} \rrbracket_i} = g^{\mathbf{x}}.$$

Then, the first three steps of the protocol converts the multiplicative shares to additive shares. The correctness of this part follows directly from the correctness of the share conversion protocol. Finally, the protocol checks that $\mathbf{s} = \mathbf{e} \mathbf{H}^\top$, which is straightforward since the equation is linear and \mathbf{e} is shared additively.

Security. The security of this protocol follows immediately from the security of the share conversion protocol.

4 Zero-Knowledge Protocol for R-SDP

Using the MPCitH paradigm [26], we construct a zero-knowledge protocol for proving knowledge of a solution to an instance $(\mathbf{s}, \mathbf{H}^\top)$ of R-SDP. Our five-round protocol follows a similar structure to the zero-knowledge protocol for SDP proposed in [20] and incorporates the MPC protocols described in the previous section.

The share conversion protocol described previously requires preprocessing to produce random vectors \mathbf{r}, \mathbf{t} where $\mathbf{r} = \mathbf{t}$, which can be modelled by a trusted third helper party in the context of a zero-knowledge protocol between a prover and a verifier. However, we cannot allow a trusted helper party when applying the Fiat-Shamir transformation to obtain a signature scheme and must allow the prover to generate \mathbf{r} and \mathbf{t} . The challenge lies in ensuring that the prover does not cheat by simply choosing $\mathbf{r} \neq \mathbf{t}$. The *cut-and-choose* method applied to the MPCitH paradigm, introduced in [29] and formalized by [16], is a standard technique that can be then be used to remove the need for a helper party, but

comes at the cost of significant additional computation and communication. Instead, we apply an idea from the MPCitH-based zero-knowledge protocol for the regular SDP in [17], which also relies on a form of share conversion. The analysis of their protocol's soundness is specific to the regular decoding problem, so the soundness of our protocol requires independent analysis.

The high-level idea is to first allow the prover to generate and commit to $\llbracket \mathbf{r} \rrbracket, \llbracket \mathbf{t} \rrbracket$. Then, the vectors \mathbf{r}, \mathbf{t} are shuffled by a random permutation π (provided as a challenge by the verifier) prior to being used in the MPC protocol. The shuffle does not affect the correctness of the protocol but prevents the prover from cheating with high probability. We now describe the zero-knowledge protocol in more detail.

The prover and the verifier are given \mathbf{H} , \mathbf{s} , and g generating \mathbb{G} as public inputs. The prover also holds its secret \mathbf{x} , the exponents of the restricted error vector $\mathbf{e} = (g^{x_1}, \dots, g^{x_n})$ satisfying the syndrome equation $\mathbf{e}\mathbf{H}^\top = \mathbf{s}$.

The prover begins by randomly sampling a master seed, which it then uses to generate N seeds for each of the N simulated MPC parties using a Tree PRG [24]. The shares of \mathbf{r} , \mathbf{t} , and \mathbf{x} are then generated using the seeds. After all secret shares have been generated, the prover commits to the states of the N parties. Note that for $N - 1$ parties, the state consists of only their seed, since the rest of their state can be completely generated using the seed. The commitments are hashed by a collision-resistant hash function and sent to the verifier.

The verifier then challenges the prover with a random permutation π . The prover shuffles $\llbracket \mathbf{r} \rrbracket, \llbracket \mathbf{t} \rrbracket$ with π and then simulates the MPC protocol with $(\llbracket \mathbf{x} \rrbracket, \pi(\llbracket \mathbf{r} \rrbracket), \pi(\llbracket \mathbf{t} \rrbracket))$ as input. All shares broadcasted in the MPC protocol are hashed and sent to the verifier.

The verifier chooses a random party $i^* \in \{1, \dots, N\}$ as the second challenge to the prover. The prover then sends the views of all parties except i^* to the verifier, along with the challenged party's commitment and their broadcasted share from the MPC protocol. The prover does not have to send its share of the syndrome computed in the MPC protocol, since the verifier can compute it from the other $N - 1$ shares as the syndrome is known publicly. The verifier finally simulates the MPC protocol for the $N - 1$ revealed parties and recomputes the hashes to check their consistency with what was sent by the prover.

Theorem 1. *The protocol of Section 4 is complete, sound and honest-verifier zero-knowledge.*

Proof. The proof is an adaptation of the ones in [20] and [17].

Completeness. If the prover is honest and knows the solution \mathbf{x} , then the checks always pass, following from the correctness of the MPC protocol.

Soundness. Let p be the cheating probability for the first random permutation challenge. Then, the protocol has soundness error

$$\varepsilon = \frac{1}{N} + p \left(1 - \frac{1}{N} \right).$$

For simplicity, assume the commitment scheme is perfectly binding. Suppose there exists an efficient prover $\hat{\mathcal{P}}$ that, given $(\mathbf{s}, \mathbf{H}^\top)$ as input, convinces the honest verifier to accept with probability $\hat{\varepsilon} > \varepsilon$. We show that there then exists an extraction algorithm \mathcal{E} that, given black-box rewindable access to $\hat{\mathcal{P}}$, finds a witness \mathbf{x} such that $g^{\mathbf{x}}\mathbf{H}^\top = \mathbf{s}$.

Let R_h be a random variable for the randomness used by $\hat{\mathcal{P}}$ to generate the first round message h . Let $\text{Succ}_{\hat{\mathcal{P}}}$ denote the event that $\hat{\mathcal{P}}$ successfully convinces the verifier V . By assumption,

$$\Pr[\text{Succ}_{\hat{\mathcal{P}}}] = \hat{\varepsilon} > \varepsilon.$$

Choose $\alpha \in (0, 1)$ such that $(1 - \alpha)\hat{\varepsilon} > \varepsilon$, which is possible since $\hat{\varepsilon} > \varepsilon$. We call a realization ρ_h of R_h good if

$$\Pr[\text{Succ}_{\hat{\mathcal{P}}} \mid R_h = \rho_h] \geq (1 - \alpha)\hat{\varepsilon}.$$

By the Splitting Lemma in [20],

$$\Pr[\rho_h \text{ is good} \mid \text{Succ}_{\hat{\mathcal{P}}}] \geq \alpha.$$

Now suppose T_0 is a transcript of a successful zero-knowledge proof between $\hat{\mathcal{P}}$ and \mathcal{V} , and let ρ_h be the randomness used by $\hat{\mathcal{P}}$ in the first round of this interaction. Let i_0^* be the second challenge in T_0 . If ρ_h is good, then

$$\Pr[\text{Succ}_{\hat{\mathcal{P}}} \mid R_h = \rho_h] \geq (1 - \alpha)\hat{\varepsilon} > \varepsilon > \frac{1}{N}.$$

Thus, there must exist another transcript T_1 where $\hat{\mathcal{P}}$ uses the same randomness ρ_h with fourth round challenge $i_1^* \neq i_0^*$. We now show that given transcripts T_0, T_1 , a uniquely defined witness $\mathbf{x}, \mathbf{r}, \mathbf{t}$ consistent with both transcripts can be extracted.

Let (π_0, i_0^*) and (π_1, i_1^*) be the verifier challenges in transcripts T_0 and T_1 . Recall that $\hat{\mathcal{P}}$ used the same first round randomness ρ_h in T_0 and T_1 , so the first round commitment h is the same in the two transcripts. If the revealed shares of $\llbracket \mathbf{x} \rrbracket$, $\llbracket \mathbf{r} \rrbracket$, and $\langle\langle \mathbf{t} \rangle\rangle$ in the two transcripts are not consistent, then collision resistance of the hash function is violated, since we assume the commitment scheme to be perfectly binding. Thus, the opened shares must be consistent, hence the underlying witness $(\mathbf{x}, \mathbf{r}, \mathbf{t})$ must be unique in T_0 and T_1 .

Next, we show that the witness $(\mathbf{x}, \mathbf{r}, \mathbf{t})$ is a valid witness, i.e. $g^{\mathbf{x}}\mathbf{H}^\top = \mathbf{s}$ and $\mathbf{r} = \mathbf{t} \neq 0$. Suppose the witness is not valid, and in particular $\mathbf{r} \neq \mathbf{t}$. Let bad denote the event that that execution of the MPC protocol with $(\llbracket \mathbf{x} \rrbracket, \pi(\llbracket \mathbf{r} \rrbracket), \pi(\langle\langle \mathbf{t} \rangle\rangle))$ as inputs accepts with $\mathbf{v} = \mathbf{s}$. By definition, $\Pr[\text{bad}] \leq p$. Then,

$$\begin{aligned} & \Pr[\text{Succ}_{\hat{\mathcal{P}}} \mid R_h = \rho_h] = \\ &= \Pr[\text{Succ}_{\hat{\mathcal{P}}} \wedge \text{bad} \mid R_h = \rho_h] + \Pr[\text{Succ}_{\hat{\mathcal{P}}} \wedge \neg \text{bad} \mid R_h = \rho_h] \\ &\leq p + (1 - p) \Pr[\text{Succ}_{\hat{\mathcal{P}}} \mid R_h = \rho_h \wedge \neg \text{bad}]. \end{aligned}$$

Under the condition $\neg \text{bad}$, an honest execution of the MPC protocol with input $(\llbracket \mathbf{x} \rrbracket, \pi(\llbracket \mathbf{r} \rrbracket), \pi(\llbracket \mathbf{t} \rrbracket))$ must result in $\mathbf{v} \neq \mathbf{s}$, so the prover must cheat for the simulation of at least one party to have a successful transcript. On the other hand, by construction all but one party's inputs are revealed, so this probability is at most $\frac{1}{N}$. Therefore,

$$\Pr[\text{Succ}_{\hat{\mathcal{P}}} \mid R_h = \rho_h] \leq p + (1 - p) \frac{1}{N} = \varepsilon,$$

which contradicts the assumption that ρ_h is good. We can thus conclude that the witness $(\mathbf{x}, \mathbf{r}, \mathbf{t})$ corresponding to T_0 and T_1 is valid.

Below we describe the extractor algorithm:

Extractor \mathcal{E} :

- Run $\hat{\mathcal{P}}$ with honest verifier V until a successful transcript T_0 is obtained.
- Do N_1 times:
 - Run $\hat{\mathcal{P}}$ with honest V and same randomness as in T_0 to get transcript T_1 .
 - If T_1 is a successful transcript with $i_{T_0}^* \neq i_{T_1}^*$, extract the unique witness $\mathbf{x}, \mathbf{r}, \mathbf{t}$. If $(\mathbf{x}, \mathbf{r}, \mathbf{t})$ is a valid witness, output \mathbf{x} .

After finding a successful transcript T_0 , the probability of finding the second successful transcript T_1 using the same randomness ρ_h given that ρ_h is good is

$$\begin{aligned} & \Pr[\text{Succ}_{\hat{\mathcal{P}}}^{T_1} \wedge i_0^* \neq i_1^* \mid \rho_h \text{ good}] \\ &= \Pr[\text{Succ}_{\hat{\mathcal{P}}}^{T_1} \mid \rho_h \text{ good}] - \Pr[\text{Succ}_{\hat{\mathcal{P}}}^{T_1} \wedge i_0^* = i_1^* \mid \rho_h \text{ good}] \\ &\geq (1 - \alpha)\hat{\varepsilon} - \frac{1}{N} \\ &\geq (1 - \alpha)\hat{\varepsilon} - \varepsilon \end{aligned}$$

If we choose $N_1 = \frac{\ln 2}{(1 - \alpha)\hat{\varepsilon} - \varepsilon}$, then the extractor \mathcal{E} finds a second successful transcript T_1 with probability at least $1/2$.

Let C be the number of calls made by \mathcal{E} to $\hat{\mathcal{P}}$. \mathcal{E} makes one initial call to obtain T_0 . If T_0 is unsuccessful, then \mathcal{E} makes another $\mathbb{E}[C]$ calls on average. If T_1 is successful, then ρ_h is good with probability α , in which case \mathcal{E} finds a successful T_1 in N_1 calls with probability $1/2$. Otherwise, \mathcal{E} makes N_1 calls and starts from the beginning, making another $\mathbb{E}[C]$ calls on average. This occurs

with probability at most $1 - \alpha/2$. Thus,

$$\begin{aligned}
\mathbb{E}[C] &\leq 1 + (1 - \Pr[\text{Succ}_{\hat{p}}]) \cdot \mathbb{E}[C] + \Pr[\text{Succ}_{\hat{p}}] \cdot (N_1 + \left(1 - \frac{\alpha}{2}\right) \cdot \mathbb{E}[C]) \\
&\leq 1 + (1 - \hat{\varepsilon}) \cdot \mathbb{E}[C] + \hat{\varepsilon} \cdot (N_1 + \left(1 - \frac{\alpha}{2}\right) \cdot \mathbb{E}[C]) \\
&\leq 1 + \hat{\varepsilon} \cdot N_1 + \mathbb{E}[C] \cdot \left(1 - \frac{\hat{\varepsilon}\alpha}{2}\right) \\
&\leq \frac{2}{\alpha\hat{\varepsilon}} \cdot (1 + \hat{\varepsilon} \cdot N_1) \\
&\leq \frac{2}{\alpha\hat{\varepsilon}} \cdot \left(1 + \hat{\varepsilon} \cdot \frac{\ln 2}{(1 - \alpha)\hat{\varepsilon} - \varepsilon}\right)
\end{aligned}$$

If we choose α so that $(1 - \alpha)\hat{\varepsilon} = \frac{1}{2}(\varepsilon + \hat{\varepsilon})$, then $\alpha = \frac{1}{2}\left(1 - \frac{\varepsilon}{\hat{\varepsilon}}\right)$, and

$$\mathbb{E}[C] \leq \frac{4}{\hat{\varepsilon} - \varepsilon} \cdot \left(1 + \hat{\varepsilon} \cdot \frac{2 \ln 2}{\hat{\varepsilon} - \varepsilon}\right).$$

Therefore, the extractor succeeds on average after a number of calls polynomial in $(\hat{\varepsilon} - \varepsilon)^{-1}$, which concludes the proof.

Piecewise Simulability Let $\text{pk} = (\mathbf{s}, \mathbf{H}^\top)$ be an instance of R-SDP. We now build the two simulators A and B of Definition 9.

- $A = (A_1, A_2)$: A_1 computes an error vector \mathbf{e}' such that $\mathbf{e}'\mathbf{H}^\top = \mathbf{s}$ but it is not restricted. Then it picks random vectors $\mathbf{x}', \mathbf{t}', \mathbf{r}'$ such that $g^{\mathbf{x}'} \cdot \frac{\mathbf{r}'}{\mathbf{t}'} = \mathbf{e}'$. Finally, A_1 picks a random permutation π and define $\mathbf{t} = \pi^{-1}(\mathbf{t}')$ and $\mathbf{r} = \pi^{-1}(\mathbf{r}')$. Then A_1 execute the protocols on input $(\mathbf{x}', \mathbf{t}, \mathbf{r}, \pi)$ while A_2 act honestly. By construction the verification at the end is satisfied and since only $N - 1$ shares of \mathbf{t}, \mathbf{r} are revealed the simulation is indistinguishable from a real execution.
- $B = (B_1, B_2)$: B_1 takes as input pk and follows the protocol normally, except that it sets $\llbracket \mathbf{x} \rrbracket_N$ randomly. On input π , B_2 picks a random index i^* , then for all $i \neq i^*$ follows the protocol normally, while it sets $\llbracket \mathbf{v} \rrbracket_{i^*} = \mathbf{s} - \sum_{1 \leq i < N, i \neq i^*} \llbracket \mathbf{v} \rrbracket_i$, so that $\sum_{i=1}^N \mathbf{v}_i = \mathbf{s}$. The only difference between the real execution and the simulated one is that \mathbf{v}_{i^*} is not $\alpha \star \llbracket \mathbf{r} \rrbracket_i \mathbf{H}^\top$, however by construction seed_{i^*} is the only one not being revealed and thus, as per Proposition 1, the simulation is indistinguishable from a real execution.

□

Communication Cost. The challenges by the verifier are excluded from the communication cost since they do not contribute to signature size when the protocol is made non-interactive via the Fiat-Shamir transform. The communication from the prover to the verifier consists of two hashes, the views of $N - 1$ parties not challenged by the verifier, and the commitment for the party challenged by the verifier. For all parties except one, say party N , their entire initial state can be

generated from their seeds, and it takes $\lambda \log(N)$ bits to send $N - 1$ seeds of a Tree PRG. For party N , the initial state also has to include $\llbracket \mathbf{x} \rrbracket_N$ and $\langle\langle \mathbf{t} \rangle\rangle_N$, which have sizes $n \lceil \log(z) \rceil$ and $n \lceil \log(q) \rceil$ respectively. Finally, the broadcast share $\llbracket \alpha \rrbracket_{i^*}$ and cmt_{i^*} , with sizes $n \lceil \log(q) \rceil$ and 2λ , are also sent. The protocol must be repeated τ times to achieve the desired soundness. However, the hashes for all the iterations of the protocol can be combined, so only two hashes need to be sent. Thus, the protocol's total cost of communication is at most

$$4\lambda + \tau(\lambda \log(N) + n \lceil \log(z) \rceil + 2n \lceil \log(q) \rceil + 2\lambda).$$

4.1 Computing the cheating probability

To analyze the security of our eventual signature scheme and determine the required MPCitH parameters, a bound on the cheating probability p for the first challenge (the permutation π) needs to be computed.

We consider a prover that knows some error vector \mathbf{e}' that satisfies the syndrome equation $(\mathbf{e}' \mathbf{H}^\top = \mathbf{s})$, but is not restricted ($\mathbf{e}' \notin \mathbb{G}^n$). The prover can try to cheat by manipulating \mathbf{r} , \mathbf{t} , and \mathbf{x} to end up with $\llbracket \mathbf{e}' \rrbracket$ in the protocol for the syndrome equation check. The vectors \mathbf{r} and \mathbf{t} are shuffled by a random permutation π before being used to compute the converted shares, so the prover cheats successfully when $\pi(\mathbf{r}/\mathbf{t}) = \mathbf{r}/\mathbf{t}$.

Notice that if two components of the prover's error vector $e'_j, e'_{j'}$, lie in the same coset of \mathbb{G} , then the prover can always choose \mathbf{r} , \mathbf{t} , and \mathbf{x} such that $\frac{r_j}{t_j} = \frac{r_{j'}}{t_{j'}}$. In this case, the cheating probability is then the probability that $\pi(e_j)$ and e_j lie in the same coset of \mathbb{G} for all $j = 1, \dots, n$. This is equivalent to the probability that $\mathbf{e}^* = \pi(\mathbf{e}^*)$ for some $\mathbf{e}^* \in \mathbb{F}_I^n$, where I is the number of cosets of \mathbb{G} , so $I = \frac{q-1}{z}$. For a fixed \mathbf{e}^* with k_1 components which are the first field element, k_2 components which are the second field element, etc., the probability that π fixes \mathbf{e}^* is

$$\frac{k_1! \cdots k_I!}{n!}.$$

Assuming that \mathbf{e}' is uniformly distributed, then the average cheating probability is

$$\sum_{k_1 + \dots + k_I = n} \frac{\binom{n}{k_1, \dots, k_I}}{I^n} \cdot \frac{k_1! \cdots k_I!}{n!} = \sum_{k_1 + \dots + k_I = n} \frac{1}{I^n} = \frac{\binom{n+I-1}{I-1}}{I^n}.$$

This suggests that for reasonably chosen parameters, the probability should be negligible compared to $\frac{1}{N}$, where N is the number of parties in the MPC simulation, for most choices of \mathbf{e}' .

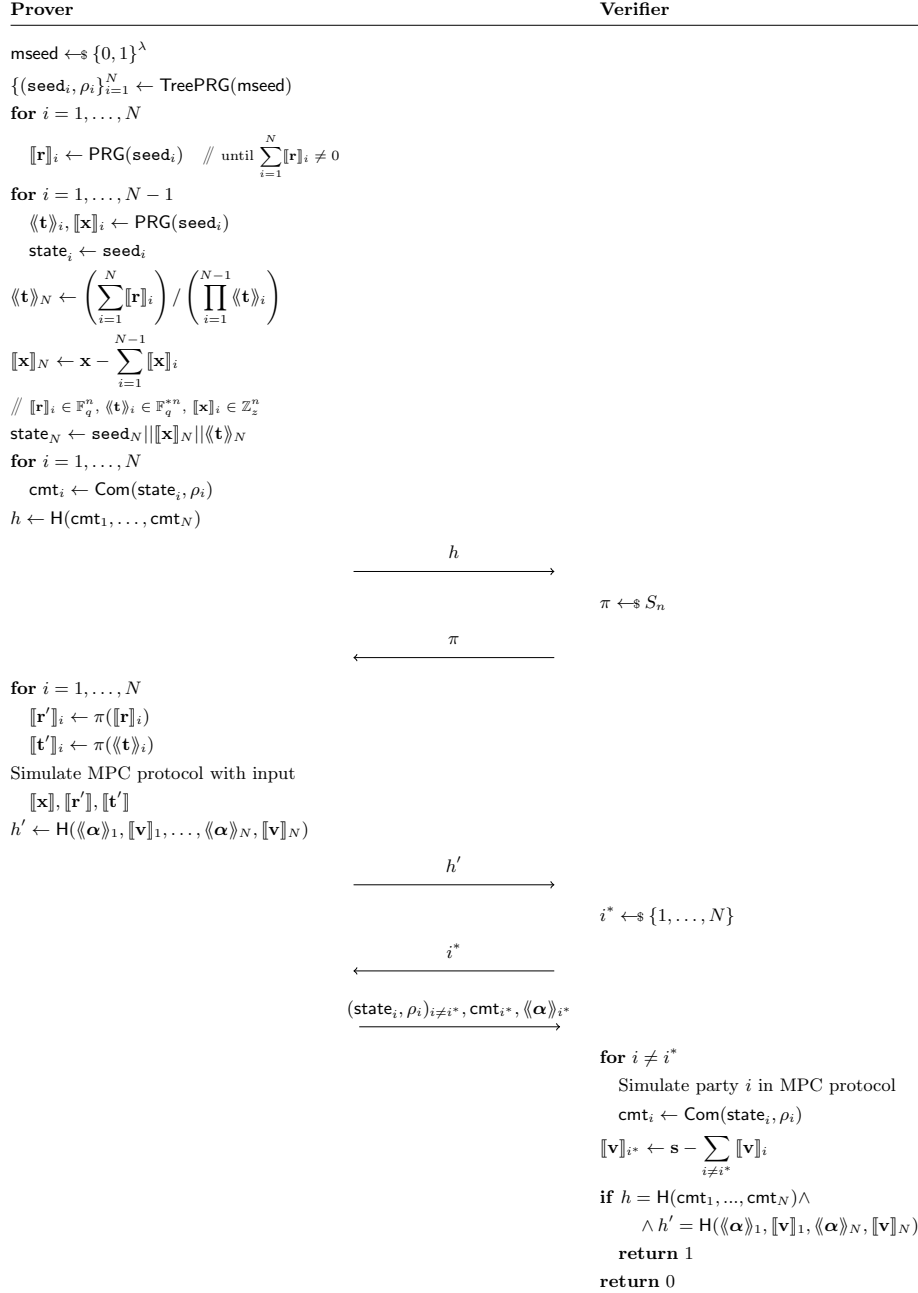


Fig. 2. Zero-Knowledge protocol for RSDP.

5 Digital Signatures

A signature scheme consists of three PPT algorithms: key generation, signing, and verification. Key generation takes as input 1^λ for security parameter λ and outputs a key pair (sk, pk) . The signing algorithm takes a message msg and the secret key sk as input and outputs a signature sig . The verification algorithm takes as input a message msg , a signature sig and a public key pk , outputs 1 if sig is a valid signature for msg under pk , and outputs 0 otherwise. Informally, a signature scheme is secure if an adversary, given oracle access to the signing algorithm for sk , cannot efficiently forge a signature for a message that was not queried to the oracle.

The HVZK protocol described previously can be turned into a signature scheme by applying the Fiat-Shamir transform [23]. Instead of having the verifier issue challenges, the prover computes challenges deterministically from the commitments and the message to be signed without interacting with the verifier. The signature then consists of the commitments and the responses to the challenges. The verifier can compute the challenges in the same way, and verify the signature by checking that the commitments, challenges, and responses form a valid transcript of the zero-knowledge protocol.

5.1 Description of signature scheme

Key generation. Taking as inputs the security parameter λ and the restriction \mathbb{G} , an instance of R-SDP is randomly sampled from a seed $\text{mseed} \in \{0, 1\}^{2\lambda}$. Using mseed , a public seed seed_{pk} and the exponents for the error vector $\mathbf{x} \in \mathbb{F}_z^n$ are generated from a PRG. Then, seed_{pk} is used to sample the parity check matrix $\mathbf{H} \in \mathbb{F}_q^{(n-k) \times n}$, and the syndrome is computed as $\mathbf{s} = \mathbf{e}\mathbf{H}^\top$.

The secret key sk is mseed , and the public key pk consists of the seed seed_{pk} and the syndrome \mathbf{s} . Thus, the key sizes are

$$|\text{sk}| = 2\lambda, |\text{pk}| = 2\lambda + (n - k) \log(q).$$

Signing. To sign a message \mathbf{m} , one simply follows the prover's steps in the zero-knowledge protocol described previously, except they must compute their own challenges. To prevent seed collisions, an additional random string $\text{salt} \in \{0, 1\}^{2\lambda}$ is introduced [18]. Challenges $\text{ch}_1 = \pi, \text{ch}_2 = i^*$ are computed as follows:

- $h_1 = \text{H}(\mathbf{m}, \text{salt}, h)$
- $\text{ch}_1 = \text{PRG}(h_1)$
- $h_2 = \text{H}(\mathbf{m}, \text{salt}, h, h')$
- $\text{ch}_2 = \text{PRG}(h_2)$

The final signature then consists of salt, h_1, h_2 , and the communication from the prover to the verifier in the zero-knowledge protocol required for verification $(\{\text{state}_i, \rho_i\}_{i \neq i^*}, \text{cmt}_{i^*}, \llbracket \boldsymbol{\alpha} \rrbracket_{i^*})$.

Verification. First, the challenges are computed from h_1 and h_2 included in the signature. Then, the verifier follows the zero-knowledge protocol to recompute the hashes and check that they match h_1, h_2 .

5.2 Signature Sizes

Choice of parameters The R-SDP parameters are chosen according to the CROSS specification document [7]. q and z are always 127 and 7 respectively, while n and k depend on the desired security level.

The MPCitH parameters τ (number of repetitions) and N (number of parties) need to be chosen so that the cost of forgery is greater than 2^λ . Signatures constructed from 5-round zero-knowledge protocols are subject to the attack presented in [27], which results in a cost of forgery of

$$\min_{\tau_1, \tau_2: \tau_1 + \tau_2 = \tau} \left\{ \frac{1}{\sum_{i=\tau_1}^{\tau} \binom{\tau}{i} p^i (1-p)^{\tau-i}} + N^{\tau_2} \right\},$$

where p is the false positive probability for the first challenge.

For “fast” variants of the signature schemes, we choose $N = 32$, and for “short” variants, we choose $N = 256$. Then, τ is computed to satisfy the security requirements.

In our protocol, the probability of cheating the first challenge, p , depends on how the number of components of the error vector in each coset of \mathbb{G} is distributed. We call this the coset distribution of the error vector. Assuming the error vector is uniformly distributed, with high probability, p is negligible compared to $\frac{1}{N}$, so we first choose τ such that $(\frac{1}{N})^\tau < 2^\lambda$. Then, we verify computationally that with high enough probability ($> 1 - 2^{-\lambda}$), p is small enough such that the cost of forgery from the attack in [27] attains the desired security level ($\geq 2^\lambda$). As an example, we show the computation for $\lambda = 128$, $N = 256$, and $\tau = 17$ below.

First, we compute the maximum value of p such that the cost of forgery is greater than 2^{128} , i.e., the maximum p satisfying

$$\min_{1 \leq \tau_1 \leq \tau} \left\{ \frac{1}{\sum_{i=\tau_1}^{\tau} \binom{\tau}{i} p^i (1-p)^{\tau-i}} + N^{\tau-\tau_1} \right\} \geq 2^{128}.$$

For the parameters listed above, it can be verified numerically that p is around 2^{-68} . Then, we want to compute the probability of a random error vector having a coset distribution with corresponding p that is greater than 2^{-68} , and check that this probability is less than 2^{-128} . For a fixed error vector, $p = \frac{k_1! \cdots k_I!}{n!}$, and we only consider error vectors with $p > 2^{-68}$. Let D be the set of coset distributions with corresponding $p > 2^{-68}$. Then,

$$\sum_D \frac{\binom{n}{k_1, \dots, k_I}}{I^n} < \sum_{k_1 + \dots + k_{18} = 127} \frac{2^{68}}{18^{127}} = \binom{127 + 18 - 1}{18 - 1} \cdot \frac{2^{68}}{18^{127}} < 2^{-389} < 2^{-128}.$$

Comparison of signature sizes. Below is a comparison of signature sizes between CROSS and our MPCitH scheme. Parameters and signature sizes for CROSS are taken from [7]. Although different choices for the finite field or the restriction set could possibly yield better performance, following the example of [12], we choose instead to keep the parameters used in the original CROSS submission for comparative purposes.

Scheme	Signature size	Variant	λ	n	k	τ	w	Bytes
CROSS	$6\lambda + 3\lambda w + (\tau - w)(2\lambda + n\lceil\log(q)\rceil) + n\lceil\log(z)\rceil$	fast	128	127	76	157	82	18432
		fast	192	187	111	239	125	41406
		fast	256	251	150	321	167	74590
		short	128	127	76	520	488	12432
	$6\lambda + (\tau - w)(n\lceil\log(q)\rceil + n\lceil\log(z)\rceil) + 3\lambda\left(\lfloor(\tau - w)\log(\frac{\tau}{\tau - w})\rfloor + wt_H(\tau) - 1\right)$	short	192	187	111	580	527	28391
		short	256	251	150	832	762	50818
MPCitH	$6\lambda + \tau(\lambda\log(N) + n\lceil\log(z)\rceil) + 2n\lceil\log(q)\rceil + 2\lambda$	fast	128	127	76	26	-	10080
		fast	192	187	111	39	-	22257
		fast	256	251	150	52	-	39660
		short	128	127	76	17	-	7440
		short	192	187	111	25	-	16119
		short	256	251	150	33	-	28407

Table 1. Comparison of signature sizes between CROSS and our MPCitH scheme.

5.3 Computational Benchmarking

While we are currently lacking a concrete implementation, we provide here an analysis of its theoretical performance. Specifically, we focus on the most resource-intensive operations, namely the total number of cryptographic hash computations and vector-matrix multiplications required. Let N the number of parties in the MPCitH protocol and τ the number of parallel repetition as per Section 5.2. In a single repetition, the signature algorithm requires

- $2N - 1$ PRG calls for the N seeds \mathbf{seed}_i .
- N PRG calls for the shares of \mathbf{r} .
- $2(N - 1)$ PRG calls for the shares of \mathbf{t} and \mathbf{x} .
- N PRG calls for the computation of all commitments \mathbf{cmt}_i .
- 2 hash calls for h and h' .
- N vector-matrix multiplications for MPC protocol.

We also need to add two hashes for π and i^* . Thus for a signature we have $(6N - 1)\tau + 2$ hash calls and $N\tau$ vector-matrix multiplications. To verify the signature, we have

- $N - 1$ PRG calls for the computation of all commitments \mathbf{cmt}_i .
- 2 hash calls for h and h' .
- $N - 1$ vector-matrix multiplications for MPC protocol.

This we have a total of $(N + 1)\tau$ hash calls and $(N - 1)\tau$ vector-matrix multiplications.

The short CROSS requires (here w is the challenge weight) $5\tau + 8$ PRG calls and 2τ vector-matrix multiplications to sign and $w + 2\tau + 7$ PRG calls and 3τ vector-matrix multiplications to verify a signature. On the other hand, the fast version reduces the number of operations, requiring only $w + 2\tau + 7$ PRG calls and τ vector-matrix multiplications and $\tau + w + 13$ PRG calls and τ vector-matrix multiplications to verify.

Scheme	λ	n	k	τ	w	Bytes	PRG S Mult. S	PRG V Mult. V		
CROSS Short	128	127	76	520	488	12432	2608	1040	1535	1560
	192	187	111	580	527	28391	2908	1160	1694	1740
	256	251	150	832	762	50818	4168	1664	2433	2496
CROSS Fast	128	127	76	157	82	18432	403	157	252	157
	192	187	111	239	125	41406	610	239	377	239
	256	251	150	321	167	74590	816	321	501	321
MPCitH Short	128	127	76	17	-	7440	26095	4352	4369	4335
	192	187	111	25	-	16119	38377	6400	6425	6375
	256	251	150	33	-	28407	50657	8448	8481	8415
MPCitH Fast	128	127	76	26	-	10080	4968	832	858	806
	192	187	111	39	-	22257	7451	1248	1287	1209
	256	251	150	52	-	39660	9934	1664	1716	1612

Table 2. Comparison of the number of operations between CROSS and our MPCitH scheme.

We see that the fast version of our scheme requires significantly more PRG calls and roughly the same number of vector matrix-multiplication to sign, while having slightly lower number of operations for verification and shorter signatures compared to the short version of CROSS. On the other hand, the short version of our scheme offers a more extreme trade-off, where the number of operations is significantly higher but offering way smaller signature sizes.

6 Optimization via the Hypercube Technique

The hypercube technique proposed in [4] can be generically applied to MPCitH-based protocols [25] to reduce computation and communication costs (signature sizes). Below we briefly describe the technique and its application to our protocol for restricted syndrome decoding.

In the standard MPCitH paradigm, secret shares are generated for N parties, and an MPC protocol is simulated with the N parties to obtain a zero-knowledge protocol with soundness error of $1/N$. For simplicity, assume $N = 2^D$. With hypercube MPCitH, secret shares are generated for 2^D parties, and soundness error of $1/2^D$ can be achieved by performing D simulations of the MPC protocol,

in which only $1 + D$ parties are simulated in total. This is done by leveraging a geometric hypercube structure when generating secret shares.

We call each of the 2^D parties holding a secret share a leaf party, and we think of the leaf parties as being arranged in a D -dimensional hypercube, indexed by $(i_1, \dots, i_D) \in \{0, 1\}^D$. For each dimension $k \in \{1, \dots, D\}$, we designate a main party (k, j) who holds an aggregate share of all the shares of the (2^{D-1}) leaf parties whose k^{th} coordinate is j , i.e.,

$$\llbracket x \rrbracket_{(k,j)} = \sum_{i_1, \dots, i_{k-1}, i_{k+1}, \dots, i_N \in \{0,1\}} \llbracket x \rrbracket_{(i_1, \dots, i_{k-1}, j, i_{k+1}, \dots, i_N)}.$$

Now, for each dimension k , the aggregate shares $\llbracket x \rrbracket_{(k,0)}, \llbracket x \rrbracket_{(k,1)}$ also form a 2-party secret sharing. Simulating D instances of the MPC protocols using the D aggregate secret sharings then gives soundness error of $1/2^D$ (see [4], [25] for formal proofs). Furthermore, only $1 + D$ parties need to actually be simulated, since 2 aggregate parties are simulated for the first instance, and after this the outputs for all D instances are known since the same secret is the input to all instances. Then, only all but one (in this case 1) of the aggregate parties need to be simulated for subsequent instances of the MPC protocol, as the last party's share can be deduced. The communication cost of a zero-knowledge protocol employing the hypercube technique does not change, since revealing the shares of all but one of the leaf parties is equivalent to revealing the shares of all but one of each of the aggregate parties. The amount of computation is significantly reduced compared to standard MPCitH, where to achieve the same soundness, 1 MPC protocol with 2^D parties would have to be simulated.

It was shown in [25] that the hypercube technique can be applied to any MPCitH protocol relying on additive secret sharing. It is likewise applicable to multiplicative secret sharing, since multiplication has the same associative and commutative properties as addition. Thus, the hypercube technique can be applied to our MPCitH protocol for R-SDP as well. In practice, there are two ways in which this could be beneficial. The first is to increase the number of MPC parties N while maintaining similar computational efficiency, which would reduce the number of required repetitions of the protocol to reach the desired soundness. This would result in smaller signature sizes, further adding to our advantage over CROSS in this respect. Alternatively, the parameter N could be kept the same, leading to an improved computational efficiency, reducing the overhead in this respect in comparison to CROSS.

Below, we show reduced signature sizes for our MPCitH scheme by increasing the number of MPC parties N , which is feasible due to the hypercube optimization. The R-SDP parameters are chosen according to [7], and the number of iterations τ is computed as described in Section 5.2.

λ	N	τ	Bytes
128	2^8	17	7440
128	2^{10}	13	6128
128	2^{12}	11	5552
192	2^8	25	16 119
192	2^{10}	20	13 884
192	2^{12}	17	11 823
256	2^8	33	28 407
256	2^{10}	26	24 086
256	2^{12}	22	21 818

Table 3. Signature sizes for MPCitH scheme with increased number of MPC parties.

As we can see, the final signature size is roughly 40% – 50% compared to the original CROSS submission. Table 4 shows the signature size of our protocols compared to the other VOLEitH and MPCitH protocols in the NIST competition, for 128 bits of security.

MPCitH Schemes	Signature Size (Bytes)
Our Contribution	5552
FAEST	3096
Mirath	2902
MQOM	2820
PERK	5780
RYDE	2988
SDitH	3705

Table 4. Signature sizes for MPCitH and VOLEitH schemes in the NIST competition. Only the shorter version for each protocol is reported.

6.1 Threshold secret-sharing

Additive secret sharing used in the MPC protocol can generally be replaced with threshold secret-sharing schemes (such as Shamir’s secret sharing [32]) to improve the soundness from $1/N$ to $1/\binom{N}{\ell}$ for $(\ell + 1)$ -threshold schemes [21, 22]. Applying a low-threshold scheme can also significantly reduce the number of parties that need to be simulated by the prover and the verifier, removing dependence on N . However, Shamir’s secret sharing is only additively homomorphic, and the share conversion protocol would also require a multiplicative threshold secret-sharing scheme. Rather than requiring both a multiplicative sharing and an additive sharing of a random value in the preprocessing phase of the sharing conversion protocol, we can instead generate additive shares of r and g^r and treat $g^{\llbracket r \rrbracket}$ as the

multiplicative sharing. While this reduces the possibilities of \mathbf{r} to vectors in \mathbb{G}^n , it does not affect security as the cost of recovering \mathbf{e} by guessing the uniformly distributed \mathbf{r} is equal to simply guessing $\mathbf{e} \in \mathbb{G}^n$ randomly.

7 Conclusions

This work successfully introduces a novel Multi-Party Computation-in-the-Head (MPCitH) protocol tailored for the Restricted Syndrome Decoding Problem (R-SDP), significantly reducing the CROSS signature size to almost 40% with respect to the original protocol and making it only slightly bigger compared to the other MPCitH-based protocols. In this sense our protocol offers a valuable alternative to the short version of CROSS, proposing a more extreme trade-off of computational efficiency for more favourable signature sizes.

Future works A variant of R-SDP is the subgroup R-SDP [9], known as R-SDP(G), which further restricts error vectors to a subgroup G of \mathbb{G}^n generated by $\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(m)} \in \mathbb{G}^n$, where the group operation is component-wise multiplication. Equivalently, G can be represented as a code generated by the matrix \mathbf{M}_G , where the j th row of \mathbf{M}_G is $\mathbf{i}^{(j)}$ where $\mathbf{x}^{(j)} = g^{\mathbf{i}^{(j)}}$. Then, each codeword generated by this matrix corresponds to the exponents of a restricted vector in G . Signature schemes based on R-SDP(G) allow for even smaller signature sizes, since error vectors can now be represented as exponents for each of the generators of G .

The share conversion protocol can be modified into a protocol for R-SDP(G) in a straightforward way: Let \mathbf{u} be the vector of exponents for the generators of G corresponding to the error vector $\mathbf{e} \in G$, i.e.,

$$\mathbf{e} = (\mathbf{x}^{(1)})^{u_1} \star \dots \star (\mathbf{x}^{(m)})^{u_m} = (g^{\sum_{j=1}^m u_j i_1^{(j)}}, \dots, g^{\sum_{j=1}^m u_j i_n^{(j)}}).$$

To convert shares of \mathbf{u} into shares of \mathbf{e} , each party first locally computes $\llbracket \mathbf{y} \rrbracket_i = \llbracket \mathbf{u} \rrbracket_i \mathbf{M}_G$. Then, $g^{\llbracket \mathbf{y} \rrbracket}$ can be treated as a multiplicative sharing of \mathbf{e} , and the share conversion protocol can be applied. Generating $\llbracket \mathbf{e} \rrbracket$ in this manner implicitly ensures that $\mathbf{e} \in G$, and $\llbracket \mathbf{e} \rrbracket$ can then be used to check the syndrome equation.

The protocol for R-SDP(G) has smaller communication since the size of \mathbf{u} ($m \lceil \log(z) \rceil$) is smaller than the size of \mathbf{x} ($n \lceil \log(z) \rceil$) in the protocol for R-SDP after adjusting the parameters to maintain security. However, separate analysis of the soundness of a protocol based on R-SDP(G) is still needed.

Acknowledgements

The first author is supported by the Italian Ministry of University and Research (MUR) under the PRIN 2022 program with projects “Mathematical Primitives for Post Quantum Digital Signatures” (CUP I53D23006580001) and “Post quantum Identification and eNcryption primiTi ves: dEsign and Realization (POINTER)” (CUP I53D23003670006), by project SERICS (PE00000014) under the MUR

National Recovery and Resilience Plan, funded by the European Union - Next Generation EU and by MUR under the Italian Fund for Applied Science (FISA 2022), Call for tender No. 1405 published on 13-09-2022 - project title “Quantum-safe cryptographic tools for the protection of national data and information technology assets” (QSAFEIT) - No. FISA 2022-00618 (CUP I33C24000520001), Grant Assignment Decree no. 15461 adopted on 02.08.2024. The last author wants to thank the support of the Technical University of Munich – Institute for Advanced Study, funded by the German Excellence Initiative.

References

- [1] N. Aaraj, S. Bettaieb, L. Bidoux, A. Budroni, V. Dyseryn, A. Esser, T. Feneuil, P. Gaborit, M. Kulkarni, V. Mateu, M. Palumbi, L. Perin, M. Rivain, J. Tillich, and K. Xagawa. *PERK*. Tech. rep. available at <https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures>. National Institute of Standards and Technology, 2024.
- [2] G. Adj, N. Aragon, S. Barbero, M. Bardet, E. Bellini, L. Bidoux, J. Chi-Domínguez, V. Dyseryn, A. Esser, T. Feneuil, P. Gaborit, R. Neveu, M. Rivain, L. Rivera-Zamarripa, C. Sanna, J. Tillich, J. Verbel, and F. Zweyding. *Mirath (merger of MIRA/MiRitH)*. Tech. rep. available at <https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures>. National Institute of Standards and Technology, 2024.
- [3] C. Aguilar Melchor, S. Bettaieb, L. Bidoux, T. Feneuil, P. Gaborit, N. Gama, S. Gueron, J. Howe, A. Hülsing, D. Joseph, A. Joux, M. Kulkarni, E. Persichetti, T. H. Randrianarisoa, M. Rivain, and D. Yue. *SDitH — Syndrome Decoding in the Head*. Tech. rep. available at <https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures>. National Institute of Standards and Technology, 2024.
- [4] C. Aguilar-Melchor, N. Gama, J. Howe, A. Hülsing, D. Joseph, and D. Yue. “The Return of the SDitH”. In: *Advances in Cryptology – EUROCRYPT 2023*. 2023, 564–596.
- [5] G. Alagic, M. Bros, P. Ciadoux, D. Cooper, Q. Dang, T. Dang, J. Kelsey, J. Lichtinger, Y.-K. Liu, C. Miller, et al. “Status Report on the First Round of the Additional Digital Signature Schemes for the NIST Post-Quantum Cryptography Standardization Process”. In: *NIST IR 8528* (2024).
- [6] N. Aragon, M. Bardet, L. Bidoux, J. Chi-Domínguez, V. Dyseryn, T. Feneuil, P. Gaborit, A. Joux, R. Neveu, M. Rivain, J. Tillich, and A. Vinçotte. *RYDE*. Tech. rep. available at <https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures>. National Institute of Standards and Technology, 2024.
- [7] M. Baldi, A. Barengi, M. Battagliola, S. Bitzer, M. Gianvecchio, P. Karl, F. Manganiello, A. Pavoni, G. Pelosi, P. Santini, J. Schupp, E. Signorini, F. Slaughter, A. Wachter-Zeh, and V. Weger. *CROSS — Codes and Restricted Objects Signature Scheme*. Tech. rep. available at <https://csrc.nist.gov/>

- Projects/pqc-dig-sig/round-2-additional-signatures. National Institute of Standards and Technology, 2024.
- [8] M. Baldi, M. Battaglioni, F. Chiaraluce, A.-L. Horlemann, E. Persichetti, P. Santini, and V. Weger. “A new path to code-based signatures via identification schemes with restricted errors”. In: *Advances in Mathematics of Communications* 19.5 (2025), pp. 1360–1381.
 - [9] M. Baldi, S. Bitzer, A. Pavoni, P. Santini, A. Wachter-Zeh, and V. Weger. “Zero Knowledge Protocols and Signatures from the Restricted Syndrome Decoding Problem”. In: *Public-Key Cryptography – PKC 2024*. 2024, pp. 243–274.
 - [10] S. Barg. “Some new NP-complete coding problems”. In: *Problemy Peredachi Informatsii* 30.3 (1994), pp. 23–28.
 - [11] M. Battagliola, R. Longo, F. Pintore, and et al. “A Revision of CROSS Security: Proofs and Attacks for Multi-Round Fiat–Shamir Signatures”. In: *Mediterr. J. Math.* 22.1 (2025), p. 132. DOI: 10.1007/s00009-025-02882-7.
 - [12] M. Battagliola, S. Bitzer, A. Wachter-Zeh, and V. Weger. *MPCitH-based Signatures from Restricted Decoding Problems*. 2025. arXiv: 2510.11224 [cs.CR]. URL: <https://arxiv.org/abs/2510.11224>.
 - [13] C. Baum, L. Braun, W. Beullens, C. Delpech de Saint Guilhem, M. Klooß, C. Majenz, S. Mukherjee, E. Orsini, S. Ramacher, C. Rechberger, L. Roy, and P. Scholl. *FAEST*. Tech. rep. available at <https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures>. National Institute of Standards and Technology, 2024.
 - [14] R. Benadjila, C. Bouillaguet, T. Feneuil, and M. Rivain. *MQOM — MQ on my Mind*. Tech. rep. available at <https://csrc.nist.gov/Projects/pqc-dig-sig/round-2-additional-signatures>. National Institute of Standards and Technology, 2024.
 - [15] E. Berlekamp, R. McEliece, and H. van Tilborg. “On the inherent intractability of certain coding problems (Corresp.)” In: *IEEE Transactions on Information Theory* 24.3 (1978), pp. 384–386.
 - [16] W. Beullens. “Sigma Protocols for MQ, PKP and SIS, and Fishy Signature Schemes”. In: *Advances in Cryptology – EUROCRYPT 2020*. 2020, 183–211.
 - [17] E. Carozza, G. Couteau, and A. Joux. “Short Signatures from Regular Syndrome Decoding in the Head”. In: *Advances in Cryptology – EUROCRYPT 2023*. 2023, 532–563.
 - [18] A. Chailloux and S. Etinski. “On the (in)security of optimized Stern-like signature schemes”. In: *Designs, Codes and Cryptography* 92 (2024), pp. 803–832.
 - [19] D. Evans, V. Kolesnikov, and M. Rosulek. *A Pragmatic Introduction to Secure Multi-Party Computation*. Now Publishers, 2022.
 - [20] T. Feneuil, A. Joux, and M. Rivain. “Syndrome Decoding in the Head: Shorter Signatures from Zero-Knowledge Proofs”. In: *Advances in Cryptology – CRYPTO 2022*. 2022, pp. 541–572.

- [21] T. Feneuil and M. Rivain. “Threshold Computation in the Head: Improved Framework for Post-Quantum Signatures and Zero-Knowledge Arguments”. In: *Journal of Cryptology* 38.28 (2025).
- [22] T. Feneuil and M. Rivain. “Threshold Linear Secret Sharing to the Rescue of MPC-in-the-Head”. In: *Advances in Cryptology – ASIACRYPT 2023*. 2023, 441–473.
- [23] A. Fiat and A. Shamir. “How To Prove Yourself: Practical Solutions to Identification and Signature Problems”. In: *Advances in Cryptology — CRYPTO’ 86*. 1987, pp. 186–194.
- [24] O. Goldreich, S. Goldwasser, and S. Micali. “How to construct random functions”. In: *25th Annual Symposium on Foundations of Computer Science*. 1984, pp. 464–479.
- [25] A. Hülsing, D. Joseph, C. Majenz, and A. K. Narayanan. “On round elimination for special-sound multi-round identification and the generality of the hypercube for MPCitH”. In: *Advances in Cryptology — CRYPTO 2024*. 2024, 373–408.
- [26] Y. Ishai, E. Kushilevitz, R. Ostrovsky, and A. Sahai. “Zero-Knowledge from Secure Multiparty Computation”. In: *Proceedings of the Thirty-Ninth Annual ACM Symposium on Theory of Computing*. 2007, 21–30.
- [27] D. Kales and G. Zaverucha. “An Attack on Some Signature Schemes Constructed from Five-Pass Identification Schemes”. In: *Cryptology and Network Security: 19th International Conference, CANS 2020*. 2020, 3–22.
- [28] D. Kales and G. Zaverucha. “An Attack on Some Signature Schemes Constructed from Five-Pass Identification Schemes”. In: *CANS 20*. Ed. by S. Krenn, H. Shulman, and S. Vaudenay. Vol. 12579. LNCS. Springer, Cham, Dec. 2020, pp. 3–22. DOI: 10.1007/978-3-030-65411-5_1.
- [29] J. Katz, V. Kolesnikov, and X. Wang. “Improved Non-Interactive Zero Knowledge with Applications to Post-Quantum Signatures”. In: *Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security*. 2018, 525–537.
- [30] Y. Lindell. *How To Simulate It – A Tutorial on the Simulation Proof Technique*. Cryptology ePrint Archive, Paper 2016/046. 2025. URL: <https://eprint.iacr.org/2016/046.pdf>.
- [31] J. Maire and D. Vergnaud. “Efficient Zero-Knowledge Arguments and Digital Signatures via Sharing Conversion in the Head”. In: *Computer Security – ESORICS 2023*. 2024, pp. 435–454.
- [32] A. Shamir. “How to share a secret”. In: *Commun. ACM* 22.11 (1979), 612–613.
- [33] Z. Xia, Q. Gu, W. Zhou, L. Xiong, J. Weng, and N. Xiong. “STR: Secure Computation on Additive Shares Using the Share-Transform-Reveal Strategy”. In: *IEEE Transactions on Computers* 73.2 (2024), pp. 340–352.