MiRitH: Efficient Post-Quantum Signatures from MinRank in the Head

IACR Transactions on Cryptographic Hardware and Embedded Systems
ISSN 2569-2925, Vol. 2024, No. 2, pp. 304–328. DOI:10.46586/tches.v2024.i2.304-328

MiRitH: Efficient Post-Quantum Signatures from MinRank in the Head

Gora Adj[1], Stefano Barbero[2], Emanuele Bellini[1], Andre Esser[1]*, Luis Rivera-Zamarripa[1], Carlo Sanna[2]†, Javier Verbel[1] and Floyd Zweydinger[1]

[1] Technology Innovation Institute, Abu Dhabi, UAE
{gora.adj,emanuele.bellini,andre.esser,luis.zamarripa,javier.verbel,floyd.zweydinger}@tii.ae
[2] Politecnico di Torino, Torino, Italy
stefano.barbero,carlo.sanna@polito.it

Abstract. Since 2016's NIST call for standardization of post-quantum cryptographic primitives, developing efficient post-quantum secure digital signature schemes has become a highly active area of research. The difficulty in constructing such schemes is evidenced by NIST reopening the call in 2022 for digital signature schemes, because of missing diversity in existing proposals. In this work, we introduce the new post-quantum digital signature scheme MiRitH. As direct successor of a scheme recently developed by Adj, Rivera-Zamarripa and Verbel (Africacrypt '23), it is based on the hardness of the MinRank problem and follows the MPC-in-the-Head paradigm. We revisit the initial proposal, incorporate design-level improvements and provide more efficient parameter sets. We also provide the missing justification for the quantum security of all parameter sets following NIST metrics. In this context we design a novel Grover-amplified quantum search algorithm for solving the MinRank problem that outperforms a naive quantum brute-force search for the solution.
MiRitH obtains signatures of size 5.7 kB for NIST category I security and therefore competes for the smallest signatures among any post-quantum signature following the MPCitH paradigm.
At the same time MiRitH offers competitive signing and verification timings compared to the state of the art. To substantiate those claims we provide extensive implementations. This includes a reference implementation as well as optimized constant-time implementations for Intel processors (AVX2), and for the ARM (NEON) architecture. The speedup of our optimized AVX2 implementation relies mostly on a redesign of the finite field arithmetic, improving over existing implementations as well as an improved memory management.

Keywords: Digital Signature · MinRank · MPCitH · Post-Quantum · ZKPoK · Quantum Analysis

1 Introduction

The development of digital signature schemes that are efficient *and* post-quantum secure is a persistent challenge. As so, the first standardization process for such schemes launched

*supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – Project-ID MA 2536/12

†C. Sanna was partially supported by the project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union – NextGenerationEU; and by the project QUBIP (https://www.qubip.eu) under the Horizon Europe framework programme [grant agreement no. 101119746] funded by the European Union.

Received: 2023-10-15 Accepted: 2023-12-15 Published: 2024-03-12

in 2016 by the National Institute of Standards and Technology (NIST) failed to achieve the desired level of diversity among final candidates. This led to NIST initiating a renewed call for proposals whose submission deadline recently passed in June 2023. For this new standardization effort to be successful significant research efforts are required. In order to solve the problem of missing diversity, new efficient schemes based on different hardness assumptions are required or established schemes need to be enhanced to reach a competitive level. This requires a thorough examination on all levels of the design process of those schemes, including their initial construction, parameter selection, security justification as well as improved implementations for different platforms.

In this work we contribute to this challenge of defining the next post-quantum secure standards by presenting the new post-quantum digital signature scheme MiRitH, currently submitted to the renewed NIST call. At its core MiRitH is based on the recently proposed scheme by Adj, Rivera-Zamarripa and Verbel [ARZV23], which itself is constructed via the Fiat–Shamir transform from a zero-knowledge proof of knowledge (ZKPoK) following the MPC-in-the-Head (MPCitH) paradigm. The security of the ARZV scheme and correspondingly the security of MiRitH rely on the hardness of the well-established MinRank problem.

MiRitH extends and improves over the initial proposal from [ARZV23] in various ways, by leveraging recent techniques from the literature as well as by introducing new design improvements. We provide improved parameters and the so far missing quantum-security justification for all parameter sets. Furthermore, we provide extensive implementations for the scheme, including a reference implementation, as well as optimized constant-time implementations for Intel processors (AVX2) and the ARM (NEON) architecture. Overall, MiRitH positions as a competitive post-quantum secure digital signature scheme.

Fiat–Shamir and MPCitH Signature schemes constructed from ZKPoKs via the Fiat–Shamir transform have been popular ever since. In the context of solving the diversity challenges of currently available post-quantum secure signature proposals those constructions have received even more attention. Such schemes offer an alternative to trapdoor constructions by allowing to base security on random instances of well-established problems. However, one significant limitation of these constructions has been their relatively large signature sizes.

When transforming a ZKPoK into a signature scheme, the signature size is typically proportional to the communication cost of the underlying ZK protocol. Consequently, a substantial portion of the signature is composed of protocol-related messages, such as commitments and auxiliary information, which are independent of the chosen problem foundation. However, the increased interest in these constructions has led to various improvements at the protocol level [IKOS07, KKW18, Beu20, AGH+23] , significantly reducing this generic communication cost and in turn decreasing the corresponding signature sizes.

One notable such improvement is the MPCitH paradigm [IKOS07], in which a prover simulates all N parties of an MPC protocol (in his head). This MPC protocol is carefully designed so that if all parties accept, it proves that the initial input shares of the parties form a witness for a valid solution to the underlying problem. Following the simulation, the verifier challenges the prover to disclose the initial states of all parties except one (chosen by the verifier), enabling the verification of the correct simulation for $N-1$ out of N parties. This generally leads to a ZKPoK with a cheating probability, or soundness, of $1/N$.

The introduction of MPCitH has initiated a whole line of research on constructions following this paradigm [FJR22, BG23, FJR23, ARZV23, Wan22, FMRV22, Fen22, CNP+22, BBP+23, GPS22, BGKM23]. Subsequently, numerous further protocol-level improvements have been proposed [KKW18, KZ22, BG23, AGH+23], some of which compatible with the

MPCitH idea we have incorporated into the construction of MiRitH.

The MinRank Problem and Related Constructions. MiRitH relies on the hardness of the MinRank problem. This problem was introduced in [BFS99] and has been extensively studied due to its many applications in cryptology, especially in cryptanalysis [KS99, GC00, BG06, BFP11, CSV17, Beu21, Beu22, TPD21]. However, recently the problem became more popular as foundation for signature schemes constructed via the Fiat–Shamir transform. Initiated by Courtois [Cou01] who proposed the first signature scheme relying on the hardness of the MinRank problem and followed by an extension due to Bellini, Esser, Sanna and Verbel [BESV22], called MR-DSS, leveraging the *sigma protocol with helper paradigm* due to Beullens [Beu20]. The most recent and efficient constructions are [ARZV23] shortly followed by [Fen22] which both rely on the MPCitH paradigm. Feneuil [Fen22] proposes two constructions. The first relies on an MPC protocol similar to [ARZV23], which verifies a matrix product relation of the form $A \cdot B = C$ that proves knowledge of the MinRank solution. While the protocol in [Fen22] is a slightly more efficient special case of [ARZV23], the latter uses a more efficient relation based on the Kipnis-Shamir modeling which involves smaller matrices A, B, C. MiRitH incorporates the most efficient techniques of the two schemes. The second scheme proposed in [Fen22] uses an MPC protocol for verifying roots of polynomials in $\mathbb{F}_q$ extensions, and is mostly unrelated to [ARZV23] and MiRitH.

Our Contribution We present the new signature scheme MiRitH. MiRitH obtains competitive signatures with 5.7 kB for NIST category I security. In comparison the schemes from [ARZV23] and [Fen22] obtain 7.4 kB and 7.2 kB signatures respectively, while the scheme based on q-polynomials given in [Fen22] is roughly on par with 5.5 kB signatures. Note that the q-polynomials version of [Fen22] is the basis for the signature scheme MIRA [ABB+23d] which is also currently submitted to the NIST process. MIRA also obtains signatures of roughly the same size with 5.6 kB for NIST category I security. MiRitH is therefore in competition for the smallest signatures for any construction based on the MPCitH paradigm.

Furthermore, our optimized constant-time implementations show that MiRitH offers greatly competitive performance. At 5.7 kB signature size, signing and verification can be performed in roughly 30 MCycles using our AVX2 implementation, which improves on MIRA's 40 MCycles by about 25%. Our NEON implementation offers even better performance allowing for signing and verification in about 20 MCycles.

Our benchmarks on a single benchmarking platform reveal that the true speed advantage of MiRitH over MIRA is almost 50% for short signatures (around 5.7 kB) and about a factor of x10 for the faster instantiations of both schemes. The speed advantage for fast instantiations stems from the fact that MiRitH offers great size-performance trade-offs. An increase of the signature to 7.9 kB allows for a speedup of a factor of roughly x6. More precisely, signing and verification in that case can be performed at 5 MCycles using our AVX2 implementation and 3 MCycles using our NEON implementation.

In order to obtain those improvements and to build a solid foundation for MiRitH, we provide improvements on multiple levels of the design process. In the following we categorize the different contributions.

Design Related Improvements First, we improve the core of the construction, namely the MPC protocol used to prove the matrix product relation, using two improvements found in the literature. The first originates from Kales and Zaverucha [KZ22, Section 2.5] and also found application in [Fen22]. This improvement is related to the aforementioned more efficient special case. Second, we apply a technique by Feneuil [Fen22] to reduce

a challenge matrix sent by the verifier at the expense of increasing the soundness error originating from guessing that challenge.

Additionally, in [ARZV23] the challenge matrix has to be sampled from an exceptional set, which is not necessary for MiRitH, improving performance. We provide the updated security statements and outline differences to the proofs from [ARZV23] whenever necessary. For the self-contained full proofs we refer to the corresponding NIST-submitted specification document of MiRitH [ABB+23b]. Additionally, we translate to MiRitH the recent MPCitH-hypercube technique by Aguilar-Melchor et al. [AGH+23], introduced in the context of a ZKPoK proving knowledge of a solution to the syndrome decoding problem. This technique allows to achieve the same soundness as an MPC protocol with N^D parties by only simulating D MPC protocols with N parties each. As still input shares for N^D parties have to be prepared, the input preparation starts dominating the computation time rather quickly. However, we show that in our setting, the hypercube technique speeds up signing and verification by more than 2.5 times.

Parameters and Security Justification We provide re-optimized parameters for the NIST security categories I, III and V, showing that some of the previous suggestions from [ARZV23] were slightly suboptimal.

Additionally, we provide the missing quantum security justification for all parameter sets. Therefore we construct a novel quantum search enhanced algorithm that outperforms a naive Grover improved brute-force for the MinRank solution. Our algorithm exploits the Kipnis-Shamir modeling, which models knowledge of a MinRank solution as a matrix-vector product involving a secret matrix K. We show that a subset of columns of K relates to a matrix with a rank defect which can be used to distinguish those columns from random ones in the search process, allowing to construct an oracle. Furthermore we show at the same time that knowledge of these few columns of K is already sufficient to recover the MinRank solution. Subsequently, we show that under NIST quantum security metrics, which restrict the maximum depth of the used quantum circuit, all parameter sets have a positive security margin, i.e., they exceed the defined quantum security thresholds.

Implementations and Benchmarks We provide extensive implementations for MiRitH in the form of a first reference implementation as well as optimized constant-time implementations for Intel processors (AVX2) and the ARM architecture (NEON). For the ARM architecture we provide an implementation for high-performance devices such as the Mac M1 as well as for low-power units such as the Cortex-M4. We provide all these implementations with and without the hypercube improvement from [AGH+23].

In the context of our optimized implementation we provide a new improved matrix arithmetic leveraging AVX2 instructions that outperform previous implementations from [BCH+23]. This is achieved in two ways: first we replace the needed lookup table of [BCH+23], with an efficient use of the `vpblendvb` instruction. Second, as the matrices in MiRitH are generally small we load the full matrices directly into the necessary registers, rather than having to move rows or columns into memory during computation.

Further we provide a comparison of MiRitH to other NIST submissions following the MPCitH / Fiat-Shamir paradigm. In this comparison we compare public key and signature sizes as well as key generation, signing and verification speed (compare to Table 6). In this context we re-benchmark all considered schemes on a single benchmarking platform to enable a fair comparison.

Outline. In Section 2 we cover general notations, introduce the MinRank problem and recall the ARZV scheme. In Section 3 we then introduce MiRitH in full detail. Subsequently in Section 4 we provide updated parameters for MiRitH for the different NIST security categories. In Section 4.2 we describe our new quantum algorithm and analyze quantum

security of the suggested parameter sets. Eventually, in Section 5 we give insights on our optimized constant-time implementations and comparison to other NIST candidates.

2 Preliminaries

2.1 General Notation

Let $\mathbb{F}_q$ be a finite field of q elements, let $\mathbb{F}_q^{m\times n}$ be the vector space of $m \times n$ matrices over $\mathbb{F}_q$, and let $\mathsf{Rank}(M)$ be the rank of the matrix M. We define $[i] := \{1, 2, \ldots, i\}$ for every positive integer i. Further, let log be the logarithm in base 2. We write $a \leftarrow \mathcal{A}(x)$ if a is the output of an algorithm $\mathcal{A}$ on input x and $a \xleftarrow{\$} \mathcal{S}$ if a is sampled uniformly at random from the set $\mathcal{S}$. We use $a \leftarrow \mathrm{PRG}(\mathsf{seed})$ if a is generated by the pseudorandom generator PRG using the seed seed. In the following Hash denotes a cryptographic-secure hash function.

Additive Sharing We recall that an additive sharing of an element a is a tuple $[\![a]\!] := ([\![a]\!]_1, \ldots, [\![a]\!]_N)$ such that $a = \sum_{i=1}^{N} [\![a]\!]_i$. In the MPC context, an additive sharing $[\![a]\!]$ is distributed between N parties, i.e., each party obtains one of the N shares $[\![a]\!]_i$. The linearity of the sum allows the parties to compute certain operations such as the summation of elements on individual shares, while maintaining a valid additive share for the result of that operation. Such operations include the sum of elements, since $[\![x + y]\!] = [\![x]\!] + [\![y]\!]$, and the multiplication by a constant c, as $[\![c \cdot x]\!] = c \cdot [\![x]\!]$. Further the addition of a constant can also be modeled since $[\![c + x]\!] = ([\![x]\!]_1, \ldots, [\![x]\!]_{N-1}, c + [\![x]\!]_N)$, i.e., only one party performs the addition.

2.2 MinRank Problem and Kipnis–Shamir Modeling

The MinRank problem is the underlying hard problem of MiRitH.

Problem 1 (MinRank)**.** *Let q,m,n,k, and r be positive integers, with q a prime power. The MinRank problem with parameters (q, m, n, k, r) is defined as:*

Given: *$(k+1)$-tuple $\boldsymbol{M} = (M_0, M_1, \ldots, M_k) \in (\mathbb{F}_q^{m\times n})^{k+1}$.*

Find: *$\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k) \in \mathbb{F}_q^k$ such that $\mathsf{Rank}\big(M_0 + \sum_{i=1}^{k} \alpha_i M_i\big) \leq r$.*

To prove the knowledge of a MinRank solution we use the Kipnis–Shamir modeling [KS99], which is based on the following fact. Given an instance $\boldsymbol{M}$ of the MinRank problem, if there exists a vector $\boldsymbol{\alpha} \in \mathbb{F}_q^k$ and a matrix $K \in \mathbb{F}_q^{r\times(n-r)}$ such that

$$\left(M_0 + \sum_{i=1}^{k} \alpha_i M_i\right) \cdot \begin{bmatrix} I \\ -K \end{bmatrix} = \mathbf{0}, \tag{1}$$

where $I \in \mathbb{F}_q^{(n-r)\times(n-r)}$ is a non-singular matrix, then $\boldsymbol{\alpha}$ is a solution to the instance $\boldsymbol{M}$.

If in Eq. (1) we fix I to be the identity matrix of size $(n-r) \times (n-r)$, then we obtain

$$M_0^L + \sum_{i=1}^{k} \alpha_i M_i^L = \left(M_0^R + \sum_{i=1}^{k} \alpha_i M_i^R\right) \cdot K, \tag{2}$$

where for any matrix $A \in \mathbb{F}_q^{m\times n}$, we let A^L, respectively A^R, be the matrix consisting of the first $n - r$ columns, respectively the last r columns, of A. Further, this allows to rewrite Eq. (2) as $\boldsymbol{M}_{\boldsymbol{\alpha}}^L = \boldsymbol{M}_{\boldsymbol{\alpha}}^R \cdot K$, with $\boldsymbol{M}_{\boldsymbol{\alpha}} := M_0 + \sum_{i=1}^{k} \alpha_i M_i$.

2.3 The ARZV Signature Scheme

In its core the ARZV scheme relies on an MPC protocol Π to verify matrix-multiplication triples in zero-knowledge, which was introduced together with the scheme in [ARZV23]. A matrix-multiplication triple is a set (X, Y, Z) of matrices fulfilling $X \cdot Y = Z$. This protocol Π is then used to verify the triple $(\boldsymbol{M}_{\boldsymbol{\alpha}}^L, \boldsymbol{M}_{\boldsymbol{\alpha}}^R, K)$, i.e., to show that this triple satisfies the Kipnis-Shamir equation from Eq. (2), which in turn proves that $\boldsymbol{\alpha}$ is a solution to the MinRank problem defined on the matrices $\boldsymbol{M}$.

In order to verify a matrix-multiplication triple, the protocol requires as input an auxiliary matrix-multiplication triple (A, B, C) (shared among the parties). This auxiliary triple is used in the verification process, but no information on it is revealed.

From there the scheme is constructed using the general MPCitH framework to first construct a ZKPoK in which the prover simulates all MPC parties and then relies on the Fiat-Shamir transform to obtain a EUF-CMA secure signature scheme in the random oracle model.

3 MiRitH Signature Scheme

In this section, we present a new MPCitH-based signature scheme called MiRitH. We first describe its underlying MPC protocol, denoted by Π_s, and its corresponding proof of knowledge protocol. Afterwards, we outline the signature scheme which is obtained by employing the Fiat-Shamir transformation.

3.1 MPC Protocol

MiRitH uses an MPC protocol with additive shares between N parties to verify the validity of a MinRank solution via the Kipnis-Shamir modeling, i.e., the protocol verifies that $\boldsymbol{M}_{\boldsymbol{\alpha}}^L = \boldsymbol{M}_{\boldsymbol{\alpha}}^R \cdot K$ (compare to Section 2.2). Therefore, each party i holds an additive share $[\![\boldsymbol{\alpha}]\!]_i$ of the MinRank solution $\boldsymbol{\alpha}$, such that $[\![\boldsymbol{\alpha}]\!] = \sum_{i=1}^N [\![\boldsymbol{\alpha}]\!]_i$, and an additive share $[\![K]\!]_i$ of the matrix K, such that $[\![K]\!] = \sum_{i=1}^N [\![K]\!]_i$. Based on its shares, each party can construct its own share of $\boldsymbol{M}_{\boldsymbol{\alpha}}^L$ and $\boldsymbol{M}_{\boldsymbol{\alpha}}^R$.

Once each party obtained its input shares the protocol Π_s is executed. Similarly to [ARZV23], MiRitH uses a MPC protocol to prove that the shares of $(\boldsymbol{M}_{\boldsymbol{\alpha}}^L, \boldsymbol{M}_{\boldsymbol{\alpha}}^R, K)$ of all parties form a valid matrix-multiplication triple, which in turn proves knowledge of the MinRank solution. However, the MPC protocol employed by MiRitH, which we call Π_s, improves upon the protocol introduced in [ARZV23] in several ways. In Fig. 1, we outline the full protocol Π_s that verifies the validity of the solution $\boldsymbol{\alpha}$ with a false-positive rate of $1/q^s$ in the semi-honest setting, i.e., when all the parties follow the protocol. The improvements of Π_s over the MPC protocol of [ARZV23] are the following.

First, the used auxiliary matrix-multiplication triple (A, B, C) is chosen as (A, K, C) with A being chosen randomly, rather than A and B being chosen randomly. Besides involving one matrix less this leads to further cancellation effects in the subsequent computations, which results in increased efficiency.

Second, the random matrix R has no special structure, while the protocol in [ARZV23] requires to select R from a special set of *exceptional matrices*, which is computationally more expensive. Third, employing an optimization by Feneuil [Fen22], the random matrix R is chosen to have size $s \times m$, with $s < m$. This consequently reduces the size of the matrices S, C and K in Fig. 1 which in turn reduces the communication cost and ultimately the signature size.

Inputs: Each party knows the MinRank instance $\boldsymbol{M}$ and takes a share of the following sharings as input: $[\![\boldsymbol{\alpha}]\!]$ and $[\![K]\!]$, where $\boldsymbol{\alpha} \in \mathbb{F}_q^k$ and $K \in \mathbb{F}_q^{r\times(n-r)}$; $[\![A]\!]$, where A has been uniformly sampled from $\mathbb{F}_q^{s\times r}$; and $[\![C]\!]$, where $C = A \cdot K$.

MPC Protocol:

1 : The parties locally compute $[\![\boldsymbol{M}_{\boldsymbol{\alpha}}^L]\!]$ and $[\![\boldsymbol{M}_{\boldsymbol{\alpha}}^R]\!]$.

2 : The parties get a random matrix $R \xleftarrow{\$} \mathbb{F}_q^{s\times m}$.

3 : The parties locally set $[\![S]\!] = R \cdot [\![\boldsymbol{M}_{\boldsymbol{\alpha}}^R]\!] + [\![A]\!]$.

4 : The parties open $[\![S]\!]$ so that they all obtain S.

5 : The parties locally compute $[\![V]\!] = S \cdot [\![K]\!] - R \cdot [\![\boldsymbol{M}_{\boldsymbol{\alpha}}^L]\!] - [\![C]\!]$.

6 : The parties open $[\![V]\!]$ to obtain V.

7 : The parties output **accept** if $V = \mathbf{0}$ and **reject** otherwise.

Figure 1: The MPC protocol Π_s to check that $\boldsymbol{M}_{\boldsymbol{\alpha}}^L = \boldsymbol{M}_{\boldsymbol{\alpha}}^R \cdot K$.

Proposition 1 states the correctness the protocol Π_s (Fig. 1) and its false-positive rate.

Proposition 1. *If $\boldsymbol{M}_{\boldsymbol{\alpha}}^L = \boldsymbol{M}_{\boldsymbol{\alpha}}^R \cdot K$ then the protocol Π_s in Fig. 1 always outputs **accept**. If $\boldsymbol{M}_{\boldsymbol{\alpha}}^L \neq \boldsymbol{M}_{\boldsymbol{\alpha}}^R \cdot K$ then Π_s outputs **accept** with probability at most $1/q^s$.*

Proof (sketch). Correctness easily follows. If $\Delta := \boldsymbol{M}_{\boldsymbol{\alpha}}^L - \boldsymbol{M}_{\boldsymbol{\alpha}}^R \cdot K \neq \mathbf{0}$ then Π_s accepts if and only if $R \cdot \Delta = \mathbf{0}$ and, by the Rouché–Capelli theorem, the number of such R's is at most $q^{(m-1)s}$, thus the claim follows. □

3.2 ZKPoK for MinRank

Starting from the MPC protocol Π_s, we build a ZKPoK of a solution to an instance of the MinRank problem via a 5-pass protocol. A pseudocode of the ZKPoK is shown in Fig. 2. It employs a commitment scheme Com, a pseudorandom generator PRG, and a seed tree TreePRG. The phases of the protocol can be interpreted as follows

1) The prover prepares all inputs for the different parties of the MPC protocol and commits to those initial states.

2) The verifier provides the first challenge $R \in \mathbb{F}_q^{s\times m}$.

3) The prover executes the MPC protocol Π_s for each party based on the challenge R and commits to the final views of all parties.

4) The verifier provides the second challenge $i^* \in [N]$.

5) The prover opens all commitments corresponding to parties $i \neq i^*$ and provides the final view of party i^*, to allow the verifier to recompute all parties's views and commitments of the protocol.

It can be proven that the protocol described in Fig. 2 is correct, honest-verifier ZKPoK, and sound with soundness error

$$\varepsilon := \frac{1}{q^s} + \left(1 - \frac{1}{q^s}\right)\frac{1}{N}.$$

Proofs for these statements follow mostly along the lines of the proofs provided in [ARZV23]. Only the change to a different first challenge matrix R leads to minor necessary adaptations.

$\mathsf{ZKProof}(\mathsf{Prover}(\boldsymbol{M}, \boldsymbol{\alpha}, K), \mathsf{Verifier}(\boldsymbol{M}))$

Phase 1: Prover sets up the inputs for the MPC protocol Π_s:

1 : $\mathsf{seed} \xleftarrow{\$} \{0,1\}^\lambda, \quad (\mathsf{seed}_i, \rho_i)_{i\in[N]} \leftarrow \mathrm{TreePRG}(\mathsf{seed})$

2 : For each party $i \in [N-1]$

$[\![\boldsymbol{\alpha}]\!]_i, [\![K]\!]_i, [\![A]\!]_i, [\![C]\!]_i \leftarrow \mathrm{PRG}(\mathsf{seed}_i)$

$\mathsf{state}_i \leftarrow \mathsf{seed}_i$

3 : $[\![\boldsymbol{\alpha}]\!]_N \leftarrow \boldsymbol{\alpha} - \sum_{i\neq N} [\![\boldsymbol{\alpha}]\!]_i, \quad [\![K]\!]_N \leftarrow K - \sum_{i\neq N} [\![K]\!]_i$

4 : $[\![A]\!]_N \leftarrow \mathrm{PRG}(\mathsf{seed}_N), \quad [\![C]\!]_N \leftarrow A \cdot K - \sum_{i\neq N} [\![C]\!]_i$

5 : $\mathsf{aux} \leftarrow ([\![\boldsymbol{\alpha}]\!]_N, [\![K]\!]_N, [\![C]\!]_N), \quad \mathsf{state}_N \leftarrow (\mathsf{seed}_N, \mathsf{aux})$

6 : Commit to each party's state: $\mathsf{com}_i \leftarrow \mathsf{Com}\big(\mathsf{state}_i, \rho_i\big)$, for all $i \in [N]$.

7 : Prover computes $h_1 \leftarrow \mathrm{Hash}(\mathsf{com}_1, \ldots, \mathsf{com}_N)$ and sends it to Verifier.

Phase 2: Verifier samples $R \xleftarrow{\$} \mathbb{F}_q^{s\times m}$ and sends it to Prover.

Phase 3: Prover simulates the MPC protocol Π_s:

8 : The parties locally compute $[\![\boldsymbol{M}^L_{\boldsymbol{\alpha}}]\!]$ and $[\![\boldsymbol{M}^R_{\boldsymbol{\alpha}}]\!]$.

9 : The parties locally set $[\![S]\!] = R \cdot [\![\boldsymbol{M}^R_{\boldsymbol{\alpha}}]\!] + [\![A]\!]$.

10 : The parties open $[\![S]\!]$ so that they all obtain S.

11 : The parties locally compute $[\![V]\!] = S \cdot [\![K]\!] - R \cdot [\![\boldsymbol{M}^L_{\boldsymbol{\alpha}}]\!] - [\![C]\!]$.

12 : Prover computes $h_2 \leftarrow \mathrm{Hash}\big([\![S]\!]_1, [\![V]\!]_1, \ldots, [\![S]\!]_N, [\![V]\!]_N\big)$.

13 : Prover sends h_2 to Verifier.

Phase 4: Verifier samples $i^* \xleftarrow{\$} [N]$ and sends it to Prover.

Phase 5: Prover sends $\mathsf{rsp} := \big((\mathsf{state}_i, \rho_i)_{i\neq i^*}, \mathsf{com}_{i^*}, [\![S]\!]_{i^*}\big)$ to Verifier.

Verification:

14 : Verifier recompute $([\![S]\!]_i, [\![V]\!]_i)_{i\neq i^*}$ from $(\mathsf{state}_i)_{i\neq i^*}$.

15 : $[\![V]\!]_{i^*} \leftarrow -\sum_{i\neq i^*} [\![V]\!]_i$

16 : Verifier accepts if and only if $\mathsf{com}_i = \mathsf{Com}(\mathsf{state}_i, \rho_i)$, for each $i \neq i^*$, $h_1 = \mathrm{Hash}(\mathsf{com}_1, \ldots, \mathsf{com}_N), h_2 = \mathrm{Hash}\big([\![S]\!]_1, [\![V]\!]_1, \ldots, [\![S]\!]_N, [\![V]\!]_N\big)$.

Figure 2: Zero-knowledge proof of knowledge for MinRank.

3.3 Signature Scheme via the Fiat–Shamir Transform

To transform the ZKPoK of Fig. 2 into a non-interactive signature scheme, which we call MiRitH, we use a standard generalization of the Fiat–Shamir transform for canonical 5-pass protocols [FS87]. We provide the corresponding algorithms for signing and verification in Fig. 4 and Fig. 5.

Theorem 1 shows that MiRitH is existentially unforgeable under adaptive chosen-message attacks in the random oracle model.

Theorem 1 (Unforgeability)**.** *Suppose that* PRG *is* $(t, \varepsilon_{\mathrm{PRG}})$*-secure and that any adversary running in time* t *has at most an advantage* $\varepsilon_{\mathrm{MR}}$ *against the underlying MinRank problem associated with a public key* $\boldsymbol{M}$*. Moreover, assume that* Hash_0*,* Hash_1*, and* Hash_2 *are modeled as random oracles. Let* $\mathcal{A}$ *be an adaptive chosen-message adversary against the signature scheme described in Fig. 4, running in time* t*, making* q_s *signing queries, and* q_i *queries to* Hash_i *for* $i = 0, 1, 2$*. Then* $\mathcal{A}$ *succeeds in outputting a valid forgery with*

probability

$$\Pr[\mathsf{Forge}] \leq \frac{3(q_0+\tau N q_s)^2}{2^{2\lambda+1}} + \frac{q_s(q_s+q_0+q_1+q_2)}{2^{2\lambda}} + q_s\tau\varepsilon_{\mathrm{PRG}} + \varepsilon_{\mathrm{MR}} + \max_{0\leq t\leq\tau} P(t),$$

where $P(t) = \left(1-\left[1-\left(\frac{1}{q^s}\right)^t\left(1-\frac{1}{q^s}\right)^{\tau-t}\binom{\tau}{t}\right]^{q_1}\right)\left(1-\left[1-\frac{1}{N^{\tau-t}}\right]^{q_2}\right).$

Proof. This theorem is essentially obtained from Theorem 4 of [ARZV23] by considering the probability $\frac{1}{q^s}$ instead of $\frac{1}{q^n}$ due to the corresponding change to the ZKPoK in Section 3.2. □

Key Generation The uncompressed public and secret key of our scheme are a random instance $\boldsymbol{M}$ of the MinRank problem and the corresponding witness $(\boldsymbol{\alpha}, K)$, respectively. If stored directly, this would require $mn(k+1)\log q$ bits for the public and $\big(k+r(n-r)\big)\log q$ bits for the secret key.

KeyGen()

1 : $(\mathsf{seed}_{\mathsf{pk}}, \mathsf{seed}_{\mathsf{sk}}) \xleftarrow{\$} \{0,1\}^\lambda \times \{0,1\}^\lambda$
2 : $M_1, \ldots, M_k \leftarrow \mathsf{PRG}(\mathsf{seed}_{\mathsf{pk}})$
3 : $\boldsymbol{\alpha}, K, E^R \leftarrow \mathsf{PRG}(\mathsf{seed}_{\mathsf{sk}})$
4 : $E \leftarrow E^R \cdot K \mid E^R$
5 : $M_0 \leftarrow E - \sum_{i=1}^{k} \alpha_i M_i$
6 : $\mathsf{pk} \leftarrow (\mathsf{seed}_{\mathsf{pk}}, M_0)$
7 : $\mathsf{sk} \leftarrow \mathsf{seed}_{\mathsf{sk}}$
return $(\mathsf{pk}, \mathsf{sk})$

DecompressPK(pk)

1 : $\mathsf{seed}_{\mathsf{pk}}, M_0 \leftarrow \mathsf{pk}$
2 : $M_1, \ldots, M_k \leftarrow \mathsf{PRG}(\mathsf{seed}_{\mathsf{pk}})$
return $(M_0, \ldots, M_k)$

DecompressSK(sk)

1 : $\mathsf{seed}_{\mathsf{sk}} \leftarrow \mathsf{sk}$
2 : $\boldsymbol{\alpha}, K, E^R \leftarrow \mathsf{PRG}(\mathsf{seed}_{\mathsf{sk}})$
return $(\boldsymbol{\alpha}, K)$

Figure 3: Algorithms for generating, compressing, and decompressing the keys.

We employ the original key generation scheme which compresses the public key into $\lambda + mn\log q$ bits and the secret key into λ bits. This key generation scheme is detailed in Fig. 3.

We remark that using the key generation scheme proposed by Di Scala and Sanna [DSS23] would allow to compress the public key further to $\lambda + (m(n-r)-k)\log q$ bits, at the cost of computation time.

3.4 Hypercube Variant

Recently, Aguilar-Melchor et al. [AGH+23] introduced a general technique, called *hypercube*, that makes possible to reduce from N^D to only ND the number of parties that an MPCitH based on a linear secret sharing needs to simulate. They applied the hypercube to a MPCitH for the syndrome decoding problem to produce a code-based digital signature. In MiRitH, we applied the hypercube technique to our MPCitH of MinRank. In the following, we briefly describe how the technique is applied. Overall, the hypercube application results in further computational speedups.

Consider the original ZKPoK from Fig. 2 with N^D parties (instead of N). In this ZKPoK protocol, the shares for the N^D parties, which are called *leaf parties* in the following, are set up and committed as usual. So, the hypercube variant uses the same initialization.

$\mathsf{Sign}(\boldsymbol{M}, \boldsymbol{\alpha}, K, \mathsf{msg})$

$\mathsf{salt} \xleftarrow{\$} \{0,1\}^{2\lambda}$

Phase 1: Set up the views for the MPC protocols

for $\ell \in [\tau]$ **do**

$\mathsf{seed}^{(\ell)} \xleftarrow{\$} \{0,1\}^{\lambda}, \quad (\mathsf{seed}_i^{(\ell)})_{i\in[N]} \leftarrow \mathrm{TreePRG}(\mathsf{salt}, \mathsf{seed}^{(\ell)})$

for $i \in [N-1]$ **do**

$[\![A^{(\ell)}]\!]_i, [\![\boldsymbol{\alpha}^{(\ell)}]\!]_i, [\![C^{(\ell)}]\!]_i, [\![K^{(\ell)}]\!]_i \leftarrow \mathrm{PRG}(\mathsf{salt}, \mathsf{seed}_i^{(\ell)})$

$\mathsf{state}_i^{(\ell)} \leftarrow \mathsf{seed}_i^{(\ell)}$

$[\![A^{(\ell)}]\!]_N \leftarrow \mathrm{PRG}(\mathsf{salt}, \mathsf{seed}_N^{(\ell)}), \quad [\![\boldsymbol{\alpha}^{(\ell)}]\!]_N \leftarrow \boldsymbol{\alpha} - \sum_{i\neq N} [\![\boldsymbol{\alpha}^{(\ell)}]\!]_i$

$[\![K^{(\ell)}]\!]_N \leftarrow K - \sum_{i\neq N} [\![K^{(\ell)}]\!]_i, \quad [\![C^{(\ell)}]\!]_N \leftarrow A^{(\ell)} \cdot K - \sum_{i\neq N} [\![C^{(\ell)}]\!]_i$

$\mathsf{aux}^{(\ell)} \leftarrow ([\![\boldsymbol{\alpha}^{(\ell)}]\!]_N, [\![K^{(\ell)}]\!]_N, [\![C^{(\ell)}]\!]_N), \quad \mathsf{state}_N^{(\ell)} \leftarrow (\mathsf{seed}_N^{(\ell)}, \mathsf{aux}^{(\ell)})$

$\mathsf{com}_i^{(\ell)} \leftarrow \mathrm{Hash}\left(\mathsf{salt}, \ell, i, \mathsf{state}_i^{(\ell)}\right),$ for all $i \in [N]$

Phase 2: First challenges

$h_1 \leftarrow \mathrm{Hash}\left(\mathsf{msg}, \mathsf{salt}, (\mathsf{com}_i^{(\ell)})_{i\in[N],\ell\in[\tau]}\right)$

$R^{(1)}, \ldots, R^{(\tau)} \leftarrow \mathrm{PRG}(h_1)$

Phase 3: Simulation of the MPC protocols

for $\ell \in [\tau]$ **do**

Compute $[\![\boldsymbol{M}_{\boldsymbol{\alpha}}^{L,(\ell)}]\!], [\![\boldsymbol{M}_{\boldsymbol{\alpha}}^{R,(\ell)}]\!]$ from $[\![\boldsymbol{\alpha}^{(\ell)}]\!]$

$[\![S^{(\ell)}]\!] \leftarrow R^{(\ell)} \cdot [\![\boldsymbol{M}_{\boldsymbol{\alpha}}^{R,(\ell)}]\!] + [\![A^{(\ell)}]\!]$

$S^{(\ell)} \leftarrow \sum_i [\![S^{(\ell)}]\!]_i$

$[\![V^{(\ell)}]\!] \leftarrow S^{(\ell)} \cdot [\![K^{(\ell)}]\!] - R^{(\ell)} \cdot [\![\boldsymbol{M}_{\boldsymbol{\alpha}}^{L,(\ell)}]\!] - [\![C^{(\ell)}]\!]$

Phase 4: Second challenges

$h_2 \leftarrow \mathrm{Hash}\left(\mathsf{msg}, \mathsf{salt}, h_1, \left([\![S^{(\ell)}]\!]_i, [\![V^{(\ell)}]\!]_i\right)_{i\in[N],\ \ell\in[\tau]}\right)$

$i^{*,(1)}, \ldots, i^{*,(\tau)} \leftarrow \mathrm{PRG}(h_2)$

Phase 5: Assembling the signature σ

$\sigma \leftarrow \left(\mathsf{salt}, h_1, h_2, \left(\left(\mathsf{state}_i^{(\ell)}\right)_{i\neq i^{*,(\ell)}}, \mathsf{com}_{i^{*,(\ell)}}^{(\ell)}, [\![S^{(\ell)}]\!]_{i^{*,(\ell)}}\right)_{\ell\in[\tau]}\right)$

return σ

Figure 4: Signing algorithm of MiRitH using τ parallel executions of the ZKPoK.

However, instead of running an MPC protocol on N^D parties, the hypercube variant expresses each leaf party $i \in [N^D]$ as coordinates in a D dimensional hypercube with sides of length N. Hence, each leaf party i is uniquely identified via a set of coordinates $(i_1, \ldots, i_D) \in [N]^D$. Next, D sets of N *main parties* each are constructed from the leaf parties. On each of those sets follows an execution of the MPC protocol with N parties. The share $[\![X]\!]_{(k,j)}$ of a value X, $k \in [D]$, $j \in [N]$, for the j-th main party in the k-th set, is defined as the sum of all leaf party shares that have coordinates lying on the j-th hyperplane orthogonal to the k-th dimension. Put differently, the share $[\![X]\!]_{(k,j)}$ is the sum of the shares $[\![X]\!]_i$ held by the leaf parties $i = (i_1, \ldots, i_D)$ having $i_k = j$. Straightforward computations show that by this construction method each of the values from the original protocol, i.e., $\boldsymbol{\alpha}$, K, A and C, is recovered by summing all corresponding shares of the N

$\mathsf{Verif}(\boldsymbol{M}, \mathsf{msg}, \sigma)$

$R^{[1]}, \ldots, R^{(\tau)} \leftarrow \mathrm{PRG}(h_1)$

$i^{*,[1]}, \ldots, i^{*,(\tau)} \leftarrow \mathrm{PRG}(h_2)$

for all $\ell \in [\tau]$ **do**

  Compute $[\![V^{(\ell)}]\!]_i$ as in Sign

  Compute $\mathsf{com}_i^{(\ell)}$ and $[\![S^{(\ell)}]\!]_i$ as in Sign, for all $i \in [N] \backslash \{i^{*,(\ell)}\}$

  $S^{(\ell)} \leftarrow \sum_i [\![S^{(\ell)}]\!]_i$

  Compute $[\![V^{(\ell)}]\!]_i$ as in Sign, for all $i \in [N] \backslash \{i^{*,(\ell)}\}$

  $[\![V^{(\ell)}]\!]_{i^{*,(\ell)}} \leftarrow -\sum_{i \neq i^{*,(\ell)}} [\![V^{(\ell)}]\!]_i$

$h_1' \leftarrow \mathrm{Hash}\big(\mathsf{msg}, \mathsf{salt}, (\mathsf{com}_i^{(\ell)})_{i \in [N], \ell \in [\tau]}\big)$

$h_2' \leftarrow \mathrm{Hash}\big(\mathsf{msg}, \mathsf{salt}, h_1, \big([\![S^{(\ell)}]\!]_i, [\![V^{(\ell)}]\!]_i\big)_{i \in [N],\ \ell \in [\tau]}\big)$

Output **accept** if $h_1' = h_1$ and $h_2' = h_2$, otherwise output **reject**

Figure 5: Verification algorithm of MiRitH.

main parties in one set.

All D MPC protocols are executed in parallel on the same initial challenge R. The rest of the protocol proceeds as usual. That is, the verifier provides a second challenge value $i^* \in [N]^D$ corresponding to one of the leaf parties. The prover then reveals all initial states of the leaf parties except for leaf party i^*, for which it provides the corresponding communication. The reason for the resulting protocol offering the same soundness as an MPC protocol with N^D parties lies in the specific crafting of the input shares, namely, disclosing all but one of the leaf party shares discloses the value of all but one main party share per set. For the full details we refer to [AGH+23].

The communication complexity of the hypercube variant is equivalent to the non-hypercube variant with N^D parties. However, the hypercube approach still offers a two-fold advantage. First, the computational complexity is improved by having to simulate exponentially fewer parties ($N \cdot D$ instead of N^D). Second, even though N^D initial states have to be prepared for the leaf parties, which usually becomes the dominating part after applying the technique, all but one of those states can be set up offline, giving rise to interesting online-offline computation ratios.

4 Parameters

In Table 2, we propose parameter sets for MiRitH achieving the different security levels I, III, and V defined by NIST, which correspond to 143, 207, and 272 bits of classical security, respectively [NIS].

The MinRank problem parameters (q, m, n, k, r) are chosen with respect to the best-known classical attacks against random instances. These attacks follow the hybrid approach introduced in [BBB+23c], which requires to guess $a < \lceil \frac{k}{m} \rceil$ vectors in the kernel of the unknown low-rank matrix E and ℓ coefficients of the solution vector $\boldsymbol{\alpha}$. For each guess, one solves a smaller P instance with parameters $(q, m, n-a, k-ma-\ell, r)$. This approach has a complexity of $q^{a \cdot r + l_v}(\mathsf{MR}(P) + k \cdot (am)^2)$ multiplications over $\mathbb{F}_q$, where $\mathsf{MR}(P)$ and $(am)^2$ are the costs of solving and building the instance P, respectively. The instance P is solved by using either the Kernel-Search (KS) [GC00], the Support-Minors (SM) [BBC+20] or the Big-k algorithm [Cou01].

Table 1: Bit complexities of the MinRank instances of MiRitH against known attacks.

Set	q	m	n	k	r	KS (a, ℓ)	Big-k (a, ℓ)	SM (a, ℓ, b, n')
Ia	16	15	15	78	6	148 (4,3)	155 (5,3)	**144** (5,0,1,8)
Ib	16	16	16	142	4	**159** (8,0)	231 (0,0)	166 (8,0,2,8)
IIIa	16	19	19	109	8	**207** (5,0)	431 (0,0)	210 (5,0,2,14)
IIIb	16	19	19	167	6	**232** (8,0)	352 (0,0)	236 (8,0,2,11)
Va	16	21	21	189	7	**269** (8,0)	452 (0,0)	274 (6,0,8,15)
Vb	16	22	22	254	6	303 (11,0)	425 (0,0)	**301**(11,0,1,11)

Table 1 shows the bit complexities to solve the MinRank instances of MiRitH, where the cost of one multiplication in $\mathbb{F}_q$ is taken as $\log_2(q)^2$ bit operations, the exponent for matrix multiplication in KS and Big-k is set to be 3, and SM is taken in its Block-Wiedemann variant. Further, in Section 4.2 we provide a justification for parameter security in a quantum setting under NIST security metrics.

Table 2: Parameters of MiRitH with corresponding public key and signatures sizes in bytes.

Set	Variant	λ	q	m	n	k	r	s	N	D	τ	Bit security	Public key	Signature
Ia	fast	128	16	15	15	78	6	5	2	4	39	144	129	7,877
	short							9	2	8	19			5,673
	shorter							12	2	12	13			5,036
	shortest							12	2	16	10			4,536
Ib	fast	128	16	16	16	142	4	5	2	4	39	159	144	9,105
	short							9	2	8	19			6,309
	shorter							12	2	12	13			5,491
	shortest							12	2	16	10			4,886
IIIa	fast	192	16	19	19	109	8	7	2	4	55	207	205	17,139
	short							9	2	8	29			12,440
	shorter							13	2	12	19			10,746
	shortest							13	2	16	15			9,954
IIIb	fast	192	16	19	19	167	6	7	2	4	55	232	205	18,459
	short							9	2	8	29			13,136
	shorter							13	2	12	19			11,202
	shortest							13	2	16	15			10,314
Va	fast	256	16	21	21	189	7	10	2	4	71	273	253	31,468
	short							10	2	8	38			21,795
	shorter							14	2	12	26			19,393
	shortest							14	2	16	20			17,522
Vb	fast	256	16	22	22	254	6	10	2	4	71	301	274	34,059
	short							10	2	8	38			23,182
	shorter							14	2	12	26			20,394
	shortest							14	2	16	20			18,292

The remaining parameters, i.e., the number of rows of the challenge matrix R, the number of parties N^D in the underlying MPC protocol and the number τ of repetitions of the identification protocol, are computed such that the forgery cost of the best known generic attack on 5-pass protocols from [KZ20] is at least 2^λ.

We provide two classes of parameter sets, namely: a and b sets. The Xa sets minimize the signature size, while precisely matching the NIST security level definitions. We re-optimize the original parameter sets and find that they allow for a slightly smaller choice of k. This reduces signature size and increases efficiency. Following [ARZV23], we provide with the Xb sets a more conservative choice of parameters which include some security

margin to account for potential future attack improvements.

Further, for each parameter set, we provide a "Fast", "Short","Shorter" and "Shortest" variant leveraging a trade-off between signing/verification time and signature size that originates from the use of the MPC protocol. In fact, lowering the number of MPC parties allows to reduce the computation time, but to maintain the same forgery cost the number of repetitions has to be increased, which in turn results in larger signature sizes.

4.1 Signature and Key Sizes

Additionally, Table 2 states the public key sizes $\lambda + mn \log q$ (compare to Section 3.3) and the corresponding maximum signature size. The maximum signature for MiRitH is of size (compare to Fig. 4)

$$\underbrace{6\lambda}_{\mathsf{salt},h_1,h_2} + \tau \left(\underbrace{(k + r(n-r) + s(n-r) + sr) \cdot \log q}_{[\![\boldsymbol{\alpha}^{(\ell)}]\!]_{N^D},[\![K^{(\ell)}]\!]_{N^D},[\![C^{(\ell)}]\!]_{N^D},[\![S^{(\ell)}]\!]_{i^*,(\ell)}} + \underbrace{\lambda \cdot D \cdot \log_N}_{(\mathsf{seed}_i^{(\ell)})_{i \neq i^*}} + \underbrace{2\lambda}_{\mathsf{com}_{i^*,(\ell)}} \right)$$

bits. Note that in comparison to Fig. 4 the number of parties changed from N to N^D due to the hypercube improvement (see Section 3.4). The secret key for each parameter set is derived from a single seed of size λ bits.

4.2 Quantum Analysis

We restrict our analysis to polynomial memory quantum algorithms. This is motivated by the fact that it is widely unclear to which degree QRAM such as quantum accessible classical memory (QACM) or quantum accessible quantum memory (QAQM) can be realized in practice. Also the computational overheads of the corresponding circuits are expected to be significant. Therefore we are interested in leveraging quantum search speedups, also known as Grover search, to solve the MinRank problem.

Grover Search [Gro96] Given a quantum accessible function $\mathcal{F}\colon D \to \{0,1\}$ with a unique element $x \in D$ for which it holds that $\mathcal{F}(x) = 1$, a Grover search finds the element $x \in D$ within $\mathcal{O}(\sqrt{D})$ calls to the quantum oracle $\mathcal{F}$ with high probability.

A first obvious strategy is a quantum search improved brute-force for the solution $\boldsymbol{\alpha} \in \mathbb{F}_q^k$. In that case the quantum oracle $\mathcal{F}$ is defined with domain $\mathbb{F}_q^k$ and $\mathcal{F}(\mathbf{x}) = 1$ iff $\mathbf{x}$ is a solution to the MinRank problem. The amount of calls to the oracle to find the solution $\boldsymbol{\alpha}$ in that case is $\mathcal{O}(2^{(k \log q)/2})$. However, in the following we construct a better suited quantum oracle to be used for a Grover search which relies on the Kipnis-Shamir modeling (see Eq. (1)).

4.2.1 A Quantum Algorithm Solving MinRank

The main idea of our quantum algorithm consists in searching for a subset of the columns of the secret matrix K from Eq. (1).

By Eq. (1), any set of t columns of K yields an affine linear system of k variables and tm equations, where the MinRank solution $\boldsymbol{\alpha}$ is also a solution to that linear system. In particular, the vector $(\boldsymbol{\alpha}, 1) \in \mathbb{F}_q^{k+1}$ belongs to the left kernel of the matrix representing the linear system. This fact is highlighted in Proposition 2, where we also show the matrix defining the linear system for every guess of the first t columns of K.

Proposition 2. *Let $\boldsymbol{M} = (M_0, \ldots, M_k)$ be a MinRank instance, where $M_\ell = [\mu_{(i,j)}^{(\ell)}]_{i,j=1}^{m,n} \in \mathbb{F}_q^{m \times n}$ with solution $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_k)$. Let $K = [\kappa_{(i,j)}]_{i,j=1}^{r,n-r} \in \mathbb{F}_q^{r \times (n-r)}$ be as in Eq. (1)*

and $1 \leq t \leq n-r$ *be an integer. For* $\mathbf{x}_t = (x_{(1,1)}, x_{(1,2)}, \ldots, x_{(r,t)}) \in \mathbb{F}_q^{rt}$ *define*

$$B(\mathbf{x}_t) = \begin{pmatrix} B_1(\mathbf{x}_t) \\ \vdots \\ B_t(\mathbf{x}_t) \end{pmatrix}^\top \in \mathbb{F}_q^{(k+1)\times(mt)},$$

where

$$B_j(\mathbf{x}_t) = -\begin{pmatrix} \mu_{(1,j)}^{(1)} & & \mu_{(1,j)}^{(k)} & \mu_{(1,j)}^{(0)} \\ \vdots & \cdots & \vdots & \vdots \\ \mu_{(m,j)}^{(1)} & & \mu_{(m,j)}^{(k)} & \mu_{(m,j)}^{(0)} \end{pmatrix} +$$

$$\begin{pmatrix} \sum_{i=1}^r \mu_{(1,n-r+i)}^{(1)} x_{(i,j)} & & \sum_{i=1}^r \mu_{(1,n-r+i)}^{(k)} x_{(i,j)} & \sum_{i=1}^r \mu_{(1,n-r+i)}^{(0)} x_{(i,j)} \\ \vdots & \cdots & \vdots & \vdots \\ \sum_{i=1}^r \mu_{(m,n-r+i)}^{(1)} x_{(i,j)} & & \sum_{i=1}^r \mu_{(m,n-r+i)}^{(k)} x_{(i,j)} & \sum_{i=1}^r \mu_{(m,n-r+i)}^{(0)} x_{(i,j)} \end{pmatrix}.$$

Then it holds that $(\alpha_1, \ldots, \alpha_k, 1) \cdot B(\boldsymbol{\kappa}_t) = \mathbf{0}$, *where* $\boldsymbol{\kappa}_t := (\kappa_{(1,1)}, \kappa_{(1,2)}, \ldots, \kappa_{(r,t)}) \in \mathbb{F}_q^{rt}$ *is formed from the first columns of* K.

Note that Proposition 2 implies that $B(\boldsymbol{\kappa}_t)$ is not full-rank, and, in particular, its last row belongs to the vector space spanned by the remaining ones. We want to use this as our distinction factor for the vector κ_t in the set $\mathbb{F}_q^{tr}$. To this end, we choose t to be large enough so that, with high probability, $B(\boldsymbol{\kappa}_t)$ is the unique matrix in $\{B(\boldsymbol{\gamma}) : \boldsymbol{\gamma} \in \mathbb{F}_q^{tr}\}$ that is not of full-rank.

Precisely, we choose t to be the smallest integer such that $\lceil k/m \rceil \leq t \leq n-r$ and $rt \leq \binom{mt}{k+1}$. Hence $k+1 \leq mt$, and $\boldsymbol{\kappa}_t \in \mathbb{F}_q^{rt}$ is expected to be the only vector such that $B(\boldsymbol{\kappa}_t)$ is not full-rank. Indeed, the total number of different matrices $B(\boldsymbol{\gamma})$ with $\boldsymbol{\gamma} \in \mathbb{F}_q^{rt}$ is at most q^{rt}, each of them has $\binom{mt}{k+1}$ maximal minors, and any of these minors is zero with probability close to $1/q$. Therefore, the expected number of matrices $B(\boldsymbol{\gamma})$ that are not full-rank is close to $q^{rt}/q^{\binom{mt}{k+1}} \leq 1$.

Finally, note that once $\boldsymbol{\kappa}_t$ is known, the MinRank solution $\boldsymbol{\alpha}$ can be efficiently recovered by classical Gaussian elimination using the relation from Proposition 2.

Let us now define the necessary quantum oracle. We aim for a quantum oracle $\mathcal{F} : \mathbb{F}_q^{rt} \to \mathbb{F}_2$ such that

$$\mathcal{F}(\boldsymbol{\gamma}) = \begin{cases} 1 & \text{if } \boldsymbol{\gamma} = \boldsymbol{\kappa}_t \\ 0 & \text{otherwise.} \end{cases}$$

The computation of our quantum oracle $\mathcal{F}$ on input $\boldsymbol{\gamma} := (\gamma_{(1,1)}, \gamma_{(1,2)}, \ldots, \gamma_{(r,t)}) \in \mathbb{F}_q^{rt}$ splits into three parts:

1. Compute the matrix $B(\boldsymbol{\gamma})$.
2. Compute, by Gaussian elimination, the row-reduced form of $B(\boldsymbol{\gamma})$.
3. Apply a NOT gate to every qubit representing the last row of $B(\boldsymbol{\gamma})$, and then output the product of such qubits.

Notice that $\mathcal{F}(\boldsymbol{\gamma}) = 1$ if and only if at the end of step 2, the last row of $B(\boldsymbol{\gamma})$ is the zero vector.

Grover's algorithm states that the vector $\boldsymbol{\kappa}_t$ can be found with high probability after $\mathcal{O}\big(2^{(rt \log q)/2}\big)$ calls to the quantum oracle $\mathcal{F}$. The complexity of computing $\mathcal{F}$ is dominated

by the Gaussian elimination step, which can be performed in $\mathcal{O}\left((k+1)^2 \cdot mt\right)$ operations in $\mathbb{F}_q$. Therefore, we estimate our quantum complexity to be

$$\mathcal{O}\big(2^{(rt\log q)/2} \cdot (k+1)^2 mt\big)$$

operations over $\mathbb{F}_q$.

4.2.2 Quantum Security of Parameter Sets

For the quantum security definition of categories I, III and V provided by NIST, the maximum depth of the used quantum circuits is limited to $2^{\texttt{maxdepth}}$ with $\texttt{maxdepth} \leq 96$. A parameter set is said to match the quantum security definition for a category if an attack requires at least $2^{b-\texttt{maxdepth}}$ quantum gates for $b = 157, 221, 285$ for category I, III and V, respectively.

We lower bound the depth of the described quantum circuit by

$$D = 2^{(rt\log q)/2} k^2,$$

which corresponds to the sequential repetition of the Grover iterations, where we lower bound the depth of the oracle with k^2.

In the case of $D > 2^{\texttt{maxdepth}}$, the most efficient strategy [Zal99] to restrict the depth of the quantum circuit is to partition the search space in P equally sized, small enough sets. Subsequently, the search has to be reapplied for each of the P partitions, which comes at a depth of

$$D_P = \frac{D}{\sqrt{P}},$$

and it leads to $D_P = 2^{\texttt{maxdepth}}$ for a choice of $P = (D/2^{\texttt{maxdepth}})^2$.

The total number of quantum gates necessary to launch the depth-limited attack becomes

$$T = \mathcal{O}(P \cdot D_P \cdot mt \log^2 q), \tag{3}$$

where we count $\log^2 q$ gates per field multiplication.

In Table 3, we state the estimated quantum security margin for each parameter set. That is the quotient of the gates necessary to launch a quantum attack according to Eq. (3) and the defined security threshold of $2^{b-\texttt{maxdepth}}$ for $b = 157, 221, 285$ for category I, III and V respectively. Note that all parameter sets have a positive margin, i.e., they offer higher quantum security than AES with the corresponding key length.

Table 3: Quantum security margin of MiRitH parameter sets.

Set	Ia	Ib	IIIa	IIIb	Va	Vb
t	6	10	6	9	10	12
security margin (bits)	23	43	9	36	37	47

5 Implementation

We provide three implementations for MiRitH, namely: one reference implementation and three optimized constant-time implementations.

The first optimized implementation is for Intel® processors and optimizes the matrix arithmetic as well as the use of symmetric primitives such as hash functions by using AVX2 instructions. The second optimized implementation is for ARM processors and optimizes those parts by using NEON instructions. Last but not least we provide an

implementation for low-power ARM devices that do not support the NEON instruction set like a Cortex M4. All of our implementations share the same code base, with the exception of the vector finite field arithmetic and the hash function. Those are specialized for each architecture. Additionally, the field is represented as $\mathbb{F}_{2^4} = \mathbb{F}_2[x]/(x^4 + x + 1)$ and we use the straightforward conversion, i.e., 1 corresponds to 1, 2 corresponds to x^1, 4 corresponds to x^2, and so forth.

5.1 Symmetric Primitives

For the symmetric primitives we use `shake256` as PRG and `sha3` as hash function leveraging the Extended Keccak Code Package (XKCP[1]), as it is currently the fastest known implementation of all Keccak associated algorithms.

5.2 Arithmetic

For all implementations: AVX2, NEON, M4 and reference implementation, we pack each column into the memory, i.e., there is no alignment between two subsequent columns, thus all our matrices are column-major.

Additionally as the matrices in MiRitH are generally small we are able to fully load each matrix into a small set of registers, and thus are not forced to reload each column upon the next step in the matrix-matrix multiplication as done by [BCH+23].

5.2.1 Reference Implementation

This implementation is written entirely in ANSI C. It has the only purpose of showing how the proposed scheme can be implemented without employing any particular optimization. As all our implementations it represents two finite field elements from $\mathbb{F}_{2^4}$ in a single byte. The reference implementation implements the finite field arithmetic over $\mathbb{F}_{2^4}$ via a lookup table. The matrix arithmetic is computed element-wise, resulting in n^3 table lookups for a matrix-matrix multiplication of two $n \times n$ matrices.

The reference implementation relies on the symmetric primitives of the OpenSSL library due to its acceptable speed and great availability among most platforms.

5.2.2 AVX Implementation

Our optimized implementation of the finite field arithmetic uses the Advanced Vector Extension (AVX) and Advanced Vector Extension 2 (AVX2) instruction sets, which have been part of every X86 CPU since the Intel Haswell generation in 2013. These instruction sets introduced 256-bit vector registers while following the Single-Instruction-Multiple-Data (SIMD) approach. Thus, a broad variety of instructions are available, which allow to manipulate a 256-bit vector as 8-/16-/32-/64-bit vector elements, while computing all vector elements simultaneously in constant time. Newer x86 CPUs even support AVX512 which introduced 512-bit vectors and instructions. But as this extensions did not find broad adoption in consumer hardware, the main focus of this work lies on an AVX2 implementation.

Finite Field Arithmetic Before we start with the description of our AVX implementation, let us first have a look into how the scalar multiplication in plain C could be implemented. Given two elements $a = a_3x^3 + a_2x^2 + a_1x + a_0$ and $b = b_3x^3 + b_2x^2 + b_1x + b_0 \in \mathbb{F}_{2^4}$, the multiplication $a \cdot b$ can be expressed as $\sum_{i=0}^{3} a_i \cdot x^i \cdot b$.

The code is straightforward, we extract the four bits of a and multiply each with $x^i b$, which is implemented as a lookup into the precomputed table p containing those

[1] https://github.com/XKCP/XKCP

```
1 tmp1 = ((a & 0x1) >> 0) * p[b*4 + 0];
2 tmp2 = ((a & 0x2) >> 1) * p[b*4 + 1];
3 tmp3 = ((a & 0x4) >> 2) * p[b*4 + 2];
4 tmp4 = ((a & 0x8) >> 3) * p[b*4 + 3];
5 c    = tmp1^tmp2^tmp3^tmp4;
```

Figure 6: Computation of $c = a \cdot b$ for $a, b, c \in \mathbb{F}_{2^4}$. The lookup table is build as follows: $\{0x^0, 0x^1, 0x^2, 0x^3, 1x^0, 1x^1, 1x^2, 1x^3, \cdots, 15x^0, 15x^1, 15x^2, 15x^3\}$

values. Lastly, we aggregate the partial products by means of XOR operations. This is the foundation of the AVX implementation, which is a vectorized version of the code in Fig. 6.

The core idea of our implementation is the usage of the `vpblendvb` instruction. Given three registers `ymm_a`, `ymm_b`, `ymm_c` $\in \mathbb{F}_{2^4}^{32}$ this instruction returns each vector element by selecting either the corresponding vector element from `ymm_b` or `ymm_a` if the highest bit in the vector element of `ymm_c` is set or not.

Since $a_i \in \mathbb{F}_2$, an addend $x^i \cdot b$ in the computation of $c = \sum_{i=0}^{3} a_i \cdot x^i \cdot b$ is selected depending on a_i, which matches the usage of the `vpblendvb` instruction.

Each register `ymm_a` and `ymm_b` is filled with the maximum amount of rows/columns of the two input matrices they can hold. The multiplication with x^i is implemented via a small lookup table, which fits into three AVX 256-bit registers utilizing the `vpshufb` instruction. This instruction shuffles 8-bit vector elements across 128-bit lanes based on a given 4-bit index. More precisely, given two 256-bit registers `ymm_v_i` $= (v_0, \ldots, v_{31}) = (0x^i, 1x^i, \ldots, 15x^i, 0x^i, \cdots, 15x^i) \in \mathbb{F}_{2^4}^{32}$, representing the lookup table and `ymm_w` $= (w_0, \ldots, w_{31}), w_i \in \mathbb{F}_{2^4}$ the lookup is performed by simply computed as:

$$\texttt{ymm\_t} = \texttt{vpshufb}(\texttt{ymm\_v\_i}, \texttt{ymm\_w}) = (v_{w_0}, \ldots, v_{w_{31}}) = (w_0 x^i, \ldots, w_{31} x^i).$$

Note that only the lowest four bits of each vector element of the variable `ymm_w` are used as a lookup index, thus the hardware restriction of shuffling only on 128-bit lanes. Additionally note that three registers `ymm_v_i` for $i = 1, 2, 3$ are sufficient to fully save all needed powers of x. Fig. 7 shows how a single addend of the sum $\sum_{i=0}^{3} a_i x^i \cdot b$ is computed. By repeating the code from Fig. 7 for each $i = 0, 1, 2, 3$, one obtains the full polynomial multiplication.

The first line applies the multiplication with x^i via the `vpshufb` instruction, while the second shifts each 4-bit nibble into the higher 4 bits of each 8-bit vector element. Lines 3 and 4 shift the lowest bit of each nibble into the highest bit of the corresponding byte, preparing it to be the selection bit for the two subsequent lines. These lines apply the `vpblendvb` instruction, selecting either zero or the permuted `ymm_b` value, depending on the i-th bit of each nibble of `ymm_a`. The final result must be added together from the low and high nibbles.

```
1 __m256i lowlookup  = _mm256_shuffle_epi8(ymm_v_i, ymm_b);
2 __m256i highlookup = _mm256_slli_epi16(lowlookup, 4);
3 __m256i tmplow  = _mm256_slli_epi16(ymm_a, 7-i);
4 __m256i tmphigh = _mm256_slli_epi16(ymm_a, 3-i);
5 __m256i tmpmu1low  = _mm256_blendv_epi8(zero, lowlookup , tmplow);
6 __m256i tmpmulhigh = _mm256_blendv_epi8(zero, highlookup, tmphigh);
7 __m256i tmp = _mm256_xor_si256(tmpmullow, tmpmulhigh);
```

Figure 7: Computation of $a_i x^i \cdot b$ for $a, b \in \mathbb{F}_{2^4}^{64}$

In total we therefore spend 12 `shift`, 3 `vpshufb`, 8 `vpblendvb` and 7 `XOR` instructions, which can be computed in 29.5 cycles on a Ryzen 7600X. This is an improvement of 11.5% over the implementation of [BCH+23] which uses 33.3 cycles on the same system.

Table 4: Median time (in MCycles) for the AVX2 and NEON-ARM implementation of MiRitH on an 11th Gen Intel(R) Core(TM) i7-11850H @ 2.50GHz (Turbo Boost disabled) and on an Apple M1 Max chip, respectively.

Set	Variant	AVX2				NEON-ARM			
		Key generation	Signing		Verification	Key generation	Signing		Verification
			Online (%)	Total			Online (%)	Total	
Ia	fast	0.109	28.02	5.19	4.72	0.083	23.75	3.33	2.92
	short		4.24	31.91	31.76		3.19	23.06	22.75
	shorter		0.41	329.98	326.08		0.30	251.04	250.40
	shortest		0.04	4,109.28	4,126.42		0.02	3,108.80	3,111.62
Ib	fast	0.197	28.11	5.51	4.94	0.158	21.37	4.15	3.77
	short		4.40	31.40	31.47		3.57	24.14	23.79
	shorter		0.43	332.39	332.53		0.34	256.70	254.56
	shortest		0.04	4,072.82	4,150.09		0.03	3,175.63	3,155.11
IIIa	fast	0.247	27.30	11.10	10.43	0.206	23.52	6.71	6.16
	short		5.65	54.42	54.93		4.09	38.85	38.33
	shorter		0.61	529.15	538.78		0.34	389.20	395.78
	shortest		0.05	6,777.93	6,750.47		0.04	4,939.69	4,963.37
IIIb	fast	0.373	27.29	12.67	12.34	0.315	22.91	7.27	6.99
	short		6.02	56.07	56.53		4.25	39.83	39.40
	shorter		0.65	532.81	539.27		0.47	400.99	397.08
	shortest		0.06	6,784.73	6,807.26		0.04	5,096.71	5,066.26
Va	fast	0.515	32.38	19.33	17.95	0.408	26.12	12.66	11.92
	short.		5.73	92.39.	92.23		4.42	64.95	64.26
	shorter		0.61	1,040.72	1,054.26		0.45	765.62	771.81
	shortest		0.05	12,869.20	12,966.45		0.04	9,485.61	9,496.04
Vb	fast	0.703	32.19	21.96	20.44	0.602	25.47	15.19	14.62
	short.		5.72	103.53	103.47		4.42	76.55	76.44
	shorter		0.62	1,047.11	1,064.93		0.47	791.42	796.771
	shortest		0.06	12,954.79	13,018.82		0.04	9,743.79	9,866.68

`PCLMULQDQ` Instruction: In our AVX2 code we are not using the `PCLMULQDQ` instruction as its computing the carry-less multiplication over $\mathbb{F}_{2^{64}}$ and not over $\mathbb{F}_{2^4}$ as we need it.

5.2.3 NEON Implementation

ARM introduced its SIMD solution in 2004 with the ARMv7 Instruction Set Architecture (ISA), which was extended for ARMv8 to process 128-bit registers. This ISA is implemented by Apple on the M1 Max chip, which is our test system.

Finite Field Arithmetic The core of our implementation is based on [BCH+23] by utilizing the `vmulq_p8` instruction. This instruction computes an 8-bit polynomial multiplication on 16 vector elements from two input registers. The subsequent reduction is an implementation using a lookup table. The only difference to the arithmetic by [BCH+23] is that our implementation, as the AVX2 version, preloads all columns/rows into registers, which speeds up the overall computation of the matrix-matrix multiplication.

The symmetric primitives are the same as in the AVX2 version, with the only difference being that we use the generic 64-bit version of `SHA3` rather than the ARMv8 assembly version, which surprisingly performed faster on our benchmarking system.

5.2.4 M4 Implementation

Additionally to the optimized NEON implementation for the ARM architecture, we provide a version for low-power devices like the Cortex-M4. These chips are often built into restricted IoT devices, because of their low power consumption.

Table 5: Median time (in MCycles) on a STM32F407G Cortex M4 over ten iterations.

Set	Variant	Reference			Bit-Slice		
		Key generation	Signing	Verification	Key generation	Signing	Verification
Ia	fast	1.74	172	141	1.18	87.5	84.8
Ib	fast	2.98	169	144	2.23	135	117

Finite Field Arithmetic Chips like the Cortex-M4 do not provide any SIMD instruction set like NEON. Therefore our implementation is based upon the proposed bitsliced implementation of [BCH+23].

This technique breaks a field element into its individual bits and processes each bit separately. Therefore one can pack the same bit of multiple field elements into a single register. As each bit is treated as independent, the same operation is performed on all corresponding bits of multiple binary numbers simultaneously. This parallel processing of individual bits can lead to significant speed improvements.

For MiRitH this means instead of using a lookup table for the multiplication, writing the multiplication circuit in software using general logic instructions. Note that, before (and after) each multiplication, one needs to transform the input data into (and from) its bitsliced form.

5.3 Performance Data

In this section, we provide performance evaluations for our optimized constant-time implementations.

For the AVX2 and NEON implementations we measured all parameter sets of MiRitH. The key generation, signature generation, and verification benchmarks were performed on an 11th Gen Intel(R) Core(TM) i7-11850H @ 2.50GHz (Turbo Boost disabled) and on an Apple M1 Max chip, respectively. The reported timings are the medians and are shown in Table 4 given in million CPU cycles.

For the Cortex M4 implementation we measured only the Ia fast and Ib fast parameter sets of MiRitH. The key generation, signature generation, and verification benchmarks were performed on a STM32F407G Cortex M4 over 10 iterations. The reported timings are the medians and are shown in Table 5 given in million CPU cycles.

5.4 Comparison with Other NIST Candidates and Beyond

In this section we compare the performance of MiRitH against its competitors in the renewed NIST standardization process as well as some selected schemes, which are not part of the current competition. Regarding NIST candidates, we include all schemes that follow a similar construction strategy, i.e., either they also rely on the MPCitH paradigm or employ more generally the Fiat-Shamir transform.[2]

Table 6 states the performance data of all considered candidates on a single benchmarking system. Additionally, we provide public key and signature sizes. All schemes besides MEDS and LESS obtain generally small public keys of size less than 0.25 kB. The smallest signatures are obtained by constructions based on symmetric primitives, such as AIMER and FAEST as well as by BISCUIT, which relies on a variant of the MQ problem. In terms of signing and verification times, many schemes lie below the threshold of 5 MCycles, including MiRitH (for its fast instantiation). We also provide the necessary script to

[2]With the exception of SQIsign [CSSF+23] as it has a completely different performance profile, targeting different applications.

Table 6: Signature Size and Performance comparison in Kilobytes (kB) and MCycles (Mcc), respectively, of MiRitH against MPCitH- and Fiat-Shamir-based NIST submissions on an 11th Gen Intel(R) Core(TM) i7-11850H @ 2.50GHz (Turbo-Boost disabled), compiled using gcc 11.4.0.

Name	Variant	Signature Size (kB)			Performance (Mcc)		
		pk	σ	$pk + \sigma$	Keygen	Sign	Verify
AIMER [KCC+23, KHS+22]	param1	0.03	5.77	5.80	0.066	1.76	1.42
	param2	0.03	4.77	4.80	0.065	3.66	3.37
	param3	0.03	4.08	4.11	0.070	11.62	11.28
	param4	0.03	3.84	3.87	0.073	56.52	56.72
BISCUIT [BKPV23]	fast	0.05	6.57	6.62	0.083	6.87	6.23
	short	0.05	4.65	4.70	0.084	56.02	55.69
CROSS [BBB+23b, BBP+23]	fast	0.04	8.46	8.50	0.036	4.06	2.84
	small	0.04	7.45	7.48	0.036	14.64	10.28
FAEST [BBK+23]	fast	0.03	6.19	6.22	0.002	2.74	2.70
	short	0.03	4.89	4.92	0.002	25.58	25.83
LESS [BBB+23a, BBPS21]	small-pk	13.38	8.20	21.58	0.922	249.88	261.07
	large-pk	93.65	5.08	98.73	5.068	200.55	209.43
MEDS [CNP+23]	fast	12.91	12.67	25.58	1.564	51.50	51.41
	small	9.69	9.66	19.35	1.193	307.02	305.20
MIRA [ABB+23d, Fen22]	fast	0.08	7.20	7.29	0.130	44.22	43.58
	short	0.08	5.51	5.59	0.132	52.20	49.72
MQOM [FR23]	gf31-fast	0.05	7.44	7.49	0.598	15.03	13.75
	gf31-short	0.05	6.20	6.25	0.605	36.13	34.70
	gf251-fast	0.06	7.63	7.68	0.454	9.01	8.20
	gf251-short	0.06	6.42	6.48	0.460	20.31	19.70
PERK [ABB+23a]	fast-3	0.15	8.15	8.30	0.085	7.52	5.13
	short-3	0.15	6.41	6.55	0.090	38.34	25.83
	fast-5	0.23	7.84	8.08	0.098	7.07	4.86
	short-5	0.23	5.92	6.15	0.104	35.21	24.21
RYDE [ABB+23c, Fen22]	fast	0.08	7.27	7.36	0.059	5.60	4.75
	short	0.08	5.82	5.90	0.076	24.62	21.52
SDitH [MFG+23]	gf256-hyp	0.12	8.05	8.17	7.083	13.59	12.63
MiRitH (this work)	fast	0.13	7.69	7.82	0.108	4.70	4.45
	short	0.13	5.54	5.67	0.108	32.08	31.90

Table 7: Median time (in MCycles) for MiRitH and MIRA implementations on an 11th Gen Intel(R) Core(TM) i7-11850H @ 2.50GHz (Turbo Boost disabled).

Set	Variant	MiRitH			MIRA		
		Key generation	Signing	Verification	Key generation	Signing	Verification
I	fast	0.109	5.19	4.72	0.126	43.71	43.16
	short		31.91	31.76	0.131	51.86	49.45
III	fast	0.247	11.10	10.43	0.336	124.49	123.97
	short		54.42	54.93	0.343	137.47	134.03
V	fast	0.515	19.33	17.95	0.795	375.20	374.66
	short		92.39	92.23	0.799	385.78	382.48

relaunch the benchmark on an arbitrary system. The single system benchmark shows that MiRitH offers a higher speedup over MIRA than implied by the specification document.

Table 8: Signature Size and Performance comparison in Kilobytes (kB) and MCycles (Mcc), respectively, of MiRitH against other MPCitH-based schemes on an 11th Gen Intel(R) Core(TM) i7-11850H @ 2.50GHz (Turbo-Boost disabled), compiled using gcc 11.4.0.

Name	Variant	Signature Size (kB)			Performance (Mcc)		
		pk	σ	$pk + \sigma$	Keygen	Sign	Verify
BN++LowMC [KZ22]	$N = 16\ \tau = 34$	0.03	12.52	12.56	0.015	7.19	6.95
	$N = 256\ \tau = 18$	0.03	7.80	7.83	0.021	60.52	61.98
BN++ Rain_3 [KZ22]	$N = 16\ \tau = 33$	0.03	6.28	6.31	0.014	2.11	1.92
	$N = 256\ \tau = 17$	0.03	4.34	4.38	0.017	15.76	15.71
Helium+AES [KZ22]	$N = 17\ \tau = 31$	0.03	17.17	17.20	0.005	23.59	21.29
	$N = 255\ \tau = 16$	0.03	9.66	9.69	0.006	60.37	57.66
Helium+LowMC [KZ22]	$N = 17\ \tau = 31$	0.03	10.91	10.94	0.018	19.37	18.37
	$N = 255\ \tau = 16$	0.03	6.43	6.46	0.020	64.81	63.19
Banquet [BDK+21]	$N = 16\ \tau = 37$	0.03	20.47	20.50	0.006	19.00	14.30
	$N = 255\ \tau = 21$	0.03	12.97	13.00	0.008	124.05	112.08
Rainer_3 [DKR+22]	$N = 16\ \tau = 33$	0.03	8.34	8.38	0.012	2.40	2.24
	$N = 256\ \tau = 17$	0.03	5.41	5.44	0.014	15.91	15.74
MiRitH (this work)	fast	0.13	7.69	7.82	0.108	4.70	4.45
	short	0.13	5.54	5.67	0.108	32.08	31.90

In Table 7 we provide a direct comparison between all suggested MiRitH and MIRA instantiations benchmarked again on a single system. We observe that the speedup of MiRitH over MIRA increases together with the security category and is generally higher than indicated by the specification documents.

Eventually, in Table 8 we provide a comparison to recently proposed schemes relying on the MPCitH paradigm in combination with symmetric primitives, which are not part of the current NIST competition.

References

[ABB+23a] Najwa Aaraj, Slim Bettaieb, Loïc Bidoux, Alessandro Budroni, Victor Dyseryn, Andre Esser, Philippe Gaborit, Mukul Kulkarni, Victor Mateu, Marco Palumbi, Lucas Perin, and Jean-Pierre Tillich. PERK specification. 2023. available at https://pqc-perk.org/assets/downloads/PERK_specifications.pdf.

[ABB+23b] Gora Adj, Stefano Barbero, Emanuele Bellini, Andre Esser, Luis Rivera-Zamarripa, Carlo Sanna, Javier Verbel, and Floyd Zweydinger. MiRitH specification. 2023. available at https://pqc-mirith.org/assets/downloads/mirith_specifications_v1.0.0.pdf.

[ABB+23c] Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Matthieu Rivain, Jean-Pierre Tillich, and Adrien Vinçotte. RYDE specification. 2023. available at https://pqc-ryde.org/assets/downloads/RYDE_Specifications.pdf.

[ABB+23d] Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, Matthieu Rivain, and Jean-Pierre Tillich. MIRA specification. 2023. available at https://pqc-mira.org/assets/downloads/mira_spec.pdf.

[AGH+23] Carlos Aguilar Melchor, Nicolas Gama, James Howe, Andreas Hülsing, David Joseph, and Dongze Yue. The return of the SDitH. In Carmit Hazay and

Martijn Stam, editors, *EUROCRYPT 2023, Part V*, volume 14008 of *LNCS*, pages 564–596. Springer, Heidelberg, April 2023.

[ARZV23] Gora Adj, Luis Rivera-Zamarripa, and Javier A. Verbel. MinRank in the head - short signatures from zero-knowledge proofs. In *AFRICACRYPT 23*, LNCS, pages 3–27. Springer Nature, June 2023.

[BBB+23a] Marco Baldi, Alessandro Barenghi, Luke Beckwith, Jean-François Biasse, Andre Esser, Kris Gaj, Kamyar Mohajerani, Gerardo Pelosi, Edoardo Persichetti, Markku-Juhani Saarinen, Paolo Santini, and Robert Wallace. LESS specification. 2023. available at https://www.less-project.com/LESS-2023-06-01.pdf.

[BBB+23b] Marco Baldi, Alessandro Barenghi, Sebastian Bitzer, Patrick Karl, Felice Manganiello, Alessio Pavoni, Gerardo Pelosi, Paolo Santini, Jonas Schupp, Freeman Slaughter, Antonia Wachter-Zeh, and Violetta Weger. CROSS specification. 2023. available at https://www.cross-crypto.com/CROSS_Specification.pdf.

[BBB+23c] Magali Bardet, Pierre Briaud, Maxime Bros, Philippe Gaborit, and Jean-Pierre Tillich. Revisiting algebraic attacks on minrank and on the rank decoding problem. *Des. Codes Cryptogr.*, 91(11):3671–3707, 2023.

[BBC+20] Magali Bardet, Maxime Bros, Daniel Cabarcas, Philippe Gaborit, Ray A. Perlner, Daniel Smith-Tone, Jean-Pierre Tillich, and Javier A. Verbel. Improvements of algebraic attacks for solving the rank decoding and MinRank problems. In Shiho Moriai and Huaxiong Wang, editors, *ASIACRYPT 2020, Part I*, volume 12491 of *LNCS*, pages 507–536. Springer, Heidelberg, December 2020.

[BBK+23] Carsten Baum, Lennart Braun, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl. FAEST specification. 2023. available at https://faest.info/faest-spec-v1.1.pdf.

[BBP+23] Marco Baldi, Sebastian Bitzer, Alessio Pavoni, Paolo Santini, Antonia Wachter-Zeh, and Violetta Weger. Zero knowledge protocols and signatures from the restricted syndrome decoding problem. *Cryptology ePrint Archive*, 2023.

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

[BCH+23] Ward Beullens, Ming-Shing Chen, Shih-Hao Hung, Matthias J. Kannwischer, Bo-Yuan Peng, Cheng-Jhih Shih, and Bo-Yin Yang. Oil and vinegar: Modern parameters and implementations. *IACR TCHES*, 2023(3):321–365, 2023.

[BDK+21] Carsten Baum, Cyprien Delpech de Saint Guilhem, Daniel Kales, Emmanuela Orsini, Peter Scholl, and Greg Zaverucha. Banquet: Short and fast signatures from AES. In Juan Garay, editor, *PKC 2021, Part I*, volume 12710 of *LNCS*, pages 266–297. Springer, Heidelberg, May 2021.

[BESV22] Emanuele Bellini, Andre Esser, Carlo Sanna, and Javier A. Verbel. MR-DSS - smaller minrank-based (ring-)signatures. In Jung Hee Cheon and Thomas Johansson, editors, *Post-Quantum Cryptography - 13th International*

Workshop, PQCrypto 2022, Virtual Event, September 28-30, 2022, Proceedings, volume 13512 of *Lecture Notes in Computer Science*, pages 144–169. Springer, 2022.

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

[Beu21] Ward Beullens. Improved cryptanalysis of UOV and Rainbow. In Anne Canteaut and François-Xavier Standaert, editors, *EUROCRYPT 2021, Part I*, volume 12696 of *LNCS*, pages 348–373. Springer, Heidelberg, October 2021.

[Beu22] Ward Beullens. Breaking Rainbow takes a weekend on a laptop. In Yevgeniy Dodis and Thomas Shrimpton, editors, *Advances in Cryptology - CRYPTO 2022*, volume 13508 of *Lecture Notes in Computer Science*, pages 464–479. Springer, 2022.

[BFP11] Luk Bettale, Jean-Charles Faugère, and Ludovic Perret. Cryptanalysis of multivariate and odd-characteristic HFE variants. In Dario Catalano, Nelly Fazio, Rosario Gennaro, and Antonio Nicolosi, editors, *PKC 2011*, volume 6571 of *LNCS*, pages 441–458. Springer, Heidelberg, March 2011.

[BFS99] Jonathan F. Buss, Gudmund Skovbjerg Frandsen, and Jeffrey O. Shallit. The computational complexity of some problems of linear algebra. *J. Comput. Syst. Sci.*, 58(3):572–596, 1999.

[BG06] Olivier Billet and Henri Gilbert. Cryptanalysis of Rainbow. In Roberto De Prisco and Moti Yung, editors, *SCN 06*, volume 4116 of *LNCS*, pages 336–347. Springer, Heidelberg, September 2006.

[BG23] Loïc Bidoux and Philippe Gaborit. Compact post-quantum signatures from proofs of knowledge leveraging structure for the PKP, SD and RSD problems. In *Codes, Cryptology and Information Security (C2SI)*, pages 10–42. Springer, 2023.

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

[BKPV23] Luk Bettale, Delaram Kahrobaei, Ludovic Perret, and Javier Verbel. Biscuit specification. 2023. available at https://www.biscuit-pqc.org/.

[CNP+22] Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. Take your MEDS: Digital Signatures from Matrix Code Equivalence. *Cryptology ePrint Archive*, 2022.

[CNP+23] Tung Chou, Ruben Niederhagen, Edoardo Persichetti, Lars Ran, Tovohery Hajatiana Randrianarisoa, Krijn Reijnders, Simona Samardjiska, and Monika Trimoska. MEDS specification. 2023. available at https://www.meds-pqc.org/spec/MEDS-2023-07-26.pdf.

[Cou01] Nicolas Courtois. Efficient zero-knowledge authentication based on a linear algebra problem MinRank. In Colin Boyd, editor, *ASIACRYPT 2001*, volume 2248 of *LNCS*, pages 402–421. Springer, Heidelberg, December 2001.

[CSSF+23] Jorge Chavez-Saab, Maria Corte-Real Santos, Luca De Feo, Jonathan Komada Eriksen, Basil Hess, David Kohel, Antonin Leroux, Patrick Longa, Michael Meyer, Lorenz Panny, Sikhar Patranabis, Christophe Petit, Francisco Rodríguez Henríquez, Sina Schaeffler, and Benjamin Wesolowski. SQIsign specification. 2023. available at https://csrc.nist.gov/csrc/media/Projects/pqc-dig-sig/documents/round-1/spec-files/sqisign-spec-web.pdf.

[CSV17] Daniel Cabarcas, Daniel Smith-Tone, and Javier A. Verbel. Key recovery attack for ZHFE. In Tanja Lange and Tsuyoshi Takagi, editors, *Post-Quantum Cryptography - 8th International Workshop, PQCrypto 2017*, pages 289–308. Springer, Heidelberg, 2017.

[DKR+22] Christoph Dobraunig, Daniel Kales, Christian Rechberger, Markus Schofnegger, and Greg Zaverucha. Shorter signatures based on tailor-made minimalist symmetric-key crypto. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security*, pages 843–857, 2022.

[DSS23] Antonio J. Di Scala and Carlo. Sanna. Smaller public keys for MinRank-based schemes. arXiv preprint, 2023. https://arxiv.org/abs/2302.12447.

[Fen22] Thibauld Feneuil. Building MPCitH-based signatures from MQ, MinRank, rank SD and PKP. Cryptology ePrint Archive, Report 2022/1512, 2022. https://eprint.iacr.org/2022/1512.

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

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

[FMRV22] Thibauld Feneuil, Jules Maire, Matthieu Rivain, and Damien Vergnaud. Zero-knowledge protocols for the subset sum problem from MPC-in-the-head with rejection. In Shweta Agrawal and Dongdai Lin, editors, *ASIACRYPT 2022, Part II*, volume 13792 of *LNCS*, pages 371–402. Springer, Heidelberg, December 2022.

[FR23] Thibauld Feneuil and Matthieu Rivain. MQOM specification. 2023. available at https://mqom.org/docs/mqom-v1.0.pdf.

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

[GC00] Louis Goubin and Nicolas Courtois. Cryptanalysis of the TTM cryptosystem. In Tatsuaki Okamoto, editor, *ASIACRYPT 2000*, volume 1976 of *LNCS*, pages 44–57. Springer, Heidelberg, December 2000.

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

[Gro96] Lov K. Grover. A fast quantum mechanical algorithm for database search. In *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*, STOC '96, page 212–219, New York, NY, USA, 1996. Association for Computing Machinery.

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

[KCC+23] Seongkwang Kim, Jihoon Cho, Mingyu Cho, Jincheol Ha, Jihoon Kwon, Byeonghak Lee, Joohee Lee, Jooyoung Lee, Sangyub Lee, Dukjae Moon, Mincheol Son, and Hyojin Yoon. AIMER specification. 2023. available at https://aimer-signature.org/docs/AIMer-NIST-Document.pdf.

[KHS+22] Seongkwang Kim, Jincheol Ha, Mincheol Son, Byeonghak Lee, Dukjae Moon, Joohee Lee, Sangyub Lee, Jihoon Kwon, Jihoon Cho, Hyojin Yoon, and Jooyoung Lee. AIM: Symmetric primitive for shorter signatures with stronger security (full version). Cryptology ePrint Archive, Paper 2022/1387, 2022. https://eprint.iacr.org/2022/1387.

[KKW18] Jonathan Katz, Vladimir Kolesnikov, and Xiao Wang. Improved non-interactive zero knowledge with applications to post-quantum signatures. In David Lie, Mohammad Mannan, Michael Backes, and XiaoFeng Wang, editors, *ACM CCS 2018*, pages 525–537. ACM Press, October 2018.

[KS99] Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In Michael J. Wiener, editor, *CRYPTO'99*, volume 1666 of *LNCS*, pages 19–30. Springer, Heidelberg, August 1999.

[KZ20] Daniel Kales and Greg Zaverucha. An attack on some signature schemes constructed from five-pass identification schemes. In Stephan Krenn, Haya Shulman, and Serge Vaudenay, editors, *CANS 20*, volume 12579 of *LNCS*, pages 3–22. Springer, Heidelberg, December 2020.

[KZ22] Daniel Kales and Greg Zaverucha. Efficient lifting for shorter zero-knowledge proofs and post-quantum signatures. Cryptology ePrint Archive, Paper 2022/588, 2022. https://eprint.iacr.org/2022/588.

[MFG+23] Carlos Aguilar Melchor, Thibauld Feneuil, Nicolas Gama, Shay Gueron, James Howe, David Joseph, Antoine Joux, Edoardo Persichetti, Tovohery H., Randrianarisoa, Matthieu Rivain, and Dongze Yue. SDITH specification. 2023. available at https://sdith.org/docs/sdith-v1.0.pdf.

[NIS] NIST. Post-Quantum Cryptography – Security (Evaluation Criteria). https://csrc.nist.gov/projects/post-quantum-cryptography/post-quantum-cryptography-standardization/evaluation-criteria/security-(evaluation-criteria). Accessed: March 15, 2023.

[TPD21] Chengdong Tao, Albrecht Petzoldt, and Jintai Ding. Efficient key recovery for all HFE signature variants. In Tal Malkin and Chris Peikert, editors, *CRYPTO 2021, Part I*, volume 12825 of *LNCS*, pages 70–93, Virtual Event, August 2021. Springer, Heidelberg.

[Wan22] William Wang. Shorter signatures from MQ. Cryptology ePrint Archive, Report 2022/344, 2022. https://eprint.iacr.org/2022/344.

[Zal99] Christof Zalka. Grover's quantum searching algorithm is optimal. *Phys. Rev. A*, 60:2746–2751, Oct 1999.