The New Security Proofs of MPC-in-the-Head Signatures in the Quantum Random Oracle Model

Haruhisa Kosuge[1] Keita Xagawa[2,a)]

Abstract: The MPC-in-the-Head paradigm is a promising approach for constructing post-quantum signature schemes. Its significance is underscored by NIST's selection of six signatures based on this paradigm and its variant, VOLE-in-the-Head, among the fourteen round-2 candidates in its additional Post-Quantum Cryptography (PQC) standardization process.
Recent work by Aguilar-Melchor et al. (ASIACRYPT 2023), Hülsing et al. (CRYPTO 2024), and Baum et al. (CRYPTO 2025) has established EUF-CMA security for these signatures in the Quantum Random Oracle Model (QROM). However, their proofs do not account for crucial optimization techniques such as rejection sampling and grinding, rendering them inapplicable to practical implementations like the NIST round-2 candidates Mirath and RYDE.
This paper addresses this gap by analyzing the QROM security of MPC-in-the-Head signatures that incorporate these optimizations, with a focus on Mirath and RYDE. We make two main contributions:
1) We provide a new EUF-CMA security proof that accommodates rejection sampling and grinding. Our proof introduces the adaptive multi-point reprogramming technique, an extension of the adaptive reprogramming from Grilo et al. (ASIACRYPT 2021), which may be of independent interest.
2) We present a new EUF-NMA security proof, also compatible with these optimizations. This result is achieved by extending the proof techniques of Don et al. (CRYPTO 2022) and Aguilar-Melchor et al. (ASIACRYPT 2023).

Keywords: MPC-in-the-Head signatures, QROM, adaptive multi-point reprogramming.

1. Introduction

MPC-in-the-head signatures: One of the most common methods for constructing digital signatures is the *Fiat-Shamir* (FS) transform [FS87], which converts an identification scheme to a signature scheme. A well-known approach for constructing such identification schemes from arbitrary hard problems is the *MPC-in-the-head* (MPCitH) paradigm. This paradigm has seen rapid development in recent years and has come to be widely used as a method for constructing post-quantum signature schemes. In the call for additional digital signature schemes by NIST in 2022 [NIS22], nine signature schemes based on the MPCitH paradigm were submitted. Remarkably, six schemes, signature schemes, FAEST [BBB+24], Mirath [AAB+24], MQOM [BBFR24], PERK [ABB+24a], RYDE [ABB+24c], and SDitH [ABB+24b], remained under consideration as of Round 2, indicating the growing importance and viability of this paradigm. Among the six Round 2 candidates, Mirath, RYDE, and MQOM adopt *Threshold Computation in the Head (TCitH)* paradigm [FR23]. TCitH and *VOLE-in-the-Head* (VOLEitH) represent major variants of the MPCitH paradigm. In this work, we focus on TCitH, which constitutes half of the Round 2 candidates.

TC-in-the-head signatures: TCitH [FR23] is designed to enable efficient zero-knowledge proofs for small circuits. In TCitH, the prover simulates a threshold computation among virtual parties and commits to their views using structured encodings. When instantiated in the Polynomial IOP (PIOP) formalism [Fen24], TCitH allows the prover to commit to low-degree polynomials representing the computation and to reveal selective evaluations while preserving zero-knowledge. The commitment mechanism relies on (batched) all-but-one vector commitments ((B)AVC) to commit to a structured set of pseudorandom seeds [*1]. (B)AVC allows the prover to hide exactly one seed per commitment while revealing the others, and the hidden seeds are used to generate masking values on the secret key. To generate the seeds, we employ a GGM tree structure rooted at a seed that is chosen randomly or generated pseudorandomly. Each node in the tree corresponds to a pseudorandom value derived via a PRF, and the leaves represent the individual seeds assigned to virtual parties in the protocol. The GGM construction ensures that all seeds can be deterministically derived from the root, while allowing selective opening by revealing only the necessary branches. In particular, to hide one party's

[1] NTT Social Informatics Laboratories, Japan
[2] Technology Innovation Institute, UAE
[a)] keita.xagawa@tii.ae

[*1] Mirath and RYDE adopts BAVC to compress τ commiments into one tree. MQOM uses AVC that computes τ trees in parallel.

seed while revealing the others, the prover discloses all sibling nodes along the path to the hidden leaf, enabling the verifier to reconstruct all revealed seeds.

Mirath, RYDE, and MQOM adopt the *grinding* technique to enhance the security, where the second challenge is extended to w bits and the signer computes the challenge untill the extended bits are all zero by incrementing a counter. Mirath and RYDE further adopted *rejecion sampling* to reduce the size of the signature; the singer increment the counter untill the response is shorter than a threshold.

1.1 Our Contribution

In this paper, we discuss the EUF-CMA (Existential Un-Forgeability against Chosen-Message Attack) security of the TCitH signatures, Mirath and RYDE.

We follow the standard proof strategy: the proof consists of reductions of EUF-CMA to EUF-NMA (No-Message Attack) and EUF-NMA to the security properties of the underlying ID scheme.

EUF-NMA *Security:* We formally prove the EUF-NMA security of Mirath in the (e)QROM. We follow the framework in [AHJ+23], the collapsed ID and FS transform, which is upon [DFMS22]. Interestingly, we formally consider *grinding and rejection sampling.*

CMA from NMA: Roughly speaking, we gradually modify the CMA security game to remove the signing key from the signing oracle and allow an NMA adversary to simulate the oracle. Our approach takes a standard way, reprogramming the random oracles and simulating the signing oracle without secret key. To treat grinding and rejection sampling also in the QROM correctly, we prove *the adaptive multi-point reprogramming* technique by extending the adaptive reprogramming technique [GHHM21].

We also discuss the provable security of MQOM, which adopted the TCitH framework also. One of its key charastarics is that the signing algorithm adopts "correlated" GGM tree based on the ideal cipher, where the correlation is a secret key, to reduce signature's size. Unfortunately, this feature prevents us to show MQOM's EUF-CMA security from its EUF-NMA security. We will discuss the detail in the full version.

2. Preliminaries

The security parameter is denoted by $\lambda \in \mathbb{Z}^+$. We use the standard O-notations. For $n \in \mathbb{Z}^+$, we let $[n] := \{0, \ldots, n-1\}$. For $n_1, n_2 \in \mathbb{Z}^+$, we let $[n_1 : n_2] := \{n_1, n_1+1, \ldots, n_2-1\}$ as Python. For a statement P, $\mathsf{bool}P$ denotes the truth value of P. DPT, PPT, and QPT stand for deterministic, probabilistic, and quantum polynomial time, respectively.

Let $\mathcal{X}$ and $\mathcal{Y}$ be two finite sets. $\mathsf{Func}(\mathcal{X}, \mathcal{Y})$ denotes a set of all functions whose domain is $\mathcal{X}$ and codomain is $\mathcal{Y}$.

For a distribution D, we often write "$x \leftarrow D$," which indicates that we take a sample x according to D. For a finite set $\mathcal{S}$, $U(\mathcal{S})$ denotes the uniform distribution over $\mathcal{S}$. We often write "$x \leftarrow \mathcal{S}$" instead of "$x \leftarrow U(\mathcal{S})$." If inp is a string,

```
Game $\mathsf{Expt}^{\text{euf-cma}}_{\mathsf{DS},\mathcal{A}}(1^\lambda)$
1 : $\mathcal{Q} := \emptyset$
2 : $(\mathtt{vk}, \mathtt{sk}) \leftarrow \mathsf{Gen}(1^\lambda)$
3 : $(\mathtt{msg}^*, \sigma^*) \leftarrow \mathcal{A}^{\textsc{Sign}}(\mathtt{vk})$
4 : if $\mathtt{msg}^* \in \mathcal{Q}$ then
5 :     return $0$
6 : return $\mathsf{Vrfy}(\mathtt{vk}, \mathtt{msg}^*, \sigma^*)$
```

```
$\textsc{Sign}(\mathtt{msg})$
1 : $\sigma \leftarrow \mathsf{Sign}(\mathtt{sk}, \mathtt{msg})$
2 : $\mathcal{Q} := \mathcal{Q} \cup \{\mathtt{msg}\}$
3 : return $\sigma$
```

Fig. 1 Games for EUF-CMA security of signature scheme (Definition 2.2).

then "$\mathsf{out} \leftarrow \mathsf{A}^{O}(\mathsf{inp})$" denotes the output of algorithm A running on input inp with an access to a set of oracles O. If A and oracles are deterministic, then out is a fixed value and we write "$\mathsf{out} := \mathsf{A}^{O}(\mathsf{inp})$." We also use the notation "$\mathsf{out} := \mathsf{A}(\mathsf{inp}; r)$" to make the randomness r of A explicit.

For a function $f\colon \{0,1\}^n \to \{0,1\}^m$, a *quantum access* to f is modeled as oracle access to unitary $O_f\colon |x\rangle|y\rangle \mapsto |x\rangle|y \oplus f(x)\rangle$. By convention, we will use the notation $\mathsf{A}^{|f\rangle, g}$ to stress A's *quantum* and classical access to f and g, respectively. For a function $f\colon \mathcal{X} \to \mathcal{Y}$, we denote the procedure reprogramming $f(x)$ with h by $f := f[x \mapsto h]$.

2.1 Digital Signature

The model for digital signature schemes is summarized as follows:

Definition 2.1. *A digital signature scheme* DS *consists of the following triple of PPT algorithms* $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$*:*

- $\mathsf{Gen}(1^\lambda) \to (\mathtt{vk}, \mathtt{sk})$*: a key-generation algorithm that, on input* 1^λ*, outputs a pair of keys* $(\mathtt{vk}, \mathtt{sk})$*.* $\mathtt{vk}$ *and* $\mathtt{sk}$ *are verification and signing keys, respectively.*
- $\mathsf{Sign}(\mathtt{sk}, \mathtt{msg}) \to \sigma$*: a signing algorithm that takes as input signing key* $\mathtt{sk}$ *and message* $\mathtt{msg} \in \mathcal{M}$ *and outputs signature* $\sigma \in \mathcal{S}$*.*
- $\mathsf{Vrfy}(\mathtt{vk}, \mathtt{msg}, \sigma) \to \mathsf{true}/\mathsf{false}$*: a verification algorithm that takes as input verification key* $\mathtt{vk}$*, message* $\mathtt{msg} \in \mathcal{M}$*, and signature* σ *and outputs its decision* 1 *for acceptance or* 0 *for rejection.*

We require statistical correctness; that is, for any message $\mathtt{msg} \in \mathcal{M}$*, we have*

$$\Pr\left[\begin{array}{l}(\mathtt{vk}, \mathtt{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \\ \sigma \leftarrow \mathsf{Sign}(\mathtt{sk}, \mathtt{msg})\end{array} : \mathsf{Vrfy}(\mathtt{vk}, \mathtt{msg}, \sigma) = 1\right] \geq 1 - \delta(\lambda)$$

for some negligible function δ*.*

Security notions: We review the standard security notion, existential unforgeability against chosen-message attack (EUF-CMA), and its variants.

We consider a weak version, existential unforgeability against no-message attack (EUF-NMA), in which the adversary cannot access the signing oracle. We also consider a strong version, sEUF-CMA security, in which the adversary wins if its forgery $(\mathtt{msg}^*, \sigma^*)$ is not equal to the pairs returned by $\textsc{Sign}$. The formal definition follows:

Definition 2.2 (EUF-NMA and EUF-CMA security)**.** *Let* $\mathsf{DS} = (\mathsf{Gen}, \mathsf{Sign}, \mathsf{Vrfy})$ *be a digital signature scheme. For any* $\mathcal{A}$*, we define its* EUF-CMA *advantage against* DS *as*

$$\mathsf{Adv}^{\text{euf-cma}}_{\mathsf{DS},\mathcal{A}}(\lambda) \coloneqq \Pr[\mathsf{Expt}^{\text{euf-cma}}_{\mathsf{DS},\mathcal{A}}(1^\lambda) = 1],$$

where $\mathsf{Expt}^{\text{euf-cma}}_{\mathsf{DS},\mathcal{A}}(1^\lambda)$ *is an experiment described in Figure 1. We say that* DS *is* EUF-CMA*-secure if* $\mathsf{Adv}^{\text{euf-cma}}_{\mathsf{DS},\mathcal{A}}(\lambda)$ *is negligible for any QPT adversary* $\mathcal{A}$*.*

For any $\mathcal{A}$*, we define its* EUF-NMA *advantage against* DS *as* $\mathsf{Adv}^{\text{euf-nma}}_{\mathsf{DS},\mathcal{A}}(\lambda) \coloneqq \Pr[\mathsf{Expt}^{\text{euf-nma}}_{\mathsf{DS},\mathcal{A}}(1^\lambda) = 1]$*, where* $\mathsf{Expt}^{\text{euf-nma}}_{\mathsf{DS},\mathcal{A}}(1^\lambda)$ *is the game* $\mathsf{Expt}^{\text{euf-cma}}_{\mathsf{DS},\mathcal{A}}(1^\lambda)$ *without the signing oracle* Sign*. We say that* DS *is* EUF-NMA*-secure if* $\mathsf{Adv}^{\text{euf-nma}}_{\mathsf{DS},\mathcal{A}}(\lambda)$ *is negligible for any QPT adversary* $\mathcal{A}$*.*

2.2 3/5-Pass Identification

We consider 3-pass and 5-pass ID schemes. We only treat *public-coin* ID schemes; that is, the verifier chooses i-th challenge uniformly at random from the challenge set $\mathcal{C}_i$. The syntax follows:

Definition 2.3 (5-pass identification)**.** *A 5-pass identification scheme* ID *consists of the following tuple of PPT algorithms* $(\mathsf{Gen}, \mathsf{P}_1, \mathcal{C}_1, \mathsf{P}_2, \mathcal{C}_2, \mathsf{P}_3, \mathsf{V})$*:*

- $\mathsf{Gen}(1^\lambda) \to (\mathtt{vk}, \mathtt{sk})$*: a key-generation algorithm that takes* 1^λ *as input, where* λ *is the security parameter, and outputs a pair of keys* $(\mathtt{vk}, \mathtt{sk})$*.* $\mathtt{vk}$ *and* $\mathtt{sk}$ *are public verification and secret keys, respectively.*
- $\mathsf{P}_1(\mathtt{sk}; \rho) \to (a_1, \mathtt{state}_1)$*: a first prover algorithm that takes signing key* $\mathtt{sk}$ *and randomness* ρ *as input, and outputs the first message* a_1 *and state* $\mathtt{state}_1$*.*
- $\mathsf{P}_2(c_1, \mathtt{state}_1) \to (a_2, \mathtt{state}_2)$*: a second prover algorithm that takes the first challenge* $c_1 \in \mathcal{C}_1$ *and state* $\mathtt{state}_1$ *as input, and outputs the second message* a_2 *and state* $\mathtt{state}_2$*.*
- $\mathsf{P}_3(c_2, \mathtt{state}_2) \to z/\bot$*: a third prover algorithm that takes the second challenge* $c_2 \in \mathcal{C}_2$ *and state* $\mathtt{state}_2$ *as input, and outputs the last message* z *or* $\bot$*.*
- $\mathsf{V}(\mathtt{vk}, a_1, c_1, a_2, c_2, z) \to b \in \{0, 1\}$*: a verification algorithm that takes verification key* $\mathtt{vk}$ *and the transcript* a_1, c_1, a_2, c_2, z *as input and outputs its decision* 1 *(true) or* 0 *(false).*

We assume perfect correctness; a verifier always outputs 1 *for an arbitrary honestly-generated key and transcript.*

The 3-pass ID is defined in the similar way. A 3-pass identification scheme $\mathsf{ID3}$ consists of the following tuple of PPT algorithms $(\mathsf{Gen}, \mathsf{P}_1, \mathcal{C}, \mathsf{P}_2, \mathsf{V})$ and a transcript is denoted by (a, c, z).

2.3 PRF

We consider the following version of PRF security.

Definition 2.4. *Let* $\mathsf{PRF}\colon \mathcal{K} \times \mathcal{X} \to \mathcal{Y}$ *be a function. We define its advantage as*

$$\mathsf{Adv}^{\text{prf}}_{\mathsf{PRF},\mathcal{A}}(\lambda) \coloneqq \left|\Pr_{\mathsf{PRF},k}[\mathcal{A}^{|\mathsf{PRF}\rangle,\mathsf{PRF}(k,\cdot)}() = 1] - \Pr_{\mathsf{PRF},f}[\mathcal{A}^{|\mathsf{PRF}\rangle,f(\cdot)}() = 1]\right|.$$

Note that we sometimes consider a joint security of PRFs, which share the key. For example we will require pseudorandomness of $(\mathsf{PRF}_{\mathsf{share}}(k, x_1), \mathsf{H}_3(k, x_2))$.

```
Game Repro_b                                  Reprogram(D)
1 : $O_0 \leftarrow \mathsf{Func}(\mathcal{X}_1, \mathcal{Y})$               1 : $(x, \mathsf{side}) \leftarrow D$
2 : $O_1 \coloneqq O_0$                                  2 : $y \leftarrow \mathcal{Y}$
3 : $b' \leftarrow \mathcal{A}^{|O_b\rangle, \textsc{Reprogram}}()$             3 : $O_1 \coloneqq O_1[x \mapsto y]$
4 : return $b'$                                 4 : return $(x, \mathsf{side})$

Game BRepro_b                                 BReprogram(D)
1 : $O_0 \leftarrow \mathsf{Func}(\mathcal{X}, \mathcal{Y})$                1 : $(\mathcal{R}, \mathsf{side}) \leftarrow D$
2 : $O_1 \coloneqq O_0$                                  2 : for $x \in \mathcal{R}$ :
3 : $b' \leftarrow \mathcal{A}^{|O_b\rangle, \textsc{BReprogram}}()$            3 :     $y \leftarrow \mathcal{Y}$
4 : return $b'$                                 4 :     $O_1 \coloneqq O_1[x \mapsto y]$
                                                5 : return $(\mathcal{R}, \mathsf{side})$

Game SRepro_b                                 SReprogram(D)
1 : $O_0 \leftarrow \mathsf{Func}(\mathcal{X} \times \mathbb{Z}_+, \mathcal{Y})$         1 : $(x, B, \mathsf{side}) \leftarrow D$
2 : $O_1 \coloneqq O_0$                                  2 : for $i \in [B]$ :
3 : $b' \leftarrow \mathcal{A}^{|O_b\rangle, \textsc{ReprogramM}}()$            3 :     $y \leftarrow \mathcal{Y}$
4 : return $b'$                                 4 :     $O_1 \coloneqq O_1[(x, i) \mapsto y]$
                                                5 : return $(x, \mathsf{side})$
```

Fig. 2 Adaptive reprogramming games.

3. Adaptive Multiple-Point Reprogramming

Adaptive Reprogramming: Grilo et al. showed that one cannot distinguish whether the random oracle is reprogrammed or not if the min-entropy of the reprogrammed point is sufficiently high [GHHM21].

Lemma 3.1 ([GHHM21], Thm.1)**.** *Let* $\mathcal{X}_1$*,* $\mathcal{X}_2$*, and* $\mathcal{Y}$ *be finite sets. Let* $\mathcal{A}$ *be an adversary that makes* R *queries to* Reprogram *and* q *quantum queries to* $|O_b\rangle$*. Then, the distinguishing advantage of* $\mathcal{A}$ *is bounded by*

$$|\Pr[\mathsf{Repro}_0 = 1] - \Pr[\mathsf{Repro}_1 = 1]| \le \frac{3}{2} \sum_{i \in [R]} \sqrt{q \cdot p^{(i)}_{\mathsf{max}}},$$

where Repro_b *and* Reprogram *are defined in Figure 2, where, in the i-th query, the adversary* $\mathcal{A}$ *samples from a distribution* D_i*, and we define* $p^{(i)}_{\mathsf{max}} = \mathbb{E} \max_{\hat{x}} \Pr_{(x,\mathsf{side}) \sim D_i}[\hat{x} = x]$*, with the expectation taken over* $\mathcal{A}$*'s behavior up to the i-th query.*

Extension to multiple-point reprogramming: To treat grinding and rejection sampling, where the random oracle will be reprogrammed on multiple points in one query, we extend the adaptive reprogramming technique as follows:

Theorem 3.1 (Batched Adaptive Reprogramming)**.** *Let* $\mathcal{X}_1$*,* $\mathcal{X}_2$*, and* $\mathcal{Y}$ *be finite sets. Let* $\mathcal{A}$ *be an adversary that makes* R *queries to* Reprogram *and* q *quantum queries to* $|O_b\rangle$*. Then, the distinguishing advantage of* $\mathcal{A}$ *is bounded by*

$$|\Pr[\mathsf{BRepro}_0 = 1] - \Pr[\mathsf{BRepro}_1 = 1]| \le \frac{3}{2} \sum_{i \in [R]} \sqrt{q \cdot p^{(i)}_{\mathsf{max}}},$$

where BRepro_b *and* BReprogram *are defined in Figure 2, where, in the i-th query, the adversary* $\mathcal{A}$ *samples from a distribution* D_i*, and we define* $p^{(i)}_{\mathsf{max}} = \mathbb{E} \max_{\hat{x}} \Pr_{(\mathcal{R},\mathsf{side}) \sim D_i}[\hat{x} \in \mathcal{R}]$*, with the expectation taken over* $\mathcal{A}$*'s behavior up to the i-th query.*

The proof appears in the full version.

We consider a special case of BRepro_b defined in Figure 2. The following corollary follows from Theorem 3.1 by supposing that $\mathcal{R} = \{(x, i) : i \in [B]\}$ is sampled in $\mathsf{SReprogramm}_b$.

Corollary 3.1 (Sequential Adaptive Reprogramming)**.** *Let $\mathcal{X}$ and $\mathcal{Y}$ be finite sets. Let $\mathcal{A}$ be an adversary that makes R queries to* SREPROGRAM *and q quantum queries to $|\mathrm{O}_b\rangle$. Then, the distinguishing advantage of $\mathcal{A}$ is bounded by*

$$|\Pr[\mathsf{SRepro}_0 = 1] - \Pr[\mathsf{SRepro}_1 = 1]| \le \frac{3}{2}\sum_{i\in[R]}\sqrt{q \cdot p^{(i)}_{\mathsf{max}}},$$

where SRepro_b and SREPROGRAM *are defined in Figure 2, where, in the i-th query, the adversary $\mathcal{A}$ samples from a distribution D_i, and we define $p^{(i)}_{\mathsf{max}} = \mathbb{E}\max_{\hat{x}}\Pr_{(x,B,\mathsf{side})\sim D_i}[\hat{x} = x]$, with the expectation taken over $\mathcal{A}$'s behavior up to the i-th query.*

4. Mirath

We review the second version of Mirath [AAB+24], whose security is based on the MinRank Syndrome problem equivalent to the MinRank problem (see, [BFG+24]). The computational MinRank Syndrome problem is defined as follows:

Definition 4.1 (MinRank Syndrome problem)**.** *Let q, m, n, k, and r be positive integers. Let $\boldsymbol{H} := [\boldsymbol{I}_{mn-k} \mid \boldsymbol{H}'] \in \mathrm{GF}(q)^{(mn-k)\times mn}$, where $\boldsymbol{H}' \leftarrow \mathrm{GF}(q)^{(mn-k)\times k}$. Let $\boldsymbol{y} \in \mathrm{GF}(q)^{mn-k}$. The computational MinRank Syndrome Problem $\mathsf{MinRankSynd}(q, m, n, k, r)$ asks, given $\boldsymbol{H}'$ and $\boldsymbol{y}$, to find $\boldsymbol{E}$ such that $\boldsymbol{H}\cdot\mathsf{vec}(\boldsymbol{E}) = \boldsymbol{y}$ and $\mathrm{rank}(\boldsymbol{E}) \le r$.*

Bidoux et al. [BFG+24] proposed the dual support modeling, where $\boldsymbol{E}$ is a product of two matrices $\boldsymbol{E} = \boldsymbol{SC}$ with $\boldsymbol{S} \in \mathrm{GF}(q)^{m\times r}$ and $\boldsymbol{C} \in \mathrm{GF}(q)^{r\times n}$, which assures $\mathrm{rank}(\boldsymbol{E}) \le r$. By using this idea, Mirath's key generation algorithm Gen is defined as follows: Choose random two seeds $\mathsf{seed}_{\mathsf{sk}}, \mathsf{seed}_{\mathsf{vk}} \leftarrow \{0,1\}^\lambda$ and expand them as $(\boldsymbol{S}, \boldsymbol{C}') \in \mathrm{GF}(q)^{m\times r}\times\mathrm{GF}(q)^{r\times(n-r)}$ and $\boldsymbol{H}' \in \mathrm{GF}(q)^{(mn-k)\times k}$. The signing key is $(\boldsymbol{S}, \boldsymbol{C}')$ and the verification key consists of $\boldsymbol{H}'$ and $\boldsymbol{y} \in \mathrm{GF}(q)^{mn-k}$, where

$$\boldsymbol{y} := \boldsymbol{H}\cdot\mathsf{vec}(\boldsymbol{SC}) \text{ with } \boldsymbol{H} = [\boldsymbol{I}_{mn-k} \mid \boldsymbol{H}'] \text{ and } \boldsymbol{C} = [\boldsymbol{I}_r\boldsymbol{C}'].$$

The signature scheme is obtained by applying a variant of the Fiat-Shamir transform to a 5-round TC-in-the-Head ID protocol that shows the relation between verification key and signing key.

The underlying 5-round ID protocol: Let us define the 5-round ID protocol $\mathsf{ID5}_{\mathsf{Mirath}}$ as follows, where we follow the 5-round TCitH protocol in the PIOP formalism (as in Section 3.3. of Mirath's specification) with some concrete instantiations. To treat the protocol as a *commit-and-open* protocol, a challenge will be considered as a subset of the challenge space. Recall that $\boldsymbol{S} \in \mathrm{GF}(q)^{m\times r}$ and $\boldsymbol{C}' \in \mathrm{GF}(q)^{r\times(n-r)}$. For ease of notation, we drop "$^{(e)}$" from superscripts. We require an additional positive integer ρ for parameter. Let $\phi\colon [N] \to S \subseteq \mathrm{GF}(q^\mu)$ be a public one-to-one function. We treat random oracles $\mathsf{H}_1\colon \{0,1\}^* \to \mathcal{Y}$, $\mathsf{H}_2\colon \{0,1\}^* \to \mathcal{Y}$, $\mathsf{H}_3\colon \{0,1\}^* \to \mathcal{Y}$, $\mathsf{H}'_3\colon \{0,1\}^* \to \mathcal{Y}$, $\mathsf{XOF}_1\colon \mathcal{Y} \to \mathcal{C}_1$, $\mathsf{XOF}_2\colon \{0,1\}^* \to \mathcal{C}_2 \times \{0,1\}^w$, where $\mathcal{Y} = \{0,1\}^{2\lambda}$, $\mathcal{C}_1 = \mathrm{GF}(q^\mu)^{\rho\times(mn-k)}$, and $\mathcal{C}_2 = [N]^\tau$.

(1) $\mathsf{P}_1(\mathtt{sk} = (\boldsymbol{S}, \boldsymbol{C}'); \mathtt{salt}, \mathtt{rseed}) \to (a_1, \mathtt{state}_1)$: The first prover samples $P_S(X) = \boldsymbol{S}\cdot X + \boldsymbol{S}_{\mathsf{base}}$ and $P_{C'}(X) = \boldsymbol{C}'\cdot X + \boldsymbol{C}'_{\mathsf{base}}$, where $\boldsymbol{S}_{\mathsf{base}} \leftarrow \mathrm{GF}(q^\mu)^{m\times r}$ and $\boldsymbol{C}'_{\mathsf{base}} \leftarrow \mathrm{GF}(q^\mu)^{r\times(n-r)}$. It also samples $P_v(X) = \boldsymbol{v}\cdot X + \boldsymbol{v}_{\mathsf{base}}$, where $\boldsymbol{v}, \boldsymbol{v}_{\mathsf{base}} \leftarrow \mathrm{GF}(q^\mu)^\rho$. It generates the first message as the commitment of those polynomials
In Mirath, the prover commits those polynomials as follows: It first calls the batched all-but-one vector commitment (BAVC) algorithm on input $\mathtt{salt}$ and $\mathtt{rseed}$, obtains the seeds $\mathtt{seed} = \{\mathtt{seed}_i\}_{i\in[N]}$, its commitment $h_{\mathsf{com}} = \mathsf{H}_3(\mathtt{com})$, where $\mathtt{com}_i$ is a commitment of $\mathtt{seed}_i$ via H'_3, and secret information $\mathtt{key} = (\mathtt{tree}, \mathtt{com})$ for decommitment. Using those seeds, it computes secret shares $(\boldsymbol{S}_{\mathtt{rnd},i}, \boldsymbol{C}'_{\mathtt{rnd},i}, \boldsymbol{v}_{\mathtt{rnd},i}) := \mathsf{PRF}_{\mathsf{share}}(\mathtt{salt}, \mathtt{seed}_i)$. It then computes $(\boldsymbol{S}_{\mathsf{acc}}, \boldsymbol{C}'_{\mathsf{acc}}, \boldsymbol{v}_{\mathsf{acc}}) := \sum_{i\in[N]}(\boldsymbol{S}_{\mathtt{rnd},i}, \boldsymbol{C}'_{\mathtt{rnd},i}, \boldsymbol{v}_{\mathtt{rnd},i})$ and $\mathtt{base} = (\boldsymbol{S}_{\mathsf{base}}, \boldsymbol{C}'_{\mathsf{base}}, \boldsymbol{v}_{\mathsf{base}}) := \sum_{i\in[N]}\phi(i)\cdot(\boldsymbol{S}_{\mathtt{rnd},i}, \boldsymbol{C}'_{\mathtt{rnd},i}, \boldsymbol{v}_{\mathtt{rnd},i})$. It then computes the offset information $\mathtt{aux} := (\boldsymbol{S}, \boldsymbol{C}') - (\boldsymbol{S}_{\mathsf{acc}}, \boldsymbol{C}'_{\mathsf{acc}})$. It also sets $\boldsymbol{v} := \boldsymbol{v}_{\mathsf{acc}}$. It finally computes $h_1 := \mathsf{H}_1(\mathtt{salt}, h_{\mathsf{com}}, \mathtt{aux})$. It sends $a_1 := (\mathtt{salt}, h_1, \mathtt{aux})$ as the commitment of the polynomials and keeps $\mathtt{state}_1 := (\mathtt{base}, \boldsymbol{v}, \mathtt{key})$, where $\boldsymbol{v}$ and $\mathtt{base}$ with $\boldsymbol{S}$ and $\boldsymbol{C}'$ represent P_S, $P_{C'}$, and P_v.

(2) The first challenge: The verifier sends a random challenge $\boldsymbol{\Gamma} \leftarrow \mathrm{GF}(q^\mu)^{\rho\times(mn-k)}$, which is shared among τ parallel repetitions.

(3) $\mathsf{P}_2(\mathtt{sk}, c_1 = \boldsymbol{\Gamma}, \mathtt{state}_1) \to (a_2, \mathtt{state}_2)$: The second prover computes a polynomial $P_\alpha(X) = \boldsymbol{\alpha}_{\mathtt{mid}}X + \boldsymbol{\alpha}_{\mathtt{base}}$ using $\boldsymbol{\Gamma}$ and sends $a_2 = (\boldsymbol{\alpha}_{\mathtt{mid}}, \boldsymbol{\alpha}_{\mathtt{base}})$ as the second message and keeps $\mathtt{state}_2 = \mathtt{key}$. In Mirath, it computes

$$P_\alpha(X) := P_v(X) + \boldsymbol{\Gamma}\cdot(\boldsymbol{H}\cdot\mathsf{vec}(P_E(X)) - \boldsymbol{y}\cdot X^2),$$

which consists of ρ polynomials in $\mathrm{GF}(q^\mu)[X]$, with $P_E(X) := P_S(X)\cdot[P_I(X) \mid P_{C'}(X)] \in (\mathrm{GF}(q^\mu)[X])^{m\times n}$, where $P_I(X) = \boldsymbol{I}_r\cdot X \in (\mathrm{GF}(q^\mu)[X])^{r\times r}$.

(4) The second challenge: The verifier sends a random challenge $s \in S = \{\phi(i)\}_{i\in[N]} \subseteq \mathrm{GF}(q^\mu)$. Concretely speaking, it sends $i^* \leftarrow [N]$.

(5) $\mathsf{P}_3(\mathtt{state}_2) \to z$: The third prover reveals $(\boldsymbol{S}_{\mathtt{eval}}, \boldsymbol{C}'_{\mathtt{eval}}, \boldsymbol{v}_{\mathtt{eval}}) = (P_S(s), P_{C'}(s), P_v(s))$ and sends the proof π that shows the consistency of the polynomials with their commitments.
In Mirath, the prover reveals seeds $\{\mathtt{seed}_i\}_{i\ne i^*}$ and $\mathtt{com}_{i^*}$, which allows the verifier to compute $(\boldsymbol{S}_{\mathtt{eval}}, \boldsymbol{C}'_{\mathtt{eval}}, \boldsymbol{v}_{\mathtt{eval}})$ from $\mathtt{seed}_i$ and $\mathtt{aux}$. The final response is $z := (\{\mathtt{seed}_i\}_{i\ne i^*}, \mathtt{com}_{i^*})$. In the τ-parallel version, the challenge will be written as a subset $c \subseteq [N]^\tau$, which is represented by $\mathbf{i}^* \in [N]^\tau$, and we will denote $\mathtt{seed}_c := \{\mathtt{seed}_i\}_{i\in c}$ and $\mathtt{com}_{-c} := \{\mathtt{com}_i\}_{i\notin c} = \{\mathtt{com}_i\}_{i\in\mathbf{i}^*}$ in the EUF-NMA security proof.

(6) $\mathsf{V}(\mathtt{vk}, a_1, c_1, a_2, c_2, z) \to v$: The verifier checks the con-

sistency of π and checks if

$$P_\alpha(s) = \boldsymbol{\alpha}_{\mathtt{eval}} \coloneqq \boldsymbol{v}_{\mathtt{eval}} + \boldsymbol{\Gamma} \cdot (\boldsymbol{H} \cdot \mathsf{vec}(\boldsymbol{E}_{\mathtt{eval}}) - \boldsymbol{y} \cdot s^2) \quad (1)$$

with $\boldsymbol{E}_{\mathtt{eval}} = \boldsymbol{S}_{\mathtt{eval}} \cdot [s\boldsymbol{I}_r \mid \boldsymbol{C}'_{\mathtt{eval}}]$.

In Mirath, the verifier receives $\mathtt{salt}$, $\mathtt{aux}$, $\{\mathtt{seed}_i\}_{i \neq i^*}$, and $\mathtt{com}_{i^*}$. It first computes $(\boldsymbol{S}_{\mathtt{eval}}, \boldsymbol{C}'_{\mathtt{eval}}, \boldsymbol{v}_{\mathtt{eval}})$, which are $(P_S(s), P_{C'}(s), P_v(s))$ with $s = \phi(i^*)$, from $\mathtt{salt}$, $\mathtt{seed}_i$, and $\mathtt{aux}$; it computes $(\boldsymbol{S}_{\mathtt{rnd},i}, \boldsymbol{C}'_{\mathtt{rnd},i}, \boldsymbol{v}_{\mathtt{rnd},i}) \coloneqq \mathsf{PRF}_{\mathsf{share}}(\mathtt{salt}, \mathtt{seed}_i)$ and computes $(\boldsymbol{S}_{\mathtt{eval}}, \boldsymbol{C}'_{\mathtt{eval}}, \boldsymbol{v}_{\mathtt{eval}}) \coloneqq \phi(i^*) \cdot (\boldsymbol{S}_{\mathtt{aux}}, \boldsymbol{C}'_{\mathtt{aux}}, \boldsymbol{0}) + \sum_{i \neq i^*} (\phi(i^*) - \phi(i)) \cdot (\boldsymbol{S}_{\mathtt{rnd},i}, \boldsymbol{C}'_{\mathtt{rnd},i}, \boldsymbol{v}_{\mathtt{rnd},i})$. It then checks the equation 1. If the equation holds, then it finally computes the (batched) all-but-one vector commitment $h_{\mathtt{com}}$ from $\{\mathtt{seed}_i\}_{i \neq i^*}$, $\mathtt{com}_{i^*}$, and $\mathtt{salt}$, and checks if $h_1 = \mathsf{H}_1(\mathtt{salt}, h_{\mathtt{com}}, \mathtt{aux})$. If it holds, then outputs 1; outputs 0 otherwise.

The collapsed 3-round ID protocol: We next consider the collapsed 3-round ID protocol $\mathsf{ID3} = (\mathsf{Gen}, \tilde{\mathsf{P}}_1, \tilde{\mathcal{C}}, \tilde{\mathsf{P}}_2, \tilde{\mathsf{V}})$ defined as follows:

(1) $\tilde{\mathsf{P}}_1(\mathtt{sk}; \mathtt{salt}, \mathtt{rseed}) \to (a, \mathtt{state})$: The first prover runs $(a_1 = (\mathtt{salt}, h_1, \mathtt{aux}), \mathtt{state}_1) \leftarrow \mathsf{P}_1(\mathtt{sk}; \rho = (\mathtt{salt}, \mathtt{rseed}))$, computes $c_1 = \boldsymbol{\Gamma} \coloneqq \mathsf{XOF}_1(h_1)$, and runs $(a_2 = (\boldsymbol{\alpha}_{\mathtt{mid}}, \boldsymbol{\alpha}_{\mathtt{base}}), \mathtt{state}_2 = \mathtt{key}) \leftarrow \mathsf{P}_2(\mathtt{sk}, c_1, \mathtt{state}_1)$. Output $a = (a_1, a_2)$ and $\mathtt{state} = \mathtt{key}$.

(2) The challenge: The verifier chooses a random challenge $c \subseteq \tilde{\mathcal{C}} = [N]^\tau$ represented as $\mathbf{i}^* \in [N]^\tau$ and sends it.

(3) $\tilde{\mathsf{P}}_2(c, \mathtt{state}) \to z$: The second prover runs $z \leftarrow \mathsf{P}_3(\mathtt{state}, c)$ and outputs $z = (\mathtt{seed}_c, \mathtt{com}_{-c})$ as a response.

(4) $\tilde{\mathsf{V}}(\mathtt{vk}, a = (a_1, a_2), c, z) \to v$: The verifier computes $\boldsymbol{\Gamma} \coloneqq \mathsf{XOF}_1(h_1)$ and output $v \leftarrow \mathsf{V}(\mathtt{vk}, a_1, c_1 = \boldsymbol{\Gamma}, a_2, c_2 = c, z)$.

We then define a commitment-reproducing algorithm Rep for this 3-round ID protocol and replace the verification algorithm as follows:

(1) Rep takes $\mathtt{vk}$, c, and $z' = (\mathtt{salt}, \mathtt{aux}, \boldsymbol{\alpha}_{\mathtt{mid}}, \mathtt{seed}_c, \mathtt{com}_{-c})$.

(2) (Re-compute BAVC:) Compute $\mathtt{com}_c$ from $\mathtt{salt}$ and $\mathtt{seed}_c$ and $h_{\mathtt{com}} = \mathsf{H}_3(\mathtt{com})$.

(3) (Re-compute h_1:) Compute $h'_1 = \mathsf{H}_1(\mathtt{salt}, h_{\mathtt{com}}, \mathtt{aux})$.

(4) (Re-compute c_1:) Compute $\boldsymbol{\Gamma}' = \mathsf{XOF}_1(h'_1)$.

(5) (Re-compute $\boldsymbol{\alpha}_{\mathtt{base}}$:) Compute $\boldsymbol{S}_{\mathtt{eval}}, \boldsymbol{C}'_{\mathtt{eval}}$, and $\boldsymbol{v}_{\mathtt{eval}}$. For each $e \in [\tau]$, compute $\boldsymbol{\alpha}'_{\mathtt{base}}[e] \coloneqq \boldsymbol{\alpha}_{\mathtt{eval}}[e] - \boldsymbol{\alpha}_{\mathtt{mid}}[e] \cdot \phi(\mathbf{i}^*[e])$, where $\boldsymbol{\alpha}'_{\mathtt{eval}}[e]$ is defined in the equation 1. Output $a' = (\mathtt{salt}, h'_1, \boldsymbol{\alpha}_{\mathtt{mid}}, \boldsymbol{\alpha}'_{\mathtt{base}})$.

Using this reproducing algorithm, we consider a variant of the collapsed 3-round commit-and-open protocol $\mathsf{ID3}' = (\mathsf{Gen}, \tilde{\mathsf{P}}'_1, \tilde{\mathcal{C}}, \tilde{\mathsf{P}}'_2, \tilde{\mathsf{V}}')$ defined as follows:

(1) $\tilde{\mathsf{P}}'_1$: The first prover outputs $a' = (\mathtt{salt}, h_1, \boldsymbol{\alpha}_{\mathtt{mid}}, \boldsymbol{\alpha}_{\mathtt{base}})$ out of $a = (\mathtt{salt}, h_1, \mathtt{aux}, \boldsymbol{\alpha}_{\mathtt{mid}}, \boldsymbol{\alpha}_{\mathtt{base}})$.

(2) $\tilde{\mathcal{C}}$: The challenge space is the same as that of $\mathsf{ID3}$.

(3) $\tilde{\mathsf{P}}'_2$: The second prover outputs $z' = (\mathtt{salt}, \mathtt{aux}, \boldsymbol{\alpha}_{\mathtt{mid}}, \mathtt{seed}_c, \mathtt{com}_{-c})$ from a and $z = (\mathtt{seed}_c, \mathtt{com}_{-c})$.

(4) $\tilde{\mathsf{V}}'$: It computes $a'' = (\mathtt{salt}, h'_1, \boldsymbol{\alpha}_{\mathtt{mid}}, \boldsymbol{\alpha}'_{\mathtt{base}}) \coloneqq \mathsf{Rep}(\mathtt{vk}, c, z')$ and checks if $a' = a''$.

We then define two variants of Mirath, which we denote $\widetilde{\mathsf{Mirath}} = \mathsf{FS}_g[\widetilde{\mathsf{ID3}}, \mathsf{H}_2, \mathsf{XOF}_2]$ and $\mathsf{Mirath}' = \mathsf{FS}_g[\mathsf{ID3}', \mathsf{H}_2, \mathsf{XOF}_2]$, and Mirath as follows:

- $\widetilde{\mathsf{Mirath}}$: The signing algorithm $\widetilde{\mathsf{Sign}}$, on input $\mathtt{sk}$ and $\mathtt{msg}$, first chooses $\mathtt{salt}$ and $\mathtt{rseed}$ uniformly at random. It computes $(a, \mathtt{state}) \coloneqq \tilde{\mathsf{P}}_1(\mathtt{sk}; \mathtt{salt}, \mathtt{rseed})$, $h_2 \coloneqq \mathsf{H}_2(\mathtt{vk}, \mathtt{salt}, \mathtt{msg}, h_1, \boldsymbol{\alpha}_{\mathtt{mid}}, \boldsymbol{\alpha}_{\mathtt{base}})$, which we will denote $\mathsf{H}_2(\mathtt{vk}, \mathtt{msg}, a')$ with $a' = (\mathtt{salt}, h_1, \boldsymbol{\alpha}_{\mathtt{mid}}, \boldsymbol{\alpha}_{\mathtt{base}})$. Let $\mathtt{ctr} \coloneqq 0$. It computes $(\mathbf{i}^*, v_{\text{grinding}}) \coloneqq \mathsf{XOF}_2(h_2, \mathtt{ctr})$ and computes $z \coloneqq \tilde{\mathsf{P}}_2(c, \mathtt{state})$, where c is computed from $\mathbf{i}^*$; If $v_{\text{grinding}} = 0^w$, then it outputs $\sigma \coloneqq (a, \mathtt{ctr}, z)$ as a signature; if not, then it increments $\mathtt{ctr}$ and retries to obtain a signature.
  The verification algorithm $\widetilde{\mathsf{Vrfy}}$ takes $\mathtt{vk}$, $\mathtt{msg}$, and $\tilde{\sigma} = (a, \mathtt{ctr}, z)$ as input. It computes $h_2 \coloneqq \mathsf{H}_2(\mathtt{vk}, \mathtt{msg}, a')$ and $(\mathbf{i}^*, v_{\text{grinding}}) \coloneqq \mathsf{XOF}_2(h_2, \mathtt{ctr})$. It outputs 1 if $\tilde{\mathsf{V}}(\mathtt{vk}, a, c, z) = 1$ and $v_{\text{grinding}} = 0^w$; outputs 0 otherwise.
- Mirath': The signing algorithm Sign', on input $\mathtt{sk}$ and $\mathtt{msg}$, first chooses $\mathtt{salt}$ and $\mathtt{rseed}$ uniformly at random. It computes $(a', \mathtt{state}) \coloneqq \tilde{\mathsf{P}}'_1(\mathtt{sk}; \mathtt{salt}, \mathtt{rseed})$, where $a' = (\mathtt{salt}, h_1, \boldsymbol{\alpha}_{\mathtt{mid}}, \boldsymbol{\alpha}_{\mathtt{base}})$ and $h_2 \coloneqq \mathsf{H}_2(\mathtt{vk}, \mathtt{msg}, a')$. Let $\mathtt{ctr} \coloneqq 0$. It computes $(\mathbf{i}^*, v_{\text{grinding}}) \coloneqq \mathsf{XOF}_2(h_2, \mathtt{ctr})$ and computes $z' = (\mathtt{salt}, \mathtt{aux}, \boldsymbol{\alpha}_{\mathtt{mid}}, \mathtt{seed}_c, \mathtt{com}_{-c}) \coloneqq \tilde{\mathsf{P}}'_2(c, \mathtt{state})$, where c is computed from $\mathbf{i}^*$; If $v_{\text{grinding}} = 0^w$, then it outputs $\sigma \coloneqq (a', \mathtt{ctr}, z')$ as a signature; if not, then it increments $\mathtt{ctr}$ and retries to obtain a signature.
  The verification algorithm Vrfy' takes $\mathtt{vk}$, $\mathtt{msg}$, and $\sigma' = (a', \mathtt{ctr}, z')$ as input. It computes $h_2 \coloneqq \mathsf{H}_2(\mathtt{vk}, \mathtt{msg}, a')$, $(\mathbf{i}^*, v_{\text{grinding}}) \coloneqq \mathsf{XOF}_2(h_2, \mathtt{ctr})$, and $a'' \coloneqq \mathsf{Rep}(\mathtt{vk}, c, z')$. It outputs 1 if $a' = a''$ and $v_{\text{grinding}} = 0^w$; outputs 0 otherwise.
- Mirath: The signing algorithm Sign, on input $\mathtt{sk}$ and $\mathtt{msg}$, first chooses $\mathtt{salt}$ and $\mathtt{rseed}$ uniformly at random. It computes $(a', \mathtt{state}) \coloneqq \tilde{\mathsf{P}}'_1(\mathtt{sk}; \mathtt{salt}, \mathtt{rseed})$ and $h_2 \coloneqq \mathsf{H}_2(\mathtt{vk}, \mathtt{msg}, a')$. Let $\mathtt{ctr} \coloneqq 0$. It computes $(\mathbf{i}^*, v_{\text{grinding}}) \coloneqq \mathsf{XOF}_2(h_2, \mathtt{ctr})$ and computes $z' = (\mathtt{salt}, \mathtt{aux}, \mathtt{seed}_c, \mathtt{com}_{-c}) \coloneqq \tilde{\mathsf{P}}'_2(c, \mathtt{state})$, where c is computed from $\mathbf{i}^*$; If $v_{\text{grinding}} = 0^w$ and $\pi_{\text{BAVC}} = \mathsf{compress}(c, \mathtt{seed}_c, \mathtt{com}_{-c})$ is shorter than the threshold T_{open}, then it outputs $\sigma \coloneqq (h_2, \mathtt{ctr}, \mathtt{salt}, \mathtt{aux}, \boldsymbol{\alpha}_{\mathtt{mid}}, \pi_{\text{BAVC}})$ as a signature; if not, then it increments $\mathtt{ctr}$ and retry to obtain the signature.
  The verification algorithm Vrfy takes $\mathtt{vk}$, $\mathtt{msg}$, and $\sigma = (h_2, \mathtt{ctr}, \mathtt{salt}, \mathtt{aux}, \boldsymbol{\alpha}_{\mathtt{mid}}, \pi_{\text{BAVC}})$ as input. It first computes $(\mathbf{i}^*, v_{\text{grinding}}) \coloneqq \mathsf{XOF}_2(h_2, \mathtt{ctr})$ and decompress π_{BAVC} into $(\mathtt{seed}_c, \mathtt{com}_{-c})$; if cannot, outputs 0. It then computes $a' \coloneqq \mathsf{Rep}(\mathtt{vk}, c, z')$ and $h'_2 \coloneqq \mathsf{H}_2(\mathtt{vk}, \mathtt{msg}, a')$. It outputs 1 if $h_2 = h'_2$ and $v_{\text{grinding}} = 0^w$; outputs 0 otherwise.

By a technical reason, we will consider the EUF-NMA security of $\widetilde{\mathsf{Mirath}}$ instead of Mirath. To treat it formally, we show the following theorem.

Theorem 4.1. *If* $\widetilde{\mathsf{Mirath}}$ *is EUF-NMA-secure, then* Mirath' *is EUF-NMA-secure. If* Mirath' *is EUF-NMA-secure, then* Mirath *is EUF-NMA-secure.*

In other words, if there exists an adversary $\mathcal{A}$ against Mirath *who makes Q_X queries to the oracle X, then there exists $\tilde{\mathcal{A}}$ against* $\widetilde{\mathsf{Mirath}}$ *who makes $\tilde{Q}_X$ queries to the oracle X satisfying*

$$\mathsf{Adv}^{\text{euf-nma}}_{\mathsf{Mirath},\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{\text{euf-nma}}_{\widetilde{\mathsf{Mirath}},\tilde{\mathcal{A}}}(\lambda).$$

The running time of $\tilde{\mathcal{A}}$ is about that of $\mathcal{A}$ plus a verification time. We also have $\tilde{Q}_X = Q_X + 1$ for all but H_3' and $\tilde{Q}_{\mathsf{H}_3'} = Q_{\mathsf{H}_3'} + N$.

The proof appears in the full version.

5. EUF-NMA Security of Mirath

We give a bound for the EUF-NMA advantage against Mirath. Correclty speaking, we show the EUF-NMA advantage against $\widetilde{\mathsf{Mirath}}$.

5.1 Main theorem

First of all, we explain the strategy to give a bound on the EUF-NMA advantage by following the previous papers [DFMS22] and [AHJ+23]. (We mix the notations of them.):

We define two relations:

$$R_{\mathsf{Gen}} := \left\{ \begin{matrix} (\mathtt{vk} = (\boldsymbol{H}', \boldsymbol{y}), \mathtt{sk} = (\boldsymbol{S}, \boldsymbol{C}')) \mid \\ [\boldsymbol{I} \mid \boldsymbol{H}'] \cdot \mathsf{vec}([\boldsymbol{S} \mid \boldsymbol{S}\boldsymbol{C}']) - \boldsymbol{y} = \boldsymbol{0} \end{matrix} \right\},$$

$$R_{\boldsymbol{\Gamma}} := \left\{ \begin{matrix} (\mathtt{vk} = (\boldsymbol{H}', \boldsymbol{y}), \mathtt{sk} = (\boldsymbol{S}, \boldsymbol{C}')) \mid \\ \boldsymbol{\Gamma} \cdot \big([\boldsymbol{I} \mid \boldsymbol{H}'] \cdot \mathsf{vec}([\boldsymbol{S} \mid \boldsymbol{S}\boldsymbol{C}']) - \boldsymbol{y}\big) = \boldsymbol{0} \end{matrix} \right\},$$

where $R_{\boldsymbol{\Gamma}}$ depends on $\boldsymbol{\Gamma} \in \mathcal{C}_1$. We note that $R_{\mathsf{Gen}} \subseteq R_{\boldsymbol{\Gamma}}$ for any $\boldsymbol{\Gamma} \in \mathcal{C}_1$.

Now, let us consider a quantum adversary $\mathcal{A}_{\text{nma}}$ who will break the EUF-NMA security in the QROM (or QROM+/eQROM). We additionally consider an online extractor Ext for the non-interactive protocol/signature and an extractor Ext' for the collapsed 3-round protocol, defined later. We consider the following strategy, which roughly follows Thoerem 6 of [AHJ+23] and Theorem 5.2 of [DFMS22]:

We consider the following QROM+ adversary $\mathcal{A}_{\text{snd}} = (\mathcal{A}_{\text{snd},0}, \mathcal{A}_{\text{snd},1})$ (without giving a predicate) against an online extractor: $\mathcal{A}_{\text{snd},0}$ runs the NMA adversary $\mathcal{A}_{\text{nma}}$ on input $\mathtt{vk}$ and obtains $\mathtt{msg}$ and $\sigma = (a_1, a_2, \pi_{\text{BAVC}}, \mathtt{ctr})$ with the compressed random oracles H_1, H_2, H_3, H_3', and XOF_2 and with another random oracle XOF_1; it runs the verification algorithm Vrfy on input $\mathtt{vk}$, $\mathtt{msg}$, and σ with the compressed random oracles (note that Vrfy internally decompress π_{BAVC} into $z = (\mathtt{seed}_c, \mathtt{com}_{-c})$); we then measure the database for H_1, H_2, H_3, H_3', and XOF_2 of the compressed random oracles; $\mathcal{A}_{\text{snd},1}$ receives the forgery and the measured database; it runs the inverter for the Merkle commitment h_1 with the measured database and obtains $\mathtt{seed} \in (\{0,1\}^\lambda \cup \{\perp\})^{\tau N}$; it then outputs $\mathtt{out} = (\mathtt{vk}, a_1, a_2, \mathtt{seed}, \mathtt{com}, \mathtt{ctr})$.

The online extractor Ext is define as follows: On input $(\mathtt{inst} = (\mathtt{vk}, \mathtt{salt}, \mathtt{msg}, a_2), \mathtt{aux}, \mathtt{seed})$, it finds e^* satisfying $\mathtt{seed}[e^*][i] \neq \perp$ for all $i \in [N]$ and outputs a witness $\mathtt{sk}' = (\boldsymbol{S}, \boldsymbol{C}')$ computed from $\mathtt{seed}[e^*]$ and $\mathtt{aux}$.

Considering the game for the online extraction error, we have the following two cases:

(1) The extractor Ext fails to extract the witness with respect to $R_{\boldsymbol{\Gamma}}$. This probability is bounded by the online extraction error of the non-interactive protocol, where we treat H_1, H_2, H_3, H_3', and XOF_2 as a monolithic oracle simulated by the compressed random oracle technique and XOF_1 is another one. [DFMS22], Thm.5.2 showed that the probability that this event happens is upper-bounded by the sum of the probabilities that the following three events happen:
- The EUF-NMA adversary makes the extractor with the measured database fail to output the witness for $R_{\boldsymbol{\Gamma}}$.
- The measured database contains a collision in H_1, H_2, H_3, H_3', and XOF_2.
- There is a difference between the verification algorithm with the (compressed) random oracle and that with the measured database: This is bounded by Corollary 2.7 of [DFMS22].

(2) Otherwise, the extractor Ext succeeds in finding the witness with respect to $R_{\boldsymbol{\Gamma}}$. However, the relation $R_{\boldsymbol{\Gamma}}$ involves $\boldsymbol{\Gamma} = \mathsf{XOF}_1(h_1)$ and is not directly related to R_{Gen} because of the collapse of the rounds. In this case, the definition of Ext implies that we can construct two transcripts sharing the commitment but different challenges $(a = (a_1, a_2), c, z, c', z')$ by using the index e^* such that $\mathtt{seed}[e^*][i] \neq \perp$ for all $i \in [N]$.
Let us consider this modification as an adversary for the collapsed 3-round ID; that is, we consider the following QROM+ adversary $\mathcal{A}'_{\text{snd}} = (\mathcal{A}'_{\text{snd},0}, \mathcal{A}'_{\text{snd},1})$: $\mathcal{A}'_{\text{snd},0}$ runs the NMA adversary $\mathcal{A}_{\text{nma}}$ on input $\mathtt{vk}$ and obtains $\mathtt{msg}$ and $\sigma = (a_1, a_2, \pi_{\text{BAVC}}, \mathtt{ctr})$ with the compressed random oracles H_1, H_2, H_3, H_3', XOF_1, and XOF_2 which are treated as a monolithic oracle (notice that including XOF_1 is not a problem here); it runs the verification algorithm Vrfy on input $\mathtt{vk}$, $\mathtt{msg}$, and σ with the compressed random oracles; we then measure the database of the compressed random oracles for H_1, H_2, H_3, H_3', XOF_1, and XOF_2, where we include H_2 and XOF_2 for consistency of $\mathcal{A}'_{\text{snd}}$; $\mathcal{A}'_{\text{snd},1}$ receives the forgery and the measured database; it runs the inverter for the Merkle commitment h_1 with the measured database and obtains $\mathtt{seed} \in (\{0,1\}^\lambda \cup \{\perp\})^{\tau N}$; it then runs Ext on input $(\mathtt{vk}, a_1, a_2, \mathtt{seed})$ and obtains the witness $(\boldsymbol{S}, \boldsymbol{C}')$ for $R_{\boldsymbol{\Gamma}}$; it then constructs two transcripts $(a, c, z = (\mathtt{seed}_c, \mathtt{com}_{-c}))$ and $(a, c', z' = (\mathtt{seed}_{c'}, \mathtt{com}_{-c'}))$ satisfying $c \neq c'$ and outputs them. We have the following two cases:
- The extractor Ext' for the collapsed 3-round ID can extract the witness from the database and the adver-

sary's output, and the witness is valid with respect to R_{Gen}. This probability is upper-bounded by the advantage against the underlying problem.

- The extractor Ext' for the collapsed 3-round ID fails to output the valid witness with respect to R_{Gen}. This probability is upper-bounded by the special soundness of the collapsed 3-round ID protocol. This advantage is divided into two pieces:
  - The extractor finds a collision of H_1, H_3, or H_3', or a collision for XOF_1.
  - The extractor fails to output a witness with respect to R_{Gen}. This happens if the adversary outputs the witness in $R_{\boldsymbol{\Gamma}} \setminus R_{\mathsf{Gen}}$, which implies the adversary cheats by exploiting $\boldsymbol{\Gamma} = \mathsf{XOF}_1(h_1)$: This probability is bounded by the test's false positive probability.

Wrapping up the above argument, we obtain the following theorem:

Theorem 5.1 (Adapted version of [AHJ+23], Thm.6 and [DFMS22], Thm.5.2)**.** *Let* $\mathsf{ID3} = \mathsf{Elim}[\mathsf{ID5}_{\mathsf{Mirath}}, \mathsf{XOF}_1]$. *Let* $\mathsf{Mirath} = \mathsf{FS}'[\mathsf{ID3}_{\mathsf{Mirath}}, \mathsf{H}_2, \mathsf{XOF}_2]$. *Let* $\mathcal{A}_{\mathrm{nma}}$ *be an EUF-NMA adversary against* Mirath *in the QROM. Let* Q_X *be the number of queries* $\mathcal{A}_{\mathrm{nma}}$ *makes to oracle* X. *Then, there exist a QROM+ adversary* $\mathcal{A}_{\mathrm{snd}}$ *against* Mirath *and a QROM+ adversary* $\mathcal{A}'_{\mathrm{snd}}$ *against* $\mathsf{ID3}_{\mathsf{Mirath}}$ *such that*

$$
\begin{aligned}
&\mathsf{Adv}^{\text{euf-nma}}_{\mathsf{Mirath},\mathcal{A}_{\mathrm{nma}}}(\lambda)\\
&\le \Pr[(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \mathsf{sk}' \leftarrow \mathsf{Ext}\circ\mathcal{A}_{\mathrm{snd}}(\mathsf{vk}) : (\mathsf{vk},\mathsf{sk}') \notin R_{\boldsymbol{\Gamma}}]\\
&+ \Pr[(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \mathsf{sk}' \leftarrow \mathsf{Ext}'\circ\mathcal{A}'_{\mathrm{snd}}(\mathsf{vk}) : (\mathsf{vk},\mathsf{sk}') \in R_{\boldsymbol{\Gamma}} \setminus R_{\mathsf{Gen}}]\\
&+ \Pr[(\mathsf{vk},\mathsf{sk}) \leftarrow \mathsf{Gen}(1^\lambda), \mathsf{sk}' \leftarrow \mathsf{Ext}'\circ\mathcal{A}'_{\mathrm{snd}}(\mathsf{vk}) : (\mathsf{vk},\mathsf{sk}') \in R_{\mathsf{Gen}}].
\end{aligned}
$$

The running time of $\mathsf{Ext}\circ\mathcal{A}_{\mathrm{snd}}$ *is approximately that of* $\mathcal{A}_{\mathrm{nma}}$ *plus the verification time and the numbers of queries that* $\mathsf{Ext}\circ\mathcal{A}_{\mathrm{snd}}$ *made are* $Q_{\mathrm{snd},X} = Q_X + 1$ *for all but* H_3' *and* $Q_{\mathrm{snd},\mathsf{H}_3'} = Q_{\mathsf{H}_3'} + \tau N$. *The running time of* $\mathsf{Ext}'\circ\mathcal{A}'_{\mathrm{snd}}$ *and the number of queries that it made are the same as those of* $\mathsf{Ext}\circ\mathcal{A}_{\mathrm{snd}}$.

In the following subsections, we evaluate each term. We will obtain the following concrete bound as a corollary:

Corollary 5.1 (NMA Security of Mirath)**.** *Let* H_1, H_2, H_3, H_3', XOF_1, *and* XOF_2 *be random oracles. For any EUF-NMA adversary* $\mathcal{A}_{\mathrm{nma}}$ *that queries to the oracle* X *at most* Q_X *times, there exists a quantum adversary* $\mathcal{A}$ *against* Gen *such that*

$$
\begin{aligned}
&\mathsf{Adv}^{\text{euf-nma}}_{\mathsf{Mirath},\mathcal{A}_{\mathrm{nma}}}(\lambda) \le \mathsf{Adv}^{\text{solve}}_{\mathsf{Gen},\mathcal{A}}(\lambda) + 10(Q')^3 \cdot 2^{-2\lambda}\\
&+ 10(Q'')^2 \max\left\{q^{-\mu\rho}, Q''\tau N \cdot 2^{-2\lambda}\right\}\\
&+ \tilde{Q}^2 \left(\sqrt{10\max\left\{\tilde{Q}(\tau N+2)\cdot 2^{-2\lambda}, 2^{-w}N^{-\tau}\right\}} \atop +2e\sqrt{(\tilde{Q}+1)\cdot 2^{-w}N^{-\tau}}\right)^2\\
&+ 2((\tau-1)N+4)\cdot 2^{-2\lambda},
\end{aligned}
$$

where $Q = Q_{\mathsf{H}_1} + Q_{\mathsf{H}_3} + Q_{\mathsf{H}_3'} + Q_{\mathsf{XOF}_1} + Q_{\mathsf{H}_2} + Q_{\mathsf{XOF}_2} + 5 + N$, $Q' = Q + 2(\tau-1)N + 4$, $Q'' = Q + \tau N + 3$, *and* $\tilde{Q} = Q + (\tau-1)N + 4$. *The running time of* $\mathcal{A}$ *is approximately that of* $\mathcal{A}_{\mathrm{nma}}$ *plus the running times for the compressed random oracles, the verification algorithm, and the two extractors* Ext *and* Ext'.

We omit the proof.

6. EUF-CMA Security of Mirath

Theorem 6.1 (EUF-NMA to EUF-CMA of Mirath)**.** *Let* H_1, H_2, H_3, H_3', XOF_1, *and* XOF_2 *be random oracles. Let* $\kappa_{\max} := \lceil \log_2(\tau N)\rceil$. *For a QPT adversary* $\mathcal{A}$ *that queries to the oracles* X *at most* Q_X *times, there exist QPT adversaries* $\mathcal{A}_{\mathrm{nma}}$, $\mathcal{A}_{\mathrm{prf},\kappa}$ *for* $\kappa \in [\kappa_{\max}]$, *and* $\mathcal{A}_{\mathrm{joint}}$ *such that*

$$
\begin{aligned}
&\mathsf{Adv}^{\text{euf-cma}}_{\mathsf{Mirath},\mathcal{A}}(\lambda)\\
&\le \mathsf{Adv}^{\text{euf-nma}}_{\mathsf{Mirath},\mathcal{A}_{\mathrm{nma}}}(\lambda) + \sum_{\kappa\in[\kappa_{\max}]} \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{PRF}_{\mathsf{tree}},\mathcal{A}_{\mathrm{prf},\kappa}}(\lambda)\\
&+ \mathsf{Adv}^{\mathrm{prf}}_{\mathsf{H}_3'\times\mathsf{PRF}_{\mathsf{share}},\mathcal{A}_{\mathrm{joint}}}(\lambda) + \frac{632(Q_{\mathsf{H}_2}+Q_{\mathsf{Sign}}+2)^3}{2^{\ell_{\mathsf{H}_2}}}\\
&+ \frac{3Q_{\mathsf{Sign}}}{2}\sqrt{\frac{Q_{\mathsf{H}_2}}{2^{\ell_{\mathsf{salt}}}}} + \frac{3Q_{\mathsf{Sign}}}{2}\sqrt{\frac{Q_{\mathsf{XOF}_2}}{2^{\ell_{\mathsf{H}_2}}}}.
\end{aligned}
$$

The numbers of queries that $\mathcal{A}_{\mathrm{nma}}$ *makes are the same as those that* $\mathcal{A}_{\mathrm{cma}}$ *makes.* $\mathcal{A}_{\mathrm{prf},\kappa}$ *makes at most* $Q_{\mathsf{Sign}}\tau N/2$ *classical queries to its oracle and* $\mathcal{A}_{\mathrm{joint}}$ *makes* $Q_{\mathsf{Sign}}\tau$ *classical queries and* $Q_{\mathsf{H}_3'}$ *quantum queries to its oracle.*

Due to the space limitation, we only give proof sketch. We define six games and show they are statistically or computationally close.

- Game 0: This is the original game. The signing oracle uses the real prover algorithms $\tilde{\mathsf{P}}_1'$, which uses P_1, H_1, XOF_1, P_2 internally, to compute $a = (\mathtt{salt}, h_1, \mathtt{aux}, \boldsymbol{\alpha}_{\mathtt{mid}}, \boldsymbol{\alpha}_{\mathtt{base}})$ and $\tilde{\mathsf{P}}_2' = \mathsf{P}_3$ to compute $z = (\mathtt{seed}_c, \mathtt{com}_{-c})$ and π_{BAVC}. The signature is $\sigma = (h_2, \mathtt{ctr}, \mathtt{salt}, \mathtt{aux}, \boldsymbol{\alpha}_{\mathtt{mid}}, \pi_{\mathrm{BAVC}})$, where $h_2 = \mathsf{H}_2(\mathtt{vk}, \mathtt{msg}, a')$ with $a' = (\mathtt{salt}, h_1, \boldsymbol{\alpha}_{\mathtt{mid}}, \boldsymbol{\alpha}_{\mathtt{base}})$ and $(\mathbf{i}^*, v_{\mathrm{grinding}}) := \mathsf{XOF}_2(h_2, \mathtt{ctr})$.
- Game 1: We remove a bad event corresponding to a collision in the input to XOF_2. This is justified by the collision resistance of H_2, and the bound of the collision probability in the QROM is given by Theorem 3.1 of [Zha15].
- Game 2: The signing oracle randomly chooses a hash values h_2 and $(\mathbf{i}^*, v_{\mathrm{grinding}})$ of H_2 and XOF_2 and reprograms H_2 and XOF_2 later. Since the min-entropy of inputs for H_2 and XOF_2 is sufficiently high, we can use the adaptive (multi-point) reprogramming lemma, Lemma 3.1 and Corollary 3.1.
- Game 3: We split the computations of P_3 and incremenation of $\mathtt{ctr}$ into two simulating functions Sim_1 and Sim_2. Sim_1 selects $\mathbf{i}^*$, v_{grinding}, and $\mathtt{ctr}$, with the grinding procedure and Sim_2 computes the real computation of z and π_{BAVC}. Originally, $\mathbf{i}^*$ and v_{grinding} are computed from the outputs of P_1 and P_2. However, since they have already been modified to be chosen at random, this computation becomes independent and can

be performed at the beginning. Futhermore, the singing oracle first runs Sim_1, then runs P_1, P_2, and Sim_2.

- Game 4: We replace commitment part of P_1, that is, the commitment algorithm of the BAVC scheme, with a simulating function Sim_3. Sim_3 computes the tree nodes and commitments to be revealed correctly using $\mathsf{PRF}_{\mathsf{share}}$, while the other values are chosen at random. Since the signing oracle chooses $\mathbf{i}^*$ at the begininng, this modification becomes feasible due to the hiding property of commitments and the pseudorandomness of $\mathsf{PRF}_{\mathsf{tree}}$, $\mathsf{PRF}_{\mathsf{share}}$, and H'_3.
- Game 5: We replace the unsimulated part of P_1, that is, ComputeShares in [AAB+24], and the whole P_2 with simulating functions Sim_4 and Sim_5. This simulation leverages the zero-knowledge property of the protocol. By computing $\boldsymbol{\alpha}_{\mathtt{mid}}$ and $\boldsymbol{\alpha}_{\mathtt{base}}$ using the same procedure as in the signature verification, it becomes possible to derive these values solely from the seeds that do not correspond to $\mathbf{i}^*$. In this game, we can construct an NMA adversary.

6.1 Extension to RYDE

The structure of the RYDE is almost identical to that of Mirath, and for the following reasons, the same proof applies with only minor differences.

- XOF_2 is assumed to be collision-resistant. The usage of XOF_2 in RYDE is identical to that in Mirath, and thus inherits the same security guarantees.
- The inputs to both H_2 and XOF_2 retain sufficiently high min-entropy. As in Mirath, RYDE employs a fresh `salt` when computing H_2, and the resulting output h_2 serves as the input to XOF_2. This construction preserves min-entropy and satisfies the required assumption.
- The commitment scheme satisfies the hiding property under the assumption that the underlying $\mathsf{PRF}_{\mathsf{tree}}$, $\mathsf{PRF}_{\mathsf{share}}$, and H'_3 are secure. The commitment scheme used in RYDE is identical to BAVC from Mirath, and thus inherits its proven security properties.

The difference in RYDE lies in that a hash value is computed before the message is input to H_2. When we denote this hash function as H_0, we must assume that there are no collisions in the input to H_0. As in Theorem 6.1, the collison probability that we assume the QROM for H_0 is bounded by Theorem 3.1 of [Zha15].

References

[AAB+24] Gor Adj, Nicolas Aragon, Stefano Barbero, Magali Bardet, Emmanuele Bellini, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Andre Esser, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, Matthieu Rivain, Luis Rivera-Zamarripa, Carlo Sanna, Jean-Pierre Tillich, Javier Verbel, and Floyd Zweydinger, *Mirath (merger of MIRA/MiRitH)*, Tech. report, National Institute of Standards and Technology, 2024.

[ABB+24a] Najwa Aaraj, Slim Bettaieb, Loïc Bidoux, Alessandro Budroni, Victor Dyseryn, Andre Esser, Thibauld Feneuil, Philippe Gaborit, Mukul Kulkarni, Victor Mateu, Marco Palumbi, Lucas Perin, Matthieu Rivain, Jean-Pierre Tillich, and Keita Xagawa, *PERK*, Tech. report, National Institute of Standards and Technology, 2024.

[ABB+24b] Carlos Aguilar Melchor, Slim Bettaieb, Loïc Bidoux, Thibauld Feneuil, Philippe Gaborit, Nicolas Gama, Shay Gueron, James Howe, Andreas Hülsing, David Joseph, Antoine Joux, Mukul Kulkarni, Edoardo Persichetti, Tovohery H. Randrianarisoa, Matthieu Rivain, and Dongze Yue, *SDitH — Syndrome Decoding in the Head*, Tech. report, National Institute of Standards and Technology, 2024.

[ABB+24c] Nicolas Aragon, Magali Bardet, Loïc Bidoux, Jesús-Javier Chi-Domínguez, Victor Dyseryn, Thibauld Feneuil, Philippe Gaborit, Antoine Joux, Romaric Neveu, Matthieu Rivain, Jean-Pierre Tillich, and Adrien Vinçotte, *RYDE*, Tech. report, National Institute of Standards and Technology, 2024.

[AHJ+23] Carlos Aguilar Melchor, Andreas Hülsing, David Joseph, Christian Majenz, Eyal Ronen, and Dongze Yue, *SDitH in the QROM*, ASIACRYPT 2023, Part VII (Jian Guo and Ron Steinfeld, eds.), LNCS, vol. 14444, Springer, Singapore, December 2023, pp. 317–350.

[BBB+24] Carsten Baum, Lennart Braun, Ward Beullens, Cyprien Delpech de Saint Guilhem, Michael Klooß, Christian Majenz, Shibam Mukherjee, Emmanuela Orsini, Sebastian Ramacher, Christian Rechberger, Lawrence Roy, and Peter Scholl, *FAEST*, Tech. report, National Institute of Standards and Technology, 2024.

[BBFR24] Ryad Benadjila, Charles Bouillaguet, Thibauld Feneuil, and Matthieu Rivain, *MQOM — MQ on my Mind*, Tech. report, National Institute of Standards and Technology, 2024.

[BFG+24] Loïc Bidoux, Thibauld Feneuil, Philippe Gaborit, Romaric Neveu, and Matthieu Rivain, *Dual support decomposition in the head: Shorter signatures from rank SD and MinRank*, ASIACRYPT 2024, Part II (Kai-Min Chung and Yu Sasaki, eds.), LNCS, vol. 15485, Springer, Singapore, December 2024, pp. 38–69.

[DFMS22] Jelle Don, Serge Fehr, Christian Majenz, and Christian Schaffner, *Efficient NIZKs and signatures from commit-and-open protocols in the QROM*, CRYPTO 2022, Part II (Yevgeniy Dodis and Thomas Shrimpton, eds.), LNCS, vol. 13508, Springer, Cham, August 2022, pp. 729–757.

[Fen24] Thibauld Feneuil, *The polynomial-IOP vision of the latest MPCitH frameworks for signature schemes*, Audiovisual resource, August 2024, ACCESS Seminar, IHP.

[FR23] Thibauld Feneuil and Matthieu Rivain, *Threshold linear secret sharing to the rescue of MPC-in-the-head*, ASIACRYPT 2023, Part I (Jian Guo and Ron Steinfeld, eds.), LNCS, vol. 14438, Springer, Singapore, December 2023, pp. 441–473.

[FS87] Amos Fiat and Adi Shamir, *How to prove yourself: Practical solutions to identification and signature problems*, CRYPTO'86 (Andrew M. Odlyzko, ed.), LNCS, vol. 263, Springer, Berlin, Heidelberg, August 1987, pp. 186–194.

[GHHM21] Alex B. Grilo, Kathrin Hövelmanns, Andreas Hülsing, and Christian Majenz, *Tight adaptive reprogramming in the QROM*, ASIACRYPT 2021, Part I (Mehdi Tibouchi and Huaxiong Wang, eds.), LNCS, vol. 13090, Springer, Cham, December 2021, pp. 637–667.

[HJMN24] Andreas Hülsing, David Joseph, Christian Majenz, and Anand Kumar Narayanan, *On round elimination for special-sound multi-round identification and the generality of the hypercube for MPCitH*, CRYPTO 2024, Part I (Leonid Reyzin and Douglas Stebila, eds.), LNCS, vol. 14920, Springer, Cham, August 2024, pp. 373–408.

[NIS22] NIST, *Call for additional digital signature schemes for the post-quantum cryptography standardization process*, October 2022.

[Zha15] Zhandry, Mark, *A note on the quantum collision and set equality problems*, Quantum Information and Computation, vol.15, No. 7&8, Rinton Press, May 2015, pp. 557–567.