POLITECNICO DI TORINO
Repository ISTITUZIONALE

A Framework for Group Action-Based Multi-signatures and Applications to LESS, MEDS, and ALTEQ

Original
A Framework for Group Action-Based Multi-signatures and Applications to LESS, MEDS, and ALTEQ / D'Alconzo, G., Flamini, A., Meneghetti, A., Signorini, E.. - (2025), pp. 99-133. (28th IACR International Conference on Practice and Theory of Public Key Cryptography, PKC 2025 Røros (Norway) May 12–15, 2025) [10.1007/978-3-031-91823-0_4].

Availability:
This version is available at: 11583/3000008 since: 2025-05-09T11:11:47Z

Publisher:
Springer

Published
DOI:10.1007/978-3-031-91823-0_4

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright
Springer postprint/Author's Accepted Manuscript (book chapters)

This is a post-peer-review, pre-copyedit version of a book chapter published in Public-Key Cryptography – PKC 2025. The final authenticated version is available online at: http://dx.doi.org/10.1007/978-3-031-91823-0_4

(Article begins on next page)

28 November 2026

A Framework for Group Action-Based Multi-Signatures and Applications to LESS, MEDS, and ALTEQ

Giuseppe D'Alconzo[1], Andrea Flamini[2], Alessio Meneghetti[2], and Edoardo Signorini[3,1]

[1] Polytechnic University of Turin, Turin, Italy
giuseppe.dalconzo@polito.it
[2] University of Trento, Trento, Italy
andrea.flamini@unitn.it, alessio.meneghetti@unitn.it
[3] Telsy, Turin, Italy
edoardo.signorini@telsy.it

Abstract. A multi-signature scheme allows a list of signers to sign a common message. They are widely used in scenarios where the same message must be signed and transmitted by N users, and, instead of concatenating N individual signatures, employing a multi-signature can reduce the data to be sent. In recent years there have been numerous practical proposals in the discrete logarithm setting, such as MuSig2 (CRYPTO'21) for the Schnorr signature. Recently, these attempts have been extended to post-quantum assumptions, with lattice-based proposals such as MuSig-L (CRYPTO'22). Given the growth of group action-based signatures, a natural question is whether a multi-signature can be built on the same models. In this work, we present the first construction of such a primitive relying on group action assumptions. We obtain a 3-round scheme achieving concurrent security in the ROM. Moreover, we instantiate it using the three candidates to the additional post-quantum NIST's call, namely LESS, MEDS and ALTEQ, obtaining a good compression rate for different parameters sets.

Keywords: multi-signature, cryptographic group actions, code equivalence

1 Introduction

Aggregate and Multi-Signatures. An aggregate signature scheme allows n users to combine their individual signatures on separate messages to produce a single, directly verifiable aggregate signature. This approach aims to achieve shorter signature lengths compared to trivial concatenation of individual signatures. Hence, aggregate signatures are particularly useful in scenarios where a large number of signatures need to be transmitted and the communication costs within the network are not negligible. The notion of aggregate signatures was initially introduced in a seminal paper by Boneh et al. [20]. The authors proposed

a method that allows a third party to aggregate signatures from distinct users using a public aggregation algorithm. Although this general aggregation approach is efficient and valuable in many applications, it is notoriously difficult to achieve in practice without the use of bilinear pairing [20,12], indistinguishability obfuscation [47] or non-interactive arguments of knowledge [3,37,71]. Furthermore, there are important scenarios where interactive protocols can be employed and it is enough to aggregate signatures on a common message. Such constructions are better known as multi-signatures and typically target additional properties, including key aggregation and signature compatibility with the underlying scheme. Although multi-signatures were introduced separately from aggregate signatures [48,63,62] and the usage scenarios are typically distinct, it is well known that a multi-signature can be easily transformed into an interactive aggregate signature by requiring participants to agree on a concatenation of messages to be signed [13]. Numerous multi-signatures have been proposed for Schnorr's signature [61,13,6,53,69,55,39,60], with recent near-optimal schemes MuSig2 [59] and DWMS [4] requiring only one round of interaction and allowing key aggregation.

Increasing activity in the development of post-quantum signatures has led the community to explore signature aggregation in this field. Lattice-based proposals were initially introduced in a restricted model where signatures are aggregated sequentially by each subsequent user [40]. This approach generalizes the original line of works on sequential aggregation of trapdoor permutations [52,58,23,44], extending it to hash-and-sign schemes based on the GPV construction [45], allowing aggregation of Falcon signatures. Within the Fiat-Shamir paradigm, which includes Dilithium, [22] have recently proposed a sequential aggregation scheme, achieving however only limited compression. Finally, the idea of aggregating signatures using lattice-based SNARK was recently investigated in [3] and formalized in [1]. In [72], the authors present a SNARK-based constant-size aggregate signature, leveraging the Legendre PRF. Also, in the interactive model, there is a long line of work proposing lattice-based multi-signatures [41,43,34,27] culminating with MuSig-L [21], which achieves properties similar to those of MuSig2 for lattices.

Besides lattice-based solutions and generic approaches based on SNARKs, the landscape of proposals tailored for other post-quantum assumptions is very limited. Recently, in [56], the authors generalized the sequential aggregation framework of [52,40] to generic trapdoor functions, making it compatible with hash-and-sign schemes from multivariate and code-based assumptions. For Fiat-Shamir signatures, a tailored non-interactive or sequential solution appears difficult without achieving limited compression, and no scheme has been proposed.

Group Actions. Cryptographic group actions have received a lot of attention in the last years, mainly due to the isogeny-based cryptography with the CSIDH action [24]. However, other actions raised interest in the post-quantum panorama for the competitive digital signatures based on them. Some examples are NIST's additional call proposal, LESS [7], MEDS [29] and ALTEQ [19]. Using this algebraic framework, other primitives can be built, ranging from Pseudo Random

Functions [2] and Updatable Encryption [51] to digital signature schemes with advanced functionalities. Some examples of the latter are the threshold signature given in [9], the (linkable) ring one from [15] and the threshold ring signature shown in [65]. If we additionally assume the commutativity of the action, the design space enlarges to Oblivious Transfers [2], Diffie-Hellman key exchange [32] and group signatures [14].

Following the construction given in [46] for Graph Isomorphism, a sigma protocol for group actions can be constructed, allowing a prover to convince a verifier that she knows the group element mapping a set element x_0 to another set element x_1. The first message of the protocol is given by $\tilde{x} = \tilde{g} \star x_0$ for a random group element $\tilde{g}$. The challenge is a random bit b and instructs the prover to reveal the group element mapping the set element x_b to $\tilde{x}$. This sigma protocol has knowledge error $\frac{1}{2}$, and this quantity can be reduced using parallel repetitions. Digital signatures based on cryptographic group actions (e.g. [8,30,70]) are obtained by turning parallel instances of the sigma protocol for group actions into a non-interactive protocol by applying the Fiat-Shamir transform.

1.1 Our techniques

Multi-Signature Overview. In this paper, we present a novel multi-signature scheme based on cryptographic group actions. In our scheme, the key held by an individual party P_i coincides with that of the underlying signature based on group action: the private key is a group element $\mathsf{sk}_i = g_i \in G$, and the public key is a set element $\mathsf{pk}_i = g_i \star x_0 \in X$, where x_0 is the common base point. In the signing phase, the n parties will participate in an interactive round-robin protocol in which they will compute a common commitment $\tilde{x}$, combining the actions of different group elements so that no party knows entirely the element that maps x_0 to $\tilde{x}$. To achieve this result, party P_i samples a random group element $\tilde{g}_i$ and, at the end of the round-robin, the challenge $\tilde{x}$ is equal to $(\prod_{j=1}^{n} \tilde{g}_j) \star x_0$. In this phase, the signers also generate a random salt r, by firstly committing and then simultaneously releasing random salts $r_1, \ldots, r_n$, one for each signer. Once the commitment and the salt r have been generated, the signing parties compute a random challenge $c \in \{0, \ldots, n\}$, which specifies the public key $\mathsf{pk}_c = x_c$ of the c-th signer, and instructs the signers to compute a group element which maps x_c to the commitment $\tilde{x}$. Again, this will require the signers to cooperate to compute the responses since the knowledge of the group element mapping the base point x_0 to the commitment $\tilde{x}_i$ is distributed among the signers.

This approach mimics a standard optimization used in the context of group action-based signatures, whereby multiple public keys are used to increase the challenge space and decrease the signature size [36]. On the other hand, the security model where an adversary has to forge a multi-signature involving a target user with the possibility of corrupting other signatories can be traced back to the security of a peculiar variant of the centralized scheme, where the user can generate ephemeral keys during the signing process.

Sigma Protocol Variant. As an intermediate step in our construction, we introduce a variant of the digital signature from cryptographic group actions. This variant, while less efficient than the standard signature scheme, serves as a crucial proof artifact and aids in the security reduction of our multi-signature scheme.

More specifically, we define a variant of the sigma protocol from cryptographic group actions which instructs the signer to generate a number $n-1$ of ephemeral keys $\{\hat{x}_2, \ldots, \hat{x}_n\}$ that are set elements generated by applying a random group element $\hat{g}_i$ to the base point x_0. The challenge $c \in \{1, \ldots, n\}$ specifies which set element (either the signer's public key or one of the ephemeral keys) should be used in the response calculation. This modified protocol is then transformed into a digital signature scheme using the Fiat-Shamir transform.

Although this centralized variant is less efficient due to its higher soundness error of $n/(n+1)$, it plays a crucial role in proving the security of our multi-signature scheme. In fact, the ephemeral public keys in this variant correspond to the public keys of the parties in our multi-signature scheme.

Technical Lemmas. The number of iterations required before applying the Fiat-Shamir transform to the above Sigma protocol is variable and depends on the number of ephemeral keys generated. This is a novel and non-standard approach in the construction of Fiat-Shamir signatures, but one that more closely represents the adversary's capabilities in our multi-signature. To prove the security of the centralized variant scheme, we have shown that the underlying Sigma protocol is a proof of knowledge by providing an explicit description of the knowledge extractor. Our proof does not require any additional assumptions about group action beyond those necessary for the construction of a digital signature.

Concurrent Security. In the security proof of our multi-signature, the rewinding of the adversary is not required to answer the signing queries, so that an adversary can open multiple concurrent signing sessions. Therefore, the centralized scheme adversary is able to correctly simulate the unforgeability game of the multi-signature in polynomial time. This guarantees the concurrent security of our scheme.

Note that, although random salt r generation requires an additional round of interaction, its use is crucial in the security reduction from the variant of the centralized signature. In fact, the reduction can correctly answer the signing queries without knowing the secret key of the party under its control thanks to the ability to program the random oracle, keeping negligible the probability to overwrite the hash table since the values r will be different in every signing protocol execution.

Current Limitations. In this work, our primary focus is on achieving efficient signature aggregation with provable security in the ROM. The emphasis on signature compression is particularly important because one of the main drawbacks

of digital signatures based on cryptographic group actions is their typically larger signature size compared to other post-quantum signature schemes. Our proposed scheme aims to mitigate this disadvantage, making signatures from cryptographic group actions a viable option in practical scenarios requiring multiple users to sign the same message. In addition, our scheme enjoys a tight security reduction to a centralized signature scheme, which in turn has the same security features as the underlying group action-based signature scheme, which reduces to the one-wayness of the group action. On the other hand, our construction lacks advanced features such as key aggregation, where signature verification is shared with the centralized scheme and requires the use of a single key obtained by combining that of the participants.

Outline. The paper is organised as follows. In Section 2 we recall the preliminaries on group actions-based digital signature schemes, multi-signatures and the associated security notions. In Section 3 we describe the variant of the centralized digital signature scheme based on cryptographic group actions, then we define our main contribution which is our multi-signature scheme. In Section 4 we prove our multi-signature scheme secure, building a reduction from the unforgeability of our multi-signature to the unforgeability of the variant of the centralized digital signature from group actions. Section 5 describes how to reduce the size of our multi-signature scheme by adapting some well-known techniques used to optimize group action-based digital signature schemes. Finally, in Section 6 we instantiate our multi-signature scheme using LESS, MEDS and ALTEQ as underlying centralized digital signature schemes.

2 Notation and Preliminaries

For $n, a, b \in \mathbb{N}$, we denote by $[n]$ the set $\{ 1, \ldots, n \}$ and by $[a, b]$ the set $\{ a, \ldots, b \}$. For a finite set X, we write $|X|$ for the cardinality of X and by $x \leftarrow\!\!\$\ X$, we denote the sample of the element x from $U(X)$, the uniform distribution over X. Moreover, for an algorithm A, we write $x \leftarrow A(y)$ to denote the assignment of x to the output of A on input y and for an adversary $\mathcal{A}$ and a function F, we write $x \leftarrow \mathcal{A}^{\mathsf{OF}}$ the assignment of x of the output of $\mathcal{A}$ with oracle access to F. In an interactive protocol between n parties $P_1, \ldots, P_n$, we assume that each party has access to point-to-point communication channels. When the interactive protocol is run by a party P_i, we write $x \twoheadrightarrow P_j$ to denote the transmission of x from P_i to P_j. Similarly, we write $x \twoheadleftarrow P_j$ to denote a transmission from P_j to P_i and the subsequent assignment to x.

2.1 Cryptographic Group Actions

We introduce the algebraic framework of group actions, in which many cryptographic assumptions from the literature can be modelled.

Definition 1. *A group G with identity e is said to act on a set X if there is a map $\star : G \times X \to X$ such that $e \star x = x$ and $(gh) \star x = g \star (h \star x)$ for every g, h in G and x in X. In this case, we say that the triple $(G, X, \star)$ is a* group action.

We need some additional requirements on the group actions we use, in particular, we want the action to be *effective* [2], i.e. there exist efficient algorithms to sample and represent elements in X and G, to compute products and inverses in G and to compute the action $\star$. On the other hand, the following problem must be intractable.

Definition 2. *Given the action $(G, X, \star)$, the* Group Action Inversion Problem *(GAIP) asks, on input (x, y) in X, to find, if any, an element g in G such that $g \star y = x$. Given x_0 in X, the* GAIP for x_0 *assumes that the input is of the form $(x_0, g \star x_0)$.*

To prove the security of digital signatures from group actions in the quantum random oracle model, we need to assume the hardness of the following problem.

Definition 3. *Given the action $(G, X, \star)$, the* Stabilizer Computation Problem *asks, on input x_0 in X, to find, if any, an element g in G such that g is not the identity and $g \star x_0 = x_0$. In other words, the problem asks to find a nontrivial element of $\mathrm{Stab}(x_0) = \{g \in G \mid g \star x_0 = x_0\}$, the stabilizer group of x_0.*

There are two kinds of noteworthy post-quantum actions from the cryptographic literature: the ones from isegenies, used in CSI-FiSh [16], SeaSign [36] and SCALLOP [35,26], which are regular and their stabilizers are trivial, and the ones concerning tensors [49,70] and linear codes [8,30]. Even if the latter are not regular, for cryptographic relevant parameters, the stabilizers of randomly chosen elements are heuristically trivial [18,67].

2.2 Digital Signatures

A digital signature scheme Sig is a tuple of three algorithms $(\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$:

- $\mathsf{KGen}(1^\lambda)$: takes as input a security parameter 1^λ in unary and generates a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{Sign}(\mathsf{sk}, m)$: takes as input a signing key sk and a message m and returns a signature σ.
- $\mathsf{Vrfy}(\mathsf{pk}, m, \sigma)$: takes as input a verification key pk, a message m and a signature σ and returns 1 for acceptance or 0 for rejection.

We define the standard notion of existential unforgeability against chosen-message attack (EUF-CMA) [50, Def. 13.2].

Definition 4 (EUF-CMA security). *Let O be a random oracle, let $\mathsf{Sig} = (\mathsf{KGen}, \mathsf{Sign}, \mathsf{Vrfy})$ be a signature scheme, let $\mathcal{A}$ be an adversary. We define the advantage of $\mathcal{A}$ playing the* EUF-CMA *game against Sig in the random oracle model as:*

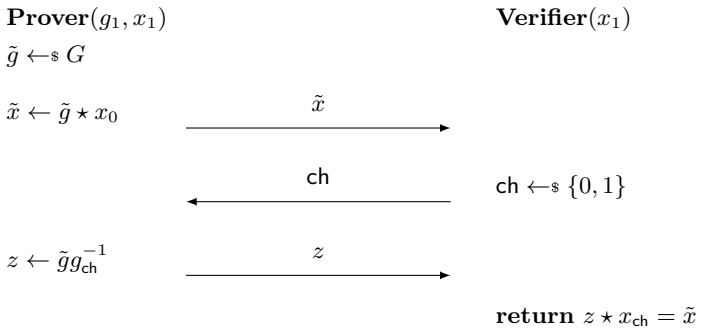
$$\mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{Sig}}(\mathcal{A}) = \Pr\left[\begin{array}{c}\mathsf{Vrfy}(\mathsf{pk}, m, \sigma) = 1 \\ \mathsf{OSign}(\mathsf{sk}, \cdot) \text{ not queried on } m\end{array} \;\middle|\; \begin{array}{c}(\mathsf{pk}, \mathsf{sk}) \leftarrow \mathsf{KGen}(1^\lambda) \\ (m, \sigma) \leftarrow \mathcal{A}^{\mathsf{O}, \mathsf{OSign}(\mathsf{sk}, \cdot)}(\mathsf{pk})\end{array}\right].$$

Algorithm 1: Identification Protocol Based on Group Action

`Setup`: Choose $x_0 \in X$ and let $g_0 = e \in G$

`Private key`: $g_1 \in G$.

`Public key`: $x_1 \leftarrow g_1 \star x_0$. Observe that $x_i = g_i \star x_0$ for $i = 0, 1$.



We say that Sig *is* existential unforgeable against chosen-message *attacks if the advantage* $\mathsf{Adv}_{\mathsf{Sig}}^{\text{EUF-CMA}}(\mathcal{A})$ *is negligible for any adversary* $\mathcal{A}$.

A slightly stronger notion can also be considered, where the adversary may produce forgery on messages already queried to the signing oracle, provided that the forged signature is distinct from the oracle's response. We denote this variant as *strong* EUF-CMA (SUF-CMA).

2.3 Signature from Cryptographic Group Action

It is possible to obtain an identification protocol from a cryptographic group action, as described in Algorithm 1.

It is known that the protocol in Algorithm 1 is complete, 2-special sound and HVZK sigma protocol. Through parallel repetitions, it is possible to amplify the knowledge soundness of the protocol and obtain a digital signature by applying the Fiat-Shamir transform [42]. Assuming some standard security notion on the group action, the EUF-CMA security of the signature is proved in the (Quantum) Random Oracle Model [38,18].

For the multi-signature of Section 3, we will consider a variant Π of the previous Sigma protocol. Intuitively, this variant allows the Prover to artificially enlarge the challenge space using ephemeral keys in the commitment phase. The resulting signature is inefficient for a direct application, but it more accurately captures the perspective of the individual signer in the multi-signature of Section 3. The modified scheme is described in Section 3.1.

Game 1: MS-UF-CMA$_{\mathsf{MS}}$

$\mathcal{M}_{\mathsf{sid}}$ is a machine running the instruction of the party P_i in the multi-signature protocol $\mathsf{MuSign}(\mathsf{sid}, \mathsf{sk}^\star, \mathsf{pk}^\star, m, L)$, where $L = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ such that $\mathsf{pk}_i = \mathsf{pk}^\star$.

$\mathcal{Q} \leftarrow \emptyset$; $\mathcal{S} \leftarrow \emptyset$
$\mathsf{pp} \leftarrow\!\!\$\ \mathsf{Setup}(\lambda)$
$(\mathsf{pk}^\star, \mathsf{sk}^\star) \leftarrow\!\!\$\ \mathsf{KGen}(\mathsf{pp})$
$(L, m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{O},\mathsf{OMuSign}}(\mathsf{pk}^\star)$
if $\mathsf{pk}^\star \notin L \vee (m, L) \in \mathcal{Q}$ **then**
  return $\bot$
return $\mathsf{MuVrfy}(L, m, \sigma)$

$\mathsf{OMuSign}(\mathsf{sid}, msg)$:
if $\mathsf{sid} \notin \mathcal{S}$ **then**
  $(m, L) \leftarrow msg$
  if $\mathsf{pk}^\star \notin L$ **then**
    return $\bot$
  $\mathcal{M}_{\mathsf{sid}} \leftarrow\!\!\$\ \mathsf{MuSign}(\mathsf{sid}, \mathsf{sk}^\star, \mathsf{pk}^\star, m, L)$
  $\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, L)\}$
  $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathsf{sid}\}$
  return $\mathcal{M}_{\mathsf{sid}}()$
return $\mathcal{M}_{\mathsf{sid}}(msg)$

2.4 Multi-Signatures

A multi-signature scheme MS is a tuple of four algorithms $(\mathsf{Setup}, \mathsf{KGen}, \mathsf{MuSign}, \mathsf{MuVrfy})$.

- $\mathsf{Setup}(1^\lambda)$: takes as input a security parameter 1^λ in unary and outputs a public parameter pp.
- $\mathsf{KGen}(\mathsf{pp})$: takes as input a public parameter pp and generates a key pair $(\mathsf{pk}, \mathsf{sk})$.
- $\mathsf{MuSign}(\mathsf{sid}, \mathsf{sk}, \mathsf{pk}, m, L)$: is an interactive protocol that is run by a party P_i taking as input a session ID sid, a key pair $(\mathsf{pk}, \mathsf{sk})$, a message to be signed m and an ordered set of co-signers' public keys $L = (\mathsf{pk}_1, \ldots, \mathsf{pk}_n)$ such that $\mathsf{pk}_i = \mathsf{pk}$. The protocol terminates with each party obtaining a signature σ as output.
- $\mathsf{MuVrfy}(L, m, \sigma)$: takes as input an ordered set of public keys L, a message m and a signature σ and returns 1 for acceptance or 0 for rejection.

Below, we show the definition of multi-signature unforgeability under adaptive chosen message (MS-UF-CMA). In this model, the forger controls all signers' private keys except for at least one honest signer. The forger can choose the keys of the rogue signers and adaptively query an aggregate signature oracle. Finally, to win the experiment, the forger must produce a valid, non-trivial multi-signature involving the public key of the honest signer. The security notion is adapted from [34] and allows the adversary to open concurrent signing sessions.

Definition 5 (MS-UF-CMA **Security).** *Let* O *be a random oracle, let* $\mathsf{MS} = (\mathsf{Setup}, \mathsf{KGen}, \mathsf{MuSign}, \mathsf{MuVrfy})$ *be a multi-signature scheme, and let* $\mathcal{A}$ *be an adversary. We define the advantage of* $\mathcal{A}$ *playing the* MS-UF-CMA *game (Game 1) against* MS *in the random oracle model as:*

$$\mathsf{Adv}^{\text{MS-UF-CMA}}_{\mathsf{MS}}(\mathcal{A}) = \Pr[\text{MS-UF-CMA}_{\mathsf{MS}}(\mathcal{A}) = 1].$$

We say that MS *is* existential unforgeable against chosen-message *attacks if the advantage* $\mathsf{Adv}_{\mathsf{MS}}^{\text{MS-UF-CMA}}(\mathcal{A})$ *is negligible for any adversary* $\mathcal{A}$.

3 The Multi-Signature Scheme

In this section, we present a multi-signature scheme based on cryptographic group actions, for which the key pairs used by the signing parties are compatible with the key pairs of standard digital signatures based on group actions such as LESS [8], MEDS [30], or ALTEQ [19].

3.1 Modified Centralized Signature

In the following, we present a variant of the base Σ-protocol from cryptographic group action (Algorithm 1), and we prove the EUF-CMA security of the associated signature. The variant allows the signer to use ephemeral keys during signature creation. Although this has no impact on security, the modified signature allows the behaviour of an adversary to be abstracted more accurately in the multi-signature protocol, and is, therefore, a useful tool in the security proof of the upcoming scheme.

Given a security parameter λ, $(G, X, \star)$ will denote a cryptographic group action, x_0 is a fixed element in X, $\hat{g}_0 = e_G$ is the identity of G, and N is a fixed positive integer. Given $n \in \mathbb{N}$, let $t(n)$ be the minimum positive integer such that $(n/(n+1))^{t(n)} \leq 2^{-\lambda}$. Let $\mathsf{Ch} \subseteq [0, N]^{t(N)}$ and $f_n \colon \mathsf{Ch} \to [0, n]^{t(n)}$ be a family of maps such that $f_n^{-1}\left(U([0, n]^{t(n)})\right) \sim U(\mathsf{Ch})$ for any $n \in [1, N]$. The introduction of such a family with these properties is needed when proving that this protocol is a proof of knowledge (Appendix A.2). A practical instantiation of such a family is given in Appendix B.1.

Protocol 1 (Group Action Σ-protocol with Ephemeral Keys)**.** *Given the public parameters* $\mathsf{pp} = (G, X, \star, x_0, \hat{g}_0, N, \mathsf{Ch}, \{f_n\})$*, the protocol between a Prover* $(\mathsf{P}_1, \mathsf{P}_2)$ *and a Verifier* $(\mathsf{V}_1, \mathsf{V}_2)$ *proceeds as follows:*

- $(g_1, x_1) \leftarrow\!\!\$\ \mathsf{Gen}(\mathsf{pp})$*: the key-generation algorithm takes as input the public parameters* pp*. It uniformly samples* $g_1 \in G$ *and computes* $x_1 \leftarrow g_1 \star x_0$*. It returns the witness-statement pair* (x_1, g_1)*.*
- $\mathsf{com} \leftarrow\!\!\$\ \mathsf{P}_1(g_1, x_1)$*: given a statement* $x_1 \in X$ *and the corresponding witness* $g_1 \in G$*, the prover chooses* $n \in [1, N]$*. Then, it uniformly samples* $\hat{g}_k$ *and computes* $\hat{x}_k \leftarrow \hat{g}_k \star x_0$ *for* $k \in [2, n]$*. Then, it uniformly samples* $\tilde{g}^{(j)}$ *and computes* $\tilde{x}^{(j)} \leftarrow \tilde{g}^{(j)} \star x_0$ *for* $j \in [1, t(n)]$*. Finally, it returns* $\mathsf{com} \leftarrow (\hat{x}_2, \ldots, \hat{x}_n, \tilde{x}^{(1)}, \ldots, \tilde{x}^{(t(n))})$*.*
- $\mathsf{ch} \leftarrow\!\!\$\ \mathsf{V}_1(\mathsf{com})$*: given a commitment* com*, the verifier returns a uniformly random challenge* $\mathsf{ch} \in \mathsf{Ch}$*.*
- $\mathsf{rsp} \leftarrow \mathsf{P}_2(g_1, x_1, \mathsf{com}, \mathsf{ch})$*: given a statement* x_1*, the corresponding witness* g_1*, a commitment* $\mathsf{com} = (\hat{x}, \tilde{x})$ *and a challenge* $\mathsf{ch} \in \mathsf{Ch}$*, the prover sets* $\hat{g}_1 = g_1, \hat{x}_1 = x_1$ *and computes* $\mathsf{ch}' \leftarrow f_n(\mathsf{ch})$*. Then, for each component* $\mathsf{ch}'_j \in [0, n]$ *of* ch'*, they compute a response* $z_j \leftarrow \tilde{g}^{(j)} \hat{g}_{\mathsf{ch}'_j}^{-1}$ *for* $j \in [1, t(n)]$*. Finally, they output* $\mathsf{rsp} \leftarrow (z_1, \ldots, z_{t(n)})$*.*

– $\{0,1\} \leftarrow \mathsf{V}_2(x_1, \mathsf{com}, \mathsf{ch}, \mathsf{rsp})$*: given a statement* $x_1 \in X$*, a commitment* $\mathsf{com} = (\hat{x}, \tilde{x})$*, a challenge* ch *and a response* $\mathsf{rsp} = (z_1, \ldots, z_{t(n)})$*, the verifier proceeds as follows. They compute* $\mathsf{ch}' \leftarrow f_n(\mathsf{ch})$ *and set* $\hat{x}_1 = x_1$*. Then, for each* $j \in [1, t(n)]$*, compute* $\tilde{y}^{(j)} \leftarrow z_j \star \hat{x}_{\mathsf{ch}'_j}$*. The verifier accepts (returns 1) if* $\tilde{y} = \tilde{x}$*, otherwise rejects (returns 0).*

We denote Protocol 1 with Π. In Appendix A, we show that Π is correct, HVZK, and knowledge sound. Once the Prover choose $n \in [1, N]$, the soundness of the protocol is $\kappa_n^{t(n)}$, with $\kappa_n = n/(n+1)$. Therefore, due to the choice of $t(n)$, the protocol has negligible soundness error.

By applying the Fiat-Shamir transform to the protocol Π, we obtain a digital signature scheme $\mathsf{FS}[\Pi]$. The signature is obtained by taking the transcript of Π without the challenge. The challenge can be recovered as the output of a random oracle H on the commitment com and the message m. For ease of notation, we modify the syntax of the signing process by admitting an additional argument, allowing the signer to choose the number of ephemeral keys to be used. The full description of the signature scheme can be found in Algorithm 2.

In the signature scheme, instead of computing f_n on the output of single random oracle H, we consider multiple, independent random oracles $\mathsf{H}^{(1)}, \ldots, \mathsf{H}^{(N)}$ with $\mathsf{H}^{(n)} \colon \{0,1\}^* \to [0,n]^{t(n)}$. In Appendix B, we show how to construct a family of functions $\{f_n\}$ that satisfies the above properties. We also prove that, given a random oracle H on $[0,N]^{t(N)}$, each f_n defines a function $\mathsf{H}^{(n)}$ which is indifferentiable [54] from a random oracle on $[0,n]^{t(n)}$. This can be achieved whithin the framework of oracle cloning and read-only indifferentiability introduced in [11]. Notice that, in practice, the functions $\mathsf{H}^{(n)}$ can be instantiated with an extendable-output function (e.g. SHAKE [57]) via domain separation.

Theorem 1. *Let* Π *be as in Protocol 1 for a cryptographic group action* $(G, X, \star)$ *with base element* x_0*. If no polynomial-time (quantum) adversary can solve the GAIP (Definition 2) and the Stabilizer Computation Problem (Definition 3) for* x_0 *except with a negligible probability, then* $\mathsf{FS}[\Pi]$ *(Algorithm 2) is* SUF-CMA *in the (quantum) random oracle model.*

Sketch of proof. The complete proof can be found in Appendix A. We first prove that the Sigma protocol underlying $\mathsf{FS}[\Pi]$ is a (quantum) proof of knowledge. We start by taking a simplified Sigma protocol $\Pi[n]$, with a fixed number of $n-1$ ephemeral keys, and showing that it is a proof of knowledge roughly equivalent to the basic protocol of Algorithm 1, with a higher knowledge error. Next, we show that we can use the knowledge extractor for $\Pi[n]$ to extract a witness from a dishonest prover against Π, moreover, we show that this does not change the knowledge error of the protocol. Finally, we show that, if the base protocol in Algorithm 1 is correct, has high min-entropy, and is HVZK, then Π also has the same properties. We observe that these properties are required in concrete applications to construct digital signatures from group actions, so we make the same assumptions in the construction of the variant. Therefore, [38, Theorem 22] can be applied to show that the signature is EUF-CMA in the QROM. □

Algorithm 2: Variant Signature Scheme based on Group Actions $\mathsf{FS}[\Pi]$

$\star\colon G \times X \to X$ is a cryptographic group action. $\mathsf{H}^{(n)}\colon \{0,1\}^* \to [0,n]^{t(n)}$ is a random oracle.

$\mathsf{Setup}(1^\lambda)$:
1: $x_0 \leftarrow\!\!\$\ X$
2: $\mathsf{pp} \leftarrow x_0$
3: **return** pp

$\mathsf{KGen}(\mathsf{pp} = x_0)$:
1: $g_1 \leftarrow\!\!\$\ G$
2: $x_1 \leftarrow g_1 \star x_0$
3: **return** $(\mathsf{pk} = x_1, \mathsf{sk} = g_1)$

$\mathsf{Vrfy}(\mathsf{pk} = \hat{x}_1,\ m,\ \sigma)$:
1: $(L, \tilde{x}, z_1, \ldots, z_{t(n)}) \leftarrow \sigma$
2: $(\hat{x}_2, \ldots, \hat{x}_n) \leftarrow L$
3: $\mathsf{ch} \leftarrow \mathsf{H}^{(n)}(L, \tilde{x}, m)$
4: **for** $j \leftarrow 1, \ldots, t(n)$ **do**
5: $\quad \tilde{x}'_j \leftarrow z_j \star \hat{x}_{\mathsf{ch}_j}$
6: $\tilde{x}' \leftarrow (\tilde{x}'_1, \ldots, \tilde{x}'_{t(n)})$
7: **return** $\tilde{x}' = \tilde{x}$

$\mathsf{Sign}(\mathsf{sk},\ \mathsf{pk},\ m,\ n)$:
1: $\hat{x}_0 \leftarrow x_0$; $\hat{g}_0 \leftarrow e$
2: $\hat{x}_1 \leftarrow \mathsf{pk}$; $\hat{g}_1 \leftarrow \mathsf{sk}$
3: **for** $k \leftarrow 2, \ldots n$ **do**
4: $\quad \hat{g}_k \leftarrow\!\!\$\ G$
5: $\quad \hat{x}_k \leftarrow \hat{g}_k \star x_0$
6: $L \leftarrow (\hat{x}_2, \ldots, \hat{x}_n)$
7: **for** $j \leftarrow 1, \ldots, t(n)$ **do**
8: $\quad \tilde{g}^{(j)} \leftarrow\!\!\$\ G$
9: $\quad \tilde{x}^{(j)} \leftarrow \tilde{g}^{(j)} \star x_0$
10: $\tilde{x} \leftarrow (\tilde{x}^{(1)}, \ldots, \tilde{x}^{t(n)})$
11: $\mathsf{ch} \leftarrow \mathsf{H}^{(n)}(L, \tilde{x}, m)$
12: **for** $j \leftarrow 1, \ldots, t(n)$ **do**
13: $\quad z_j \leftarrow \tilde{g}^{(j)} \hat{g}^{-1}_{\mathsf{ch}_j}$
14: **return** $\sigma \leftarrow (L, \tilde{x}, z_1, \ldots, z_{t(n)})$

3.2 Our Multi-Signature Scheme

The multi-signature scheme is designed in a way that closely resembles the centralized digital signature scheme described in the previous section. In fact, in the multi-signature that we present in this section, the ephemeral signing keys used in the centralized signature in Algorithm 2 are replaced by the signing keys of the other parties taking part in the signing process. We recall that the centralized signature in Algorithm 2 is not efficient in any way, but it is unforgeable under chosen message attacks. In the security analysis, we will reduce the security of the multi-signature scheme, according to Definition 5, to the unforgeability of the centralized digital signature $\mathsf{FS}[\Pi]$.

At a high level, the multi-signature signing algorithm that we present instructs each party in the signing set to perform the following operations. Suppose L is an ordered signers set of n users $P_1, \ldots, P_n$. Each party P_i in L randomly generates a salt r_i, which will be used to generate a shared randomness associated to the signing session, and contributes to the creation of the sigma protocol commitment $\tilde{x}$. We obtain a three-round protocol consisting in two robin rounds and one broadcast. In particular, the parties in L perform the following operations in a round-robin fashion:

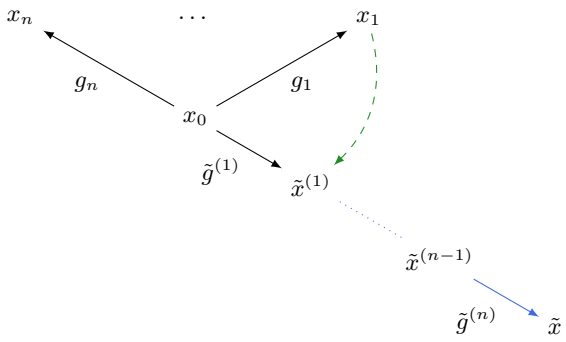


Fig. 1: High level description of MS-GA scheme of Algorithm 3, answering on $\mathsf{ch} = 1$. P_1 reveals the map from x_1 to $\tilde{x}^{(1)}$, while all other parties reveal the ephemeral group element $\tilde{g}^{(j)}$.

1. P_i commits to a random salt r_i, by computing c_i, a commitment which binds r_i also to the commitments of the parties acting before P_i in the ordered set L[4];
2. P_i contributes to the generation of the sigma protocol commitment $\tilde{x}$ by generating $t(n)$ group elements $\tilde{g}^{(i)} = (\tilde{g}_1^{(i)}, \ldots, \tilde{g}_{t(n)}^{(i)})$ and using them to compute the partial commitment in $\tilde{x}^{(i)}$ starting from the partial commitment it received from the party acting before it, namely $\tilde{x}^{(i-1)}$.

Then, P_i sends to P_{i+1} the cryptographic commitment c_i and the sigma protocol partial commitments $\tilde{x}^{(i)}$. The same operations are repeated by each party, until the last signing party P_n, which broadcasts its commitment c_n and $\tilde{x} = \tilde{x}^{(n)}$. Then, all the parties reveal their randomness r_i and check that the cryptographic commitments have been honestly computed. If this is the case, each party computes the shared randomness $r \leftarrow \mathsf{H}_1(r_1, \ldots, r_n, \mathsf{sid})$ which allows sessions to be separated. Using the shared randomness r and the commitment $\tilde{x}$, the parties generate the challenge $\mathsf{ch} \leftarrow \mathsf{H}_2^{(n)}(\tilde{x}, r, L, m)$, a string of $t(n)$ elements in $\{0, 1, \ldots, n\}$. In the response phase, a challenge $\mathsf{ch}_j = i$ requires revealing a map from x_i, the public key of P_i when $i \neq 0$, to $\tilde{x}_j = \tilde{x}_j^{(n)}$. Recall that $g_0 = e$ and $x_0 = g_0 \star x_0$. Each party P_k, for $k \neq i$, reveals the ephemeral group element $\tilde{g}_j^{(k)}$. P_i then computes the response as

$$z_j = \left(\prod_{k=0}^{n-1} \tilde{g}_j^{(n-k)} \right) g_i^{-1}. \tag{1}$$

[4] Binding the commitment to r_i to the commitments of the previous salts is useful to avoid broadcasting each commitment to each party. The only commitment that must be broadcast and seen by every party is c_n, which is a commitment to all the salts $r_1, \ldots, r_n$.

Algorithm 3: MS-GA[$\star$]

$\star\colon G \times X \to X$ is a cryptographic group action. The random oracles are $\mathsf{H}_0\colon \{0,1\}^* \to \{0,1\}^{2\lambda}, \mathsf{H}_1\colon \{0,1\}^* \to \{0,1\}^{\ell_{\mathsf{salt}}}$ and $\mathsf{H}_2^{(n)}\colon \{0,1\}^* \to [0,n]^{t(n)}$. During the execution of MuSign, each party maintains a list of active session identifiers in a list $\mathcal{S}$.

$\mathsf{Setup}(1^\lambda)$:
1: $x_0 \leftarrow\!\!\$\ X$
2: $\mathsf{pp} \leftarrow x_0$
3: **return** pp

$\mathsf{KGen}(\mathsf{pp} = x_0)$:
1: $g \leftarrow\!\!\$\ G$
2: $x \leftarrow g \star x_0$
3: **return** $(\mathsf{pk} = x, \mathsf{sk} = g)$

$\mathsf{MuVrfy}(L,\ m,\ \sigma)$:
1: $(x_1, \ldots, x_n) \leftarrow L$
2: $(\mathsf{ch}, r, z_1, \ldots, z_{t(n)}) \leftarrow \sigma$
3: **for** $j \leftarrow 1, \ldots, t(n)$ **do**
4: $\quad \tilde{x}_j \leftarrow z_j \star x_{\mathsf{ch}_j}$
5: $\tilde{x} \leftarrow (\tilde{x}_1, \ldots, \tilde{x}_{t(n)})$
6: **if** $\mathsf{H}_2^{(n)}(\tilde{x}, r, L, m) = \mathsf{ch}$ **then**
7: $\quad$ **return** 1
8: **else**
9: $\quad$ **return** 0

$\mathsf{MuSign}(\mathsf{sid},\ \mathsf{sk},\ \mathsf{pk},\ m,\ L)$:
1: $(\mathsf{pk}_1, \ldots, \mathsf{pk}_n) \leftarrow L$
2: Set i such that $\mathsf{pk}_i = \mathsf{pk}$
3: $x_i \leftarrow \mathsf{pk}$; $g_i \leftarrow \mathsf{sk}$
4: $r_i \leftarrow\!\!\$\ \{0,1\}^{\ell_{\mathsf{salt}}}$
5: $\mathsf{c}_{i-1}, \tilde{x}^{(i-1)} \twoheadleftarrow P_{i-1}$ // $\mathsf{c}_0 = \varepsilon, \tilde{x}_j^{(0)} = x_0$
6: $\mathsf{c}_i \leftarrow \mathsf{H}_0(\mathsf{c}_{i-1}, r_i)$
7: $\tilde{g}^{(i)} \leftarrow\!\!\$\ G^{t(n)}$
8: $\tilde{x}^{(i)} \leftarrow [\tilde{g}_j^{(i)} \star \tilde{x}_j^{(i-1)}]_{j \in [t(n)]}$
9: $\tilde{x}^{(i)}, \mathsf{c}_i \twoheadrightarrow P_{i+1}$
10: $\tilde{x} \leftarrow (\tilde{x}_1^{(n)}, \ldots, \tilde{x}_{t(n)}^{(n)})$ // If $i = n$ send to each party
11: $r_i \twoheadrightarrow P_k, r_k \twoheadleftarrow P_k, \forall k \neq i$
12: **if** $\exists j : \mathsf{c}_j \neq \mathsf{H}_0(\mathsf{c}_{j-1}, r_j)$ **then**
13: $\quad$ **return** $\perp$
14: $r \leftarrow \mathsf{H}_1(r_1, \ldots, r_n, \mathsf{sid})$
15: $\mathsf{ch} \leftarrow \mathsf{H}_2^{(n)}(\tilde{x}, r, L, m)$
16: **for** $j \leftarrow 1, \ldots, t(n)$ **do**
17: $\quad$ **if** $\mathsf{ch}_j = i \vee (\mathsf{ch}_j = 0 \wedge i = n)$ **then**
18: $\quad\quad \tilde{g}_j^{(k)} \twoheadleftarrow P_k, \forall k \neq i$
19: $\quad\quad z_j \leftarrow (\prod_{k=0}^{n-1} \tilde{g}_j^{(n-k)}) g_{\mathsf{ch}_j}^{-1}$ // $g_0 = e_G$
20: $\quad\quad z_j \twoheadrightarrow P_k, \forall k \neq i$
21: $\quad$ **else**
22: $\quad\quad \tilde{g}_j^{(i)} \twoheadrightarrow P_{\mathsf{ch}_j}$
23: $\quad\quad z_j \twoheadleftarrow P_{\mathsf{ch}_j}$
24: $\sigma \leftarrow (\mathsf{ch}, r, z_1, \ldots, z_{t(n)})$

In the case $\mathsf{ch}_j = 0$, it is agreed that the calculation of z_j is entrusted to the last user P_n. A high-level description of the multi-signature scheme is shown in Figure 1 for $\mathsf{ch} = 1$. The full description of the protocol is given in Algorithm 3.

4 Security Proof

In the following, we prove the MS-UF-CMA security of the protocol in Algorithm 3. In particular, we reduce security to the EUF-CMA of the centralized signature variant described in Section 2.3.

Theorem 2. *Let $\star\colon G \times X \to X$ be a cryptographic group action and let Π be as in Protocol 1. Let $\mathcal{A}$ be a* MS-UF-CMA *adversary against* MS-GA[$\star$] *in the random oracle model which makes q_S signing queries, q_H queries to the random*

oracles $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2$. *Then, there exists a* EUF-CMA *adversary* $\mathcal{B}$ *against* $\mathsf{FS}[\Pi]$ *issuing* $\mathsf{q_S}$ *signing queries and* $\mathsf{q_H}$ *queries to the random oracle* H'*, such that*

$$\mathsf{Adv}^{\text{MS-UF-CMA}}_{\mathsf{MS\text{-}GA}[\star]}(\mathcal{A}) \leq \mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{FS}[\Pi]}(\mathcal{B}) + \frac{\mathsf{q_S}(\mathsf{q_H} + \mathsf{q_S})}{2^{\ell_{\mathsf{salt}}}} + \frac{2\mathsf{q_S}\mathsf{q_H}}{2^{\ell_{\mathsf{M}}}},$$

and the running time of $\mathcal{B}$ *is about that of* $\mathcal{A}$*.*

Proof. In the following, we denote the random oracles and the signing oracle in the MS-UF-CMA game as $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2$ and $\mathsf{OMuSign}$. The EUF-CMA adversary $\mathcal{B}$ has access to a signing oracle OSign of the $\mathsf{FS}[\Pi]$ and an outer random oracle H'. After receiving the target public key $\mathsf{pk}^\star$ in the EUF-CMA game, $\mathcal{B}$ forwards $\mathsf{pk}^\star$ to $\mathcal{A}$.

At a high level, we show that controlling $n-1$ users in the multi-signature is no better than choosing $n-1$ ephemeral keys in the centralized signature $\mathsf{FS}[\Pi]$. We show that $\mathcal{B}$ can simulate $\mathsf{OMuSign}$ by querying OSign and programming the random oracle H_2 with the challenges provided by the outer random oracle H'. During a query to $\mathsf{OMuSign}$, the adversary may choose the value of the commitment $\tilde{x}$ of the sigma protocol by controlling the last user. However, the adversary can not control the value of the shared salt r, and $\mathcal{B}$ is able to program the random oracle H_2 before the adversary learns the value of r. In this way, the manipulation of the final commitment cannot influence the challenge and be exploited in parallel sessions.

The main focus of the reduction concerns the simulation of the random oracle H_2. After receiving a query containing the target public key among the participants' keys, $\mathcal{B}$ queries the outer random oracle H' and reprograms H_2 with a permutation of the received challenge. When $\mathcal{A}$ produces a valid signature, $\mathcal{B}$ will be able to map it into a signature for the centralized scheme $\mathsf{FS}[\Pi]$ using the keys of the users controlled by $\mathcal{A}$ as ephemeral keys.

In the following, we make the simplified assumption that before $\mathcal{A}$ outputs a forged signature, it makes a query on H_2, as would be done during the signature verification. Moreover, we assume that $\mathcal{A}$ always outputs a valid signature, and halts by returning $\perp$ otherwise. Notice that we can always modify $\mathcal{A}$ to behave this way by running the verification algorithm on the provided signature, and checking that the message provided was not queried to the signing oracle with the same set of signers.

More in detail, we prove the reduction by presenting a sequence of hybrid games, modifying the MS-UF-CMA game (Game 1) until it can be simulated by the EUF-CMA adversary $\mathcal{B}$ against the centralized signature $\mathsf{FS}[\Pi]$. In the following, we use the notation $\Pr[\mathsf{Game_n}(\mathcal{A}) = 1]$ to denote the probability that $\mathsf{Game_n}$ returns 1 when played by $\mathcal{A}$. The complete reduction is described in Algorithm 4.

Game_0 This is the initial strong MS-UF-CMA game against the $\mathsf{MS\text{-}GA}[\star]$ scheme, except that it uses programmable random oracles. At the start of the game, the challenger initializes three tables, $\mathsf{HT}_0, \mathsf{HT}_1, \mathsf{HT}_2$ for $\mathsf{H}_0, \mathsf{H}_1, \mathsf{H}_2$, respectively.

Algorithm 4: Full Reduction EUF-CMA $\implies$ MS-UF-CMA

$\mathcal{B}(\mathsf{pk}^\star)$:
$\mathcal{Q} \leftarrow \emptyset$; $\mathcal{S} \leftarrow \emptyset$; $\mathcal{M} \leftarrow \emptyset$
$(L, m, \sigma) \leftarrow\!\!\$\ \mathcal{A}^{\mathsf{O},\mathsf{OMuSign}}(\mathsf{pk}^\star)$
$(x_1, \ldots, x_n) \leftarrow L$
$(\mathsf{ch}, r, z_1, \ldots, z_{t(n)}) \leftarrow \sigma$
if $\mathsf{MuVrfy}(L, m, \sigma) \wedge \exists i : (x_i = x^\star \wedge (m, L) \notin \mathcal{Q})$ **then**
  Recover $\tilde{x}$ as in MuVrfy
  $m' \leftarrow \mathsf{MT}[\tilde{x}, r, L, m]$
  if $m' \in \mathcal{M}$ **then**
    raise $\mathsf{bad}_{\mathsf{mcol}}$
  $L' \leftarrow (x_2, \ldots, x_{i-1}, x_1, x_{i+1}, \ldots, x_n)$
  $\sigma' \leftarrow (L', \tilde{x}, z_1, \ldots, z_{t(n)})$
  return (m', σ')

$\mathsf{H}_0(r)$:
if $\mathsf{HT}_0[r] = \bot$ **then**
  $\mathsf{c} \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$
  $\mathsf{HT}_0[r] \leftarrow \mathsf{c}$
return $\mathsf{HT}_0[r]$

$\mathsf{H}_1(Q)$:
if $\mathsf{HT}_1[Q] = \bot$ **then**
  $r \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$
  $\mathsf{HT}_1[Q] \leftarrow r$
return $\mathsf{HT}_1[Q]$

$\mathsf{H}_2(Q = (\tilde{x}, r, L, m))$:
if $\mathsf{HT}_2[Q] \neq \bot$ **then**
  return $\mathsf{HT}_2[Q]$
$L \leftarrow (x_1, \ldots, x_n)$
if $\nexists i$ such that $x_i = \mathsf{pk}^\star$ **then**
  $\mathsf{ch} \leftarrow\!\!\$\ \mathsf{Ch}$
else
  $L' \leftarrow (x_2, \ldots, x_{i-1}, x_1, x_{i+1}, \ldots, x_n)$
  $m' \leftarrow\!\!\$\ \mathsf{M}$
  $\mathsf{ch}^{\mathcal{B}} \leftarrow \mathsf{H}'(L', \tilde{x}, m')$
  $\mathsf{ch} \leftarrow \pi_{1,i}(\mathsf{ch}^{\mathcal{B}})$
  $\mathsf{MT}[Q] \leftarrow m'$
$\mathsf{HT}_2[Q] \leftarrow \mathsf{ch}$
return ch

$\mathsf{OMuSign}(\mathsf{sid}, (m, L))$:
$\mathcal{Q} \leftarrow \mathcal{Q} \cup \{(m, L)\}$
$(x_1, \ldots, x_n) \leftarrow L$
if $\mathsf{sid} \in \mathcal{S} \vee \nexists i : x_i = x^\star$ **then**
  return $\bot$
$\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathsf{sid}\}$
$m' \leftarrow\!\!\$\ \mathsf{M}$
$\mathcal{M} \leftarrow \mathcal{M} \cup \{m'\}$
$(\mathsf{com}^{\mathcal{B}}, z_1^{\mathcal{B}}, \ldots, z_{t(n)}^{\mathcal{B}}) \leftarrow\!\!\$\ \mathsf{OSign}(m')$
$(\hat{x}_2^{\mathcal{B}}, \ldots, \hat{x}_n^{\mathcal{B}}, \tilde{x}_1^{\mathcal{B}}, \ldots, \tilde{x}_{t(n)}^{\mathcal{B}}) \leftarrow \mathsf{com}^{\mathcal{B}}$
$\mathsf{ch}^{\mathcal{B}} \leftarrow \pi_{1,i}(\mathsf{H}'(\mathsf{com}^{\mathcal{B}}, m'))$
$r_i \leftarrow\!\!\$\ \{0,1\}^\lambda$
$\mathsf{c}_{i-1}, \tilde{x}^{(i-1)} \leftarrow\!\!\diamond P_{i-1}$ // $\mathsf{c}_0 = \varepsilon, \tilde{x}_j^{(0)} = x_0$
$\mathsf{c}_i \leftarrow \mathsf{H}_0(\mathsf{c}_{i-1}, r_i)$
for $j \leftarrow 1, \ldots, t(n)$ **do**
  if $\mathsf{ch}_j^{\mathcal{B}} \neq i$ **then**
    $\tilde{g}_j^{(i)} \leftarrow\!\!\$\ G$
    $\tilde{x}_j^{(i)} \leftarrow \tilde{g}_j^{(i)} \star \tilde{x}_j^{(i-1)}$
  else
    $\tilde{x}_j^{(i)} \leftarrow \tilde{x}_j^{\mathcal{B}}$
$\tilde{x}^{(i)}, \mathsf{c}_i \rightarrow\!\!\diamond P_{i+1}$
$\tilde{x} \leftarrow (\tilde{x}_1^{(n)}, \ldots, \tilde{x}_{t(n)}^{(n)})$
Retrieve r_k such that $\mathsf{HT}_0[\mathsf{c}_{k-1}, r_k] = \mathsf{c}_k, \forall k \neq i$
$r \leftarrow \mathsf{H}_1(r_1, \ldots, r_n, \mathsf{sid})$
if $\mathsf{HT}_2[\tilde{x}, r, L, m] \neq \bot$ **then**
  raise $\mathsf{bad}_{\mathsf{hcol}}$
$\mathsf{HT}_2[\tilde{x}, r, L, m] \leftarrow \mathsf{ch}^{\mathcal{B}}$
$r_i \rightarrow\!\!\diamond P_k, \bar{r}_k \leftarrow\!\!\diamond P_k, \forall k \neq i$
if $\exists j : \bar{r}_j \neq r_j$ **then**
  return $\bot$
for $j \leftarrow 1, \ldots t(n)$ **do**
  if $\mathsf{ch}_j = i$ **then**
    $\tilde{g}_j^{(k)} \leftarrow\!\!\diamond P_k, \forall k \neq i$
    $z_j \leftarrow (\prod_{k=0}^{n-(i+1)} \tilde{g}_j^{(n-k)}) z_j^{\mathcal{B}}$
    $z_j \rightarrow\!\!\diamond P_k, \forall k \neq i$
  else if $\mathsf{ch}_j = 0 \wedge i = n$ **then**
    $\tilde{g}_j^{(k)} \leftarrow\!\!\diamond P_k, \forall k \neq i$
    $z_j \leftarrow (\prod_{k=0}^{n-1} \tilde{g}_j^{(n-k)})$
    $z_j \rightarrow\!\!\diamond P_k, \forall k \neq i$
  else
    $\tilde{g}_j^{(i)} \rightarrow\!\!\diamond P_{\mathsf{ch}_j}$

When a query Q for H_0 is received, if $\mathsf{HT}_0[Q] = \perp$ it uniformly samples $\mathsf{c} \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$ and stores $\mathsf{HT}_0[Q] \leftarrow \mathsf{c}$, finally it returns $\mathsf{HT}_0[Q]$ (similarly for H_1 and H_2). It follows that $\Pr[\mathsf{Game}_0(\mathcal{A}) = 1] = \mathsf{Adv}^{\text{MS-UF-CMA}}_{\text{MS-GA}[\star]}(\mathcal{A})$.

Game_1 This game is identical to Game_0, except that $\mathsf{OMuSign}$ aborts by raising $\mathsf{bad}_{\mathsf{hcol}}$ when the following happens: being $\tilde{x}$ the commitment and $r \leftarrow \mathsf{H}_1(r_1, \ldots, r_n, \mathsf{sid})$ the salt generated by $\mathcal{A}$ and $\mathsf{OMuSign}$ during the sign query $(\mathsf{sid}, (m, L))$, the challenger aborts the game if the random oracle H_2 was already queried at input $Q = (\tilde{x}, r, L, m)$, i.e. $\mathsf{HT}_2[Q] \neq \perp$. Otherwise, $\mathsf{OMuSign}$ samples $\mathsf{ch} \leftarrow\!\!\$\ [0,n]^{t(n)}$ and programs $\mathsf{HT}_2[Q] \leftarrow \mathsf{ch}$. It follows that $|\Pr[\mathsf{Game}_0(\mathcal{A}) = 1] - \Pr[\mathsf{Game}_1(\mathcal{A}) = 1]| \leq \Pr[\mathsf{bad}_{\mathsf{hcol}}]$.

Game_2 This game is identical to Game_1, except that $\mathsf{OMuSign}$ and H_2 are simulated as follows. At the start of the game, the challenger initializes an empty set $\mathcal{M}$, that will be used to track the messages queried by the simulator to OSign, and a look-up table MT used to map the queries to H_2 to the messages included in the queries to H'. In particular, when $\mathsf{OMuSign}$ receives a query, it samples a random message $m' \in \mathsf{M}$ and adds it to $\mathcal{M}$ before sending a sign query to OSign for m'. When H_2 receives a query $Q = (\tilde{x}, r, L, m)$ such that $\mathsf{pk}^\star \in L$, it samples a random message $m' \in \mathsf{M}$ and sets $\mathsf{MT}[Q] \leftarrow m'$ before querying H' on $(L', \tilde{x}, m')$. After the adversary outputs a valid signature $\sigma = (\mathsf{ch}, r, z_1, \ldots, z_{t(n)})$ on message m with users public keys $L = (x_1, \ldots, x_n)$, the challenger derives $\tilde{x}$ as in the execution of MuVrfy, and retrieves $m' \leftarrow \mathsf{MT}[\tilde{x}, r, L, m]$. If $m' \in \mathcal{M}$, the game aborts by raising $\mathsf{bad}_{\mathsf{mcol}}$. It follows that $|\Pr[\mathsf{Game}_1(\mathcal{A}) = 1] - \Pr[\mathsf{Game}_2(\mathcal{A}) = 1]| \leq \Pr[\mathsf{bad}_{\mathsf{mcol}}]$.

The reason why in Game_2, for the sign query to $\mathsf{OMuSign}$ and hash query to H_2, the simulator $\mathcal{B}$ is instructed to sample a random message m' resides in the definition of forgery in the multi-signature scheme and in the centralized scheme $\mathsf{FS}[\Pi]$. In particular, in the multi-signature game, $\mathcal{A}$ can produce a forgery on a message m signed on behalf of the public keys in L even if during the training with the oracle $\mathsf{OMuSign}$ it previously queried a signature for the same m but with a different set of signers L'. This does not hold for the centralized signature, where the forgery must be associated to a message that has never been queried to H'.

We now show that the EUF-CMA adversary $\mathcal{B}$ can simulate Game_2 as described in Algorithm 4. At the start of the game, $\mathcal{B}$ initializes an empty set $\mathcal{M} \leftarrow \emptyset$ that will store the queries to OSign. In the following, given a vector x, we will denote with $\pi_{i,j}(x)$ the permutation of elements with index i and j in x.

Random oracles queries. When a query Q_0 for H_0 is received, if $\mathsf{HT}_0[Q_0] = \perp$, $\mathcal{B}$ uniformly samples $\mathsf{c} \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$, stores $\mathsf{HT}_0[Q_0] \leftarrow \mathsf{c}$, and returns $\mathsf{HT}_0[Q_0]$. Similarly, when a query Q_1 for H_1 is received if $\mathsf{HT}_1[Q_1] = \perp$, $\mathcal{B}$ uniformly samples $\mathsf{c} \leftarrow\!\!\$\ \{0,1\}^{2\lambda}$, stores $\mathsf{HT}_1[Q_1] \leftarrow \mathsf{c}$, and returns $\mathsf{HT}_1[Q_1]$. Instead H_2, the random oracle employed to generate the challenge ch, is simulated as follows. Suppose a query $Q_2 = (\tilde{x}, r, L, m)$ for H_2 is received and $\mathsf{HT}_2[Q_2] = \perp$. Let $n = |L|$, if $\mathsf{pk}^\star \notin L$, i.e. the random oracle query do not refer to the public key that the forger must impersonate, $\mathcal{B}$ uniformly samples $\mathsf{ch} \leftarrow\!\!\$\ [0,n]^{t(n)}$

and returns it. Otherwise, suppose $L = (x_1, \ldots, x_n)$ such that $x_i = \mathsf{pk}^\star$ and let L' be the set of public keys in L after permuting the order of x_1 and x_i and subsequently removing x_i, i.e. $L' = (x_2, \ldots, x_{i-1}, x_1, x_{i+1}, \ldots, x_n)$. This way, L' can be used as a set of ephemeral keys for the centralized signature scheme $\mathsf{FS}[\Pi]$. Then, $\mathcal{B}$ samples a uniformly random message m' and queries $(L', \tilde{x}, m')$ to $\mathsf{H}'^{(n)}$, obtaining $\mathsf{ch}^{\mathcal{B}}$, which acts as the challenge of the centralized signature scheme. Next, it computes ch as the permutation $\pi_{1,i}(\mathsf{ch}^{\mathcal{B}})$, making it compatible with the public keys in L, and stores $\mathsf{MT}[Q_2] \leftarrow m'$ and $\mathsf{HT}_2[Q_2] \leftarrow \mathsf{ch}$. Finally, it returns $\mathsf{HT}_2[Q_2]$.

Signing queries. On a new query $Q = (\mathsf{sid}, (m, L))$, $\mathcal{B}$ runs MuSign up to Line 6. Suppose $L = (x_1, \ldots, x_n)$ such that $x_i = \mathsf{pk}^\star$. Then, it samples a uniformly random message $m' \leftarrow\!\!\$\ \mathsf{M}$, adds m' to $\mathcal{M}$, and queries OSign on m'. The signing oracle response is $(\mathsf{com}^{\mathcal{B}}, z_1^{\mathcal{B}}, \ldots, z_{t(n)}^{\mathcal{B}})$, with $\mathsf{com}^{\mathcal{B}} = (\hat{x}_2^{\mathcal{B}}, \ldots, \hat{x}_n^{\mathcal{B}}, \tilde{x}_1^{\mathcal{B}}, \ldots, \tilde{x}_{t(n)}^{\mathcal{B}})$. $\mathcal{B}$ will only use responses $z_j^{\mathcal{B}}$ from OSign that link x_i to $\tilde{x}_j^{\mathcal{B}}$, which correspond to the values 1 of the challenges sampled by the oracle. Hence, $\mathcal{B}$ computes the permutation of the challenge of the centralized signature $\mathsf{FS}[\Pi]$, obtaining $\mathsf{ch}^{\mathcal{B}} \leftarrow \pi_{1,i}(\mathsf{H}'^{(n)}(\mathsf{com}^{\mathcal{B}}, m'))$, that will be used to program the answer of H_2. Then, MuSign is simulated up to Line 10 as follows: for each $j \leftarrow 1, \ldots, t(n)$ it receives $\tilde{x}_j^{(i-1)}$. Then, if $\mathsf{ch}_j^{\mathcal{B}} \neq i$, it samples $\tilde{g}_j^{(i)} \leftarrow\!\!\$\ G$ and sets $\tilde{x}_j^{(i)} \leftarrow \tilde{g}_j^{(i)} \star \tilde{x}_j^{(i-1)}$. Otherwise, if $\mathsf{ch}_j^{\mathcal{B}} = i$, it sets $\tilde{x}_j^{(i)} \leftarrow \tilde{x}_j^{\mathcal{B}}$, since it knows the group element mapping the public key x_i to $\tilde{x}_j^{\mathcal{B}}$ from the centralized signature previously queried. Subsequently, before revealing r_i, $\mathcal{B}$ retrieves $r_1, \ldots, r_n$ from HT_0. If some c_j's were not obtained after a query to H_0 for (c_{j-1}, r_j), it follows the execution of MuSign and returns $\perp$ on Line 13. Next, it computes $r \leftarrow \mathsf{H}_1(r_1, \ldots, r_n, \mathsf{sid})$ and programs $\mathsf{HT}_2[\tilde{x}, r, L, m] \leftarrow \mathsf{ch}^{\mathcal{B}}$. Finally, the remainder of MuSign's execution is simulated as follows: for each $j \leftarrow 1, \ldots, t(n)$, if $\mathsf{ch}_j \neq i$, $\mathcal{B}$ reveals $\tilde{g}_j^{(i)}$. Otherwise, it receives $\tilde{g}_j^{(k)}$ for $k \neq i$ and computes $z_j \leftarrow \tilde{g}_j^{(n)} \cdot \ldots \cdot \tilde{g}_j^{(i+1)} z_j^{\mathcal{B}}$.

Eventually, $\mathcal{A}$ will output a valid signature $\sigma = (\mathsf{ch}, r, z_1, \ldots, z_{t(n)})$ for a message m under public keys $L = (x_1, \ldots, x_n)$. If $\mathcal{A}$ is winning the MS-UF-CMA game, then there exists an index $i \in [n]$ such that $\mathsf{pk}^\star = x_i$ and $(m, L) \notin \mathcal{Q}$. $\mathcal{B}$ can run MuVrfy up to Line 5 to recover $\tilde{x} = (\tilde{x}_1, \ldots, \tilde{x}_{t(n)})$. From our simplifying assumption on $\mathcal{A}$, σ is valid and $Q = (\tilde{x}, r, L, m)$ must have been queried to H_2. Then, $\mathcal{B}$ assigns to L' the set of public keys in L after permuting the order of x_1 and x_i and subsequently removing x_i, i.e. $L' = (x_2, \ldots, x_{i-1}, x_1, x_{i+1}, \ldots, x_n)$. Next, it retrieves $m' \leftarrow \mathsf{MT}[\tilde{x}, r, L, m]$ and aborts by raising $\mathsf{bad}_{\mathsf{mcol}}$ if $m' \in \mathcal{M}$. Otherwise, $\mathcal{B}$ wins the EUF-CMA game, returning the signature $(L', \tilde{x}, z_1, \ldots, z_{t(n)})$ on message m'. In fact, ch was obtained in the simulation of H_2 as $\mathsf{ch} = \pi_{1,i}(\mathsf{ch}')$, where $\mathsf{ch}' = \mathsf{H}'^{(n)}(L', \tilde{x}, m')$. Let $\hat{x}_k \leftarrow x_k$ for all $k \in [n], k \neq 1, i$ and let $\hat{x}_i \leftarrow x_1, \hat{x}_1 \leftarrow x_i$. For any $j \leftarrow 1, \ldots, t(n)$, it follows that:

$$z_j \star \hat{x}_{\mathsf{ch}'_j} = z_j \star x_{\mathsf{ch}_j} = \tilde{x}_j.$$

If none of the bad events happen, $\mathcal{B}$ perfectly simulate Game_2, and we obtain

$$\begin{aligned}\mathsf{Adv}^{\text{EUF-CMA}}_{\mathsf{FS}[\Pi]}(\mathcal{B}) &= \Pr[\mathsf{Game}_2(\mathcal{A}) = 1] \\ &\geq \mathsf{Adv}^{\text{MS-UF-CMA}}_{\mathsf{MS\text{-}GA}[\star]}(\mathcal{A}) - \Pr[\mathsf{bad}_{\mathsf{hcol}}] - \Pr[\mathsf{bad}_{\mathsf{mcol}}].\end{aligned}$$

$\mathcal{B}$ can simulate Game_2 with at most the same running time of $\mathcal{A}$ plus the time required for running MuVrfy.

In the following, we bound the probability of each bad event happening.

Probability of $\mathsf{bad}_{\mathsf{hcol}}$. The event $\mathsf{bad}_{\mathsf{hcol}}$ occurs on Line 25 of $\mathsf{OMuSign}$ on input $(\mathsf{sid}, (m, L))$ when, after obtaining the commitment $\tilde{x}$ and the salt $r \leftarrow \mathsf{H}_1(r_1, \ldots, r_n, \mathsf{sid})$, a value for $Q = (\tilde{x}, r, L, m)$ was already assigned in HT_2. The table HT_2 is populated by either $\mathsf{OMuSign}$ or H_2, so that its entries are at most $\mathsf{q_S} + \mathsf{q_H}$. The salt r is obtained from H_1 on inputs $r_1, \ldots, r_n$, where $r_k, k \neq i$ is provided by $\mathcal{A}$ and r_i is sampled uniformly random from $\{0,1\}^{\ell_{\mathsf{salt}}}$. The probability that a uniformly random r_i produces a collision with one of the entries is then at most $(\mathsf{q_S} + \mathsf{q_H})2^{-\ell_{\mathsf{salt}}}$. Since at most $\mathsf{q_S}$ are made to $\mathsf{OMuSign}$, then $\Pr[\mathsf{bad}_{\mathsf{hcol}}] \leq \mathsf{q_S}(\mathsf{q_S} + \mathsf{q_H})2^{-\ell_{\mathsf{salt}}}$.

Probability of $\mathsf{bad}_{\mathsf{mcol}}$. The event $\mathsf{bad}_{\mathsf{mcol}}$ occurs on Line 9 of the simulation of $\mathcal{B}$ when, after the adversary $\mathcal{A}$ outputs a valid signature $\sigma = (\mathsf{ch}, r, z_1, \ldots, z_{t(n)})$ on message m with users public keys $L = (x_1, \ldots, x_n)$, $\mathcal{B}$ derives $\tilde{x}$ as in the execution of MuVrfy, and retrieves $m' \leftarrow \mathsf{MT}[\tilde{x}, r, L, m]$, the message sampled during a random oracle query to H_2, such that $m' \in \mathcal{M}$. There are two possibilities that can cause the $\mathsf{bad}_{\mathsf{mcol}}$ event: either in H_2 (Line 8) if it samples $m' \in \mathsf{M}$ that is already in $\mathcal{M}$, or in $\mathsf{OMuSign}$ (Line 6) if it samples $m' \in \mathsf{M}$ such that it is already a value in MT. The set $\mathcal{M}$ is populated by $\mathsf{OMuSign}$, so that its entries are at most $\mathsf{q_S}$. Since $|\mathsf{M}| = 2^{\ell_{\mathsf{M}}}$, the probability that a uniformly random $m' \in \mathsf{M}$ produces a collision with one of the entries of $\mathcal{M}$ is then at most $\mathsf{q_S}/|\mathsf{M}| \leq \mathsf{q_S}/2^{\ell_{\mathsf{M}}}$. Since at most $\mathsf{q_H}$ queries are made to H_2, the probability of the first occurrence is at most $\mathsf{q_H q_S}/2^{\ell_{\mathsf{M}}}$. Similarly, the table MT is populated by H_2 with at most $\mathsf{q_H}$ entries, and the probability that a uniformly random $m' \in \mathsf{M}$ produces a collision with one of the entries of MT is at most $\mathsf{q_H}/2^{\ell_{\mathsf{M}}}$. Since at most $\mathsf{q_S}$ queries are made to $\mathsf{OMuSign}$, the probability of the second occurrence is at most $\mathsf{q_H q_S}/2^{\ell_{\mathsf{M}}}$. Therefore, we obtain $\Pr[\mathsf{bad}_{\mathsf{mcol}}] \leq 2\mathsf{q_S q_H}/2^{\ell_{\mathsf{M}}}$.

Combining the previous bound on bad events, we obtain the claimed estimate of $\mathsf{Adv}^{\text{MS-UF-CMA}}_{\mathsf{MS\text{-}GA}[\star]}(\mathcal{A})$. □

Concurrent Executions. Note that the security proof reduces the security of the multi-signature to the security of a centralized signature $\mathsf{FS}[\Pi]$, which is concurrently secure since it does not require interactions between parties. Also, the simulation does not require $\mathcal{B}$ to rewind the adversary during the execution of the training phase, when the adversary queries the sign oracle and builds signatures of chosen messages with its support. The rewinding is executed only once, when the adversary of the multi-signature produces its forgery. This means

that the execution time of the simulator $\mathcal{B}$ is polynomial in the execution time of the adversary, which is polynomial in λ.

Identifiable Abort. Our protocol can be adapted to allow the signers to identify when a party misbehaves. If during the construction of the commitment, each party broadcasts their partial commitment, then in an eventual failure of the signature protocol the honest party will always be able to identify at least one malicious party for each signing protocol execution. In fact, when the commitments are opened, one can check that the group element $\tilde{g}_i$ of the parties P_i which are not selected by the challenge actually maps the previous partial commitment $\tilde{x}_{i-1}$ to $\tilde{x}_i$.

5 Signature Optimizations

In this section, we apply some standard optimization techniques to MS in order to decrease the size of the multi-signature. Note that when defining the multi-signature protocol, it is necessary to ensure that the centralized signature keys are compatible with the interactive protocol. Therefore, all optimizations that do not intervene directly on the keys are potentially applicable.

One of the main efficiency measures for the multi-signature scheme is the *compression rate*, i.e., the reduction in the length of the signature aggregation of n users compared to the trivial concatenation of n individual signatures. Let Σ_n be the multi-signature of n users and let σ be the individual signature of the centralized scheme. The compression rate of n signatures is defined as $\tau(n) = 1 - \frac{|\Sigma_n|}{n \cdot |\sigma|}$. In order to optimize the compression rate, we reduce the size of Σ_n without affecting the security of the scheme.

Consider a group action $(G, X, \star)$ and a security parameter λ. The non-optimized version of the Σ-protocol Π is described in Algorithm 1. In the following, we assume that a group element can be represented with strings of ℓ_{G} bits, while a challenge for a single instance of the protocol can be represented with a single bit. Since Π is commitment-recoverable, we are only interested in the size of the restricted transcript $(\mathsf{ch}, \mathsf{rsp})$, where the response rsp is an element of the group. We already discussed that to achieve negligible knowledge error, it is required to parallel repeat Π for $t = \lambda$ times, obtaining $\Pi_1 = \Pi^t$. We present each optimization as a successive transformation applied on top of Π_1, analysing the updated parameters (e.g., the number of repetitions of the protocol) and the size of the signature obtained by applying Fiat-Shamir. The centralized signature associated with Π_i is denoted with $\mathsf{FS}[\Pi_i]$, while the multi-signature is denoted with $\mathsf{MS}[\Pi_i]$.

The bit size of the non-optimized centralized signature $\mathsf{FS}[\Pi_1]$ is given by:

$$|\mathsf{ch}| + |\mathsf{rsp}| = \lambda + \lambda \ell_{\mathsf{G}}.$$

In the non-optimized version of the multi-signature $\mathsf{MS}[\Pi_1]$, the signature produced by n users is given by $\Sigma_n = (\mathsf{ch}, r, z_1, \ldots, z_{t(n)})$. The expected sizes (in

bits) of the elements of Σ_n are expressed by

$$|\mathsf{ch}| = \log_2(n+1)t(n), \quad |r| = 2\lambda, \quad |z| = t(n)\ell_{\mathsf{G}}.$$

5.1 Compression of Random Elements

A basic technique used to reduce the size of the signature $\mathsf{FS}[\Pi^t]$ is based on the following simple observation: when $\mathsf{ch}_i = 0$ the response for the i-th repetition is just the uniformly random group element $\tilde{g}_i \in G$ used to build the commitment. Therefore, in practice, $\tilde{g}_i$ can be replaced by a short random seed s_i of size λ which is used as the input of a Pseudorandom Number Generator PRNG to generate the group element. Then, every time $\mathsf{ch}_i = 0$, the signer can set $\mathsf{rsp}_i \leftarrow s_i$ saving $\ell_{\mathsf{G}} - \lambda$ bits for each 0 challenge. If the challenge array is uniformly sampled from $\{0,1\}^t$, then the expected number of bits saved using this optimization is $t(\ell_{\mathsf{G}} - \lambda)/2$. This technique was already adopted in the context of isogenies [68] and later employed in group action-based signatures.

When random responses are compressed using seed of length λ, it is required to also employ a random salt of length at least 2λ to prevent collision search attacks [25]. This corresponds to a slight increase in the signature size, which now includes the random salt.

Let Π_2 be the protocol obtained by applying the aforementioned optimization to Π_1, then the expected bit size of the signature for $\mathsf{FS}[\Pi_2]$ is given by

$$|r| + |\mathsf{ch}| + |\mathsf{rsp}| = 2\lambda + \lambda + t\left(\frac{1}{2}\lambda + \frac{1}{2}\ell_{\mathsf{G}}\right),$$

where the terms of $|\mathsf{rsp}|$ correspond to the size of the responses to non-zero and zero challenges, respectively. Notice that this holds only on average and that in the worst case, the size of the response can grow up to $t\ell_{\mathsf{G}}$.

When we consider the multi-signature described in Algorithm 3, things get more complicated because the signers $P_1, \ldots, P_n$ build the responses $z_i, i \in [t(n)]$, in a round-robin fashion by multiplying the group elements that they have generated as described in Equation (1). Therefore, when the challenge $\mathsf{ch}_i = 0$, no contribution is required from the secret keys of the participants, and the response to the i-th repetition of the sigma protocol can be encoded in two possible ways:

- with the full list of seeds $(s_1, \ldots, s_n)$ corresponding to the group element of each party, which are multiplied to retrieve the response z_i;
- with the direct encoding of the group element z_i.

This means that the use of seeds is convenient as long as the representation of an element in G is heavier than the concatenation of n seeds, i.e. $n\lambda < \ell_{\mathsf{G}}$, and if the expected number of zeros in a $t(n)$-bit long challenge is $\frac{t(n)}{n+1}$, then the expected number of bits saved by using this technique is $\frac{t(n)(\ell_{\mathsf{G}} - n\lambda)}{n+1}$.

Similarly, the challenge ch can be expanded from a digest $d \in \{0,1\}^{2\lambda}$ obtained from H_2 in Algorithm 3.

Applying the aforementioned optimizations, the expected sizes (in bits) of the challenge and the response array for $\mathsf{MS}[\Pi_2]$ are approximated by

$$|\mathsf{ch}| = 2\lambda, \quad |r| = 2\lambda, \quad |z| = \frac{t(n)}{n+1}(n\ell_{\mathsf{G}} + \ell_{\mathsf{seeds}}), \quad \ell_{\mathsf{seeds}} = \min\{n\lambda, \ell_{\mathsf{G}}\}$$

where the terms of $|z|$ correspond to the size of the responses to non-zero and zero challenges, respectively.

5.2 Seed Trees

A binary tree of seeds (*seed tree*) can be used to reduce the communication cost of the seeds used to construct the random elements of the group [15]. The tree is computed by taking a master seed of length λ as the root of the tree. Then, from each node, two children are generated from the output of length 2λ of a PRNG taking as input the value of the node. To represent t seeds, this process is repeated for $\lceil \log(t) \rceil$ times so that the tree has $2^{\lceil \log(t) \rceil} \geq t$ leaves having the seeds as values. The seeds corresponding to a subset of the leaves can be revealed by sharing a suitable subset of parent nodes and computing the corresponding leaves. In particular, to communicate the value of all the t seeds except for those indexed by a subset of $\{1, \ldots, t\}$ of size ω, it is enough to send the values of the following number of nodes:

$$2^{\lceil \log_2(\omega) \rceil} + \omega(\lceil \log_2(t) \rceil - \lceil \log_2(\omega) \rceil - 1).$$

The communication cost of using a seed tree is advantageous when there are at least $\frac{t}{2}$ zero challenges. This can be enforced by sampling the challenges according to a fixed-weight distribution, as shown in the next optimization.

In the multi-signature, the expected number of zero challenges is $\frac{t(n)}{n+1}$. Therefore, the use of a seed tree is already ineffective for $n = 2$ users.

5.3 Unbalanced Challenges

When random responses corresponding to $\mathsf{ch}_i = 0$ are compressed with a seed as described above, the resulting size is much smaller than when the challenge is non-zero. A standard technique to exploit this imbalance is to modify the distribution of challenges to increase the number of zero challenges in ch [15,66,8]. More precisely, with this optimization, we choose parameters t, ω such that there are exactly ω non-zero challenges among t execution of the protocol. When the challenge space is binary, as in Algorithm 1, the number of challenges in $\{0,1\}^t$ having exactly ω components equal to 1 is $\binom{t}{\omega}$. Therefore, the choice of t, ω must be made so that

$$\binom{t}{\omega}^{-1} \leq 2^{-\lambda}. \tag{2}$$

The security of this solution is well understood in the case of special-sound Σ-protocol since, as for parallel repetition, the resulting protocol is still special-sound with challenge space of cardinality $\binom{t}{\omega}$. However, this is not trivial to

extend to generic k-special-sound Σ-protocol of multi-round protocols and was only recently proved secure in [10].

By applying this optimization on Π_2, for each response we send ω group elements corresponding to $\mathsf{ch}_i = 1$. The remaining $t - \omega$ group elements corresponding to $\mathsf{ch}_i = 0$ are replaced by random seeds that can be further compressed using the Seed Tree optimization. We obtain the protocol Π_3, the size of the signature associated with $\mathsf{FS}[\Pi_3]$ is given by:

$$|r| + |\mathsf{ch}| + |\mathsf{rsp}| = 2\lambda + \lambda + \left(2^{\lceil \log_2(\omega) \rceil} + \omega(\lceil \log_2(t) \rceil - \lceil \log_2(\omega) \rceil - 1)\right) \cdot \lambda + \omega \ell_{\mathsf{G}}.$$

Notice that when $\lambda \ll \ell_{\mathsf{G}}$, using this optimization with an appropriate choice of ω compresses the signature considerably. On the other hand, to maintain the same security level, t must be chosen according to Equation (2). This typically results in an increase in the number of parallel repetitions, leading to a trade-off between the size of the signature and the efficiency of the signing and verification process.

In the multi-signature of Section 3, the use of fixed-weight challenges can still be useful to decrease the cheating probability of the adversary. In fact, the best strategy for an adversary is to control all parties except the target user P_i and to have a challenge with few components $\mathsf{ch}_j = i$. To make this possibility negligible, a large number of parallel repetitions $t(n)$ must be chosen, making the signature inefficient. As a countermeasure, we can consider challenges where each value $i \in [1, n]$ appears the same number of times. More in detail, for each $n \in \mathbb{N}$, we choose t, ω such that the challenges are elements of $[0, n]^t$ with exactly ω components equal to i, for each $i \in [1, n]$, and the remaining $t - n\omega$ components are equal to 0. Let $\mathsf{Ch}_n^{t,\omega}$ denote the challenge set mentioned above. The number of challenges in $\mathsf{Ch}_n^{t,\omega}$ is

$$\frac{t!}{(t - n\omega)!(\omega!)^n}.$$

Once a commitment is fixed, let $\eta_{t,\omega}$ be the maximum number of challenges in $\mathsf{Ch}_n^{t,\omega}$ an adversary can answer to without knowing the private key (i.e. from the responses to such challenges it would not be possible to extract the witness). Then t, ω must be chosen such that

$$\frac{\eta_{t,\omega}}{|\mathsf{Ch}_n^{t,\omega}|} \leq 2^{-\lambda}. \tag{3}$$

Lemma 1. *Given $n \in \mathbb{N}$, the value $\eta_{t,\omega}$ can be expressed as*

$$\max_{0 \leq k \leq n-1} \frac{(t - (n-k)\omega)!}{(t - n\omega)!(\omega!)^k} \frac{((n-k)\omega)!}{(\omega!)^{n-k}}. \tag{4}$$

Proof. Suppose w.l.o.g. that the target user is P_1, so that, without knowing their private key, an adversary cannot answer two challenges with 0 and 1 in the same component. In the following, let $\mathsf{Ch}_n = \{0, \ldots, n\}$ and let Ch_n^t be the set of challenge strings of length t. For a subset $C \subset \mathsf{Ch}_n^t$, let $\mathcal{H}_C$ be the undirected graph whose vertices are the elements of Ch_n and in which, for any $x, y \in \mathsf{Ch}_n$,

there is a link between x and y in $\mathcal{H}_C$ if and only if there exist two challenge strings $\mathsf{ch}, \mathsf{ch}' \in C$ such that $\mathsf{ch}_i = x$ and $\mathsf{ch}'_i = y$ for some index $1 \leq i \leq t$.

Let $\mathcal{C}$ be the set of all subsets C of $\mathsf{Ch}_n^{t,\omega}$ such that 0 and 1 are not connected in $\mathcal{H}_C$. It follows that $\eta_{t,\omega}$ is the maximum cardinality among the sets in $\mathcal{C}$. Given a set $C \in \mathcal{C}$, let k be the number of challenges $\alpha_1, \ldots, \alpha_k \in \mathsf{Ch}_n \setminus \{1\}$ for which there is a path between 0 and α_i in $\mathcal{H}_C$. The remaining $n-k-1$ challenges $\beta_1, \ldots, \beta_{n-k-1} \in \mathsf{Ch}_n \setminus \{0,1\}$ can either be all connected to 1 or form smaller connected components. For any $\mathsf{ch} \in \mathsf{Ch}_n^{t,\omega}$, there are exactly ω components of ch equal to α_i or β_j, and $t - n\omega$ components equal to 0. Therefore, in C we can have at most $\frac{(t-(n-k)\omega)!}{(t-n\omega)!(\omega!)^k}$ choices for the entries that have a path to 0. The remaining $(n-k)\omega$ entries, can have at most $\frac{((n-k)\omega)!}{(\omega!)^{n-k}}$ choices when $\beta_1, \ldots, \beta_{n-k-1}$ are all connected to 1. Therefore, the maximal size of a set $C \in \mathcal{C}$ is given in Equation (4). □

In the following, we choose $t = (n+1)\omega$, so that each value in $\{0, \ldots, n\}$ appears exactly ω times.

Lemma 2. *Given $n \in \mathbb{N}$, let $\eta_\omega = \eta_{(n+1)\omega,\omega}$. Then*

$$\eta_\omega = \frac{(n\omega)!}{(\omega!)^n}.$$

Proof. Substituting $t = (n+1)\omega$ in Equation (4), we obtain

$$\eta_\omega = \eta_{(n+1)\omega,\omega} = \max_{0 \leq k \leq n-1} \frac{((k+1)\omega)!}{(\omega!)^{k+1}} \frac{((n-k)\omega)!}{(\omega!)^{n-k}}.$$

Consider the discrete function $f(k)$ taking values in $\{0, \ldots, n-1\}$, defined by

$$f(k) = \frac{((k+1)\omega)!}{(\omega!)^{k+1}} \frac{((n-k)\omega)!}{(\omega!)^{n-k}}.$$

Notice that $f(k) = f(n-1-k)$, it is then sufficient to prove that f is decreasing for $k \leq \lfloor (n-1)/2 \rfloor$. In fact, for any k, it holds that

$$\frac{f(k)}{f(k+1)} = \frac{((k+1)\omega)!}{((k+2)\omega)!} \frac{((n-k)\omega)!}{((n-k-1)\omega)!} = \prod_{i=1}^{\omega-1} \frac{(n-k)\omega - i}{(k+2)\omega - i}.$$

Notice that for any term in the product, it holds that

$$(n-k)\omega - i \geq (k+2)\omega - i \iff k \leq \left\lfloor \frac{n-2}{2} \right\rfloor.$$

Therefore $\eta_\omega = f(0) = f(n-1) = \frac{(n\omega)!}{(\omega!)^n}$. □

If we substitute the value of η_ω from previous lemma in Equation (3), then the choice of ω should be made such that

$$2^{-\lambda} \geq \frac{\eta_\omega}{|\mathsf{Ch}_n^{(n+1)\omega,\omega}|} = \binom{(n+1)\omega}{\omega}^{-1}.$$

The choice of ω is made with the aim of minimizing the size of the response array z, where

$$|z| = n\omega\ell_{\mathsf{G}} + \omega\ell_{\mathsf{seeds}}. \tag{5}$$

In Section 6 we provide a concrete analysis of the optimal values for ω for the selected signature schemes.

5.4 Multiple Public Keys

It is possible to consider multiple public keys for each user in order to reduce the size of the signature. This is a standard technique [36] employed in group action-based signatures to achieve a trade-off between signature size and public key size. Unlike previous optimizations, in this case the underlying security assumption is modified. The signer generates $s-1$ public keys associated to different private keys, and the challenge space is extended from $\{0,1\}$ to $\{0,\dots,s-1\}$ so that a challenge can select one of the keys. The response is then generated using the relevant private key, exhibiting a group element that maps the selected public key to the commitment. The security assumption underlying the signature is modified to the following

Definition 6. *Given a group action $(G, X, \star)$, the Multiple Group Action Inverse Problem (MGAIP$_\star$) takes as input a collection of elements $x_0, \dots, x_{s-1}$ in the orbit $G \star x_0$, and asks to find $g \in G$ such that $x_i = g \star x_j$, for some $i \neq j$.*

This problem is still hard, and reduces tightly to GAIP, for instance, a proof is given in [8, Theorem 3] and it can be easily generalized. Since the challenge space of the single instance is extended from a binary space to one of s elements, the soundness error is reduced to $1/s$. Clearly, this also reduces the number of repetitions required to $t = \lceil \lambda / \log(s) \rceil$.

To obtain a soundness error negligible in λ with a single instance of the protocol would require generating an exponential number of keys ($s = 2^\lambda$). For this reason, this approach is usually combined with the previous optimizations to reduce the size of the signature with a limited increase in the public-key size. Notice that when $\mathsf{ch}_i = 0$, the response is still a random group element that can be replaced with a short seed; while for $\mathsf{ch}_i \neq 0$ a full group element is required. We can then apply the fixed-weight optimization to send ω group elements corresponding to $\mathsf{ch}_i \neq 0$ and $t - \omega$ short seeds corresponding to $\mathsf{ch}_i = 0$. Therefore, the choice of t, ω must be made so that

$$\left[\binom{t}{\omega} (s-1)^\omega \right]^{-1} \leq 2^{-\lambda}. \tag{6}$$

By combining all previous optimizations, we obtain the protocol Π_4. The size of the signature associated with $\mathsf{FS}[\Pi_4]$ is given by:

$$|r| + |\mathsf{ch}| + |\mathsf{rsp}| = 2\lambda + \lambda + \left(2^{\lceil \log_2(\omega) \rceil} + \omega(\lceil \log_2(t) \rceil - \lceil \log_2(\omega) \rceil - 1)\right) \cdot \lambda + \omega\ell_{\mathsf{G}}. \tag{7}$$

Notice that this is the same as the fixed-weight case, but here the number of repetitions will be smaller due to the increased number of public keys, resulting in a more compact signature.

Using this optimization in the multi-signature requires minor modifications to the MS protocol described in Algorithm 3. In fact, the protocol already allows multi-bit challenges to select a specific user's key. It is, therefore, sufficient to extend the challenge space so that one of the user's keys can be selected. Notice, however, that this optimization modifies the public keys of the underlying signature, and is therefore applicable only if the signature scheme provides for it. The changes described below will then be used in Section 6 for the parameterization of signatures using multiple public keys.

In the following, we combine the use of multiple public keys with the unbalanced challenge optimization of the previous section, evaluating its impact on soundness error and signature size. Concretely, suppose each user P_i has s public keys $x_i^{(0)}, \ldots, x_i^{(s-1)} \in X$, where $x_i^{(0)} = x_0$. For a fixed $n \in \mathbb{N}$, let $\mathsf{Ch}_{n,s} = \{0, \ldots, n(s-1)\}$ be the challenge space of the single instance, where 0 identifies x_0 and $k = (i-1)(s-1) + j$ identifies $x_i^{(j)}$ of user P_i, with $1 \leq i \leq n$ and $1 \leq j \leq s-1$. Similarly to the single key case, we choose t, ω such that the challenges are elements of $\mathsf{Ch}_{n,s}^{t}$ with exactly ω components corresponding to the i-th user, and the remaining $t - n\omega$ components are equal to 0. Let $\mathsf{Ch}_{n,s}^{t,\omega}$ denote the challenge set described above. The number of challenges in $\mathsf{Ch}_{n,s}^{t,\omega}$ is

$$\frac{t!}{(t-n\omega)!(\omega!)^n}(s-1)^{n\omega}.$$

Let $\eta_{t,\omega}$ be the maximum number of challenges in $\mathsf{Ch}_{n,s}^{t,\omega}$ an adversary can answer to without knowledge of the private key. Then t, ω must be chosen such that

$$\frac{\eta_{t,\omega}}{|\mathsf{Ch}_{n,s}^{t,\omega}|} \leq 2^{-\lambda}. \tag{8}$$

As in the case of the single key, we simplify by choosing $t = (n+1)\omega$. By adapting *Lemma* 1 and *Lemma* 2, we obtain that

$$\eta_\omega = \max_{k_1+\ldots+k_s=n-1} \prod_{i=1}^{s} \frac{((k_i+1)\omega)!}{(\omega!)^{k_i+1}}(s-1)^{k_i\omega} = \frac{(n\omega)!}{(\omega!)^n}(s-1)^{(n-1)\omega}.$$

If we substitute the value of η_ω in Equation (8), then the choice of ω should be made such that

$$2^{-\lambda} \geq \frac{\eta_\omega}{|\mathsf{Ch}_{n,s}^{(n+1)\omega,\omega}|} = \binom{(n+1)\omega}{\omega}^{-1}(s-1)^{-\omega}. \tag{9}$$

With respect to Equation (5), the expression for the size of the response size remains unchanged. On the other hand, as s increases, we can choose a smaller ω in order to obtain a more compact signature.

6 Instantiation and Evaluation

In this section, we will provide concrete applications of the multi-signature scheme described in Section 3 to some digital signature schemes based on group actions, namely LESS, MEDS, and ALTEQ. We evaluate the efficiency of the scheme by measuring the compression rate, as defined in Section 5.

Consider a group action $(G, X, \star)$ and a security parameter λ. Using the optimizations described in the previous section, in Table 1 we summarize the bit length of N centralized signatures and the multi-signature of N users[5] associated with the group action. In more detail, we assume that for both centralized and multi-signature, we have the same size for random salts $\ell_{\mathsf{salt}} = 2\lambda$, outputs of the random oracle $\ell_{\mathsf{digest}} = 2\lambda$, seeds for random elements λ, and group elements ℓ_{G}. For the centralized signature, the number of repetitions t and fixed-weight parameter of the challenges ω are chosen according to Equation (6) and are reported in the parameter sets of each scheme. In the multi-signature, the fixed-weight parameter ω' is chosen according to Equation (9) and depends on the number of signers. Notice that, as discussed in Section 5.2, only the centralized signature can exploit the use of seed trees. It follows that the length of N concatenated signatures is given by $N \cdot |\sigma| = N \cdot (\ell_{\mathsf{salt}} + \ell_{\mathsf{digest}} + \omega\ell_{\mathsf{G}} + \ell_{\mathsf{seeds}})$, i.e.,

$$N \cdot \left(4\lambda + \omega\ell_{\mathsf{G}} + (2^{\lceil \log_2(\omega) \rceil} + \omega(\lceil \log_2(t) \rceil - \lceil \log_2(\omega) \rceil - 1)) \cdot \ell_{\mathsf{tree_seed}}\right).$$

On the other hand, compression of random elements can also be exploited in the case of multi-signatures if the number of signers N is less than $\lfloor \ell_{\mathsf{G}}/\lambda \rfloor$. It follows that the length of a multi-signature of N users is given by:

$$|\Sigma_N| = \ell_{\mathsf{salt}} + \ell_{\mathsf{digest}} + N\omega'\ell_{\mathsf{G}} + \ell_{\mathsf{seeds}} = 4\lambda + N\omega'\ell_{\mathsf{G}} + \min\{\omega'\ell_{\mathsf{G}}, N\omega'\lambda\}.$$

We observe that in both cases, the length of the signatures is dominated by $N\omega\ell_{\mathsf{G}}$ (resp. $N\omega'\ell_{\mathsf{G}}$). The fixed-weight parameter ω for the centralized signature is fixed for each set of parameters so that the term $N\omega\ell_{\mathsf{G}}$ grows linearly with N. On the other hand, the fixed-weight parameter ω' for the multi-signature depends on the number of signers, and is given by:

$$\omega' = \arg\min_{\omega} \left\{ \binom{(N+1)\omega}{\omega}^{-1} (s-1)^{-\omega} \leq 2^{-\lambda} \right\}.$$

We can use the following bound from [33, Theorem 11.1.3] to find a simplified expression of ω':

$$\binom{n}{k} \leq 2^{nH_b(k/n)},$$

where $H_b(n) = -n\log_2(n) - (1-n)\log_2(1-n)$ is the *binary entropy function*. Therefore, for any $N > 1$,

$$2^{(N+1)\omega' H_b(1/(N+1))}(s-1)^{\omega'} \geq \binom{(N+1)\omega'}{\omega'}(s-1)^{\omega'} \geq 2^{\lambda},$$

[5] In this section we use N to denote the number of signers as n is often used as an internal parameter for the underlying signature scheme.

Table 1: Data sizes (in bits) and choice of parameters comparison between single-signature and multi-signature schemes with N users.

	Centralized Signature	**Multi-signature**
ℓ_{salt}, ℓ_{digest}	2λ	
$\ell_{\mathsf{tree_seed}}$	λ	
ℓ_{G}	Byte size of elements in G	
s	Number of public keys	
ω	$\min\left\{\omega \;\middle|\; \binom{t}{\omega}^{-1}(s-1)^{-\omega} \leq 2^{-\lambda}\right\}$	
ℓ_{seeds}	$(2^{\lceil \log_2(\omega)\rceil} + \omega(\lceil \log_2(t)\rceil - \lceil \log_2(\omega)\rceil - 1)) \cdot \ell_{\mathsf{tree_seed}}$	$\min\{\omega\ell_{\mathsf{G}}, N\omega\lambda\}$
ℓ_{sig}	$N \cdot (\ell_{\mathsf{salt}} + \ell_{\mathsf{digest}} + \omega\ell_{\mathsf{G}} + \ell_{\mathsf{seeds}})$	$\ell_{\mathsf{salt}} + \ell_{\mathsf{digest}} + N\omega\ell_{\mathsf{G}} + \ell_{\mathsf{seeds}}$

so that $(N+1)\omega' H_b(1/(N+1)) + \omega' \log_2(s-1) \geq \lambda$, and

$$\omega' = \left\lceil \frac{\lambda}{(N+1)H_b(1/(N+1)) + \log_2(s-1)} \right\rceil.$$

It follows that a rough approximation for the compression rate $\tau(N)$ is given by

$$\tau(N) = 1 - \frac{|\Sigma_N|}{N \cdot |\sigma|} \approx 1 - \frac{N\omega'\ell_{\mathsf{G}}}{N\omega\ell_{\mathsf{G}}} = 1 - \frac{1}{\omega}\left\lceil \frac{\lambda}{(N+1)H_b(1/(N+1)) + \log_2(s-1)} \right\rceil.$$

The above expression provides an initial estimate of the effectiveness of using multi-signature. In particular, we will have higher compression as the number of signers increases, and higher maximum compression for parameter sets using a fixed-weight parameter that is not too low.

A second metric relevant to the analysis of multi-signature efficiency concerns the computational costs of producing a signature. In the case of group action-based signatures, the main parameter affecting the performance of the signing and verification process is the number t of protocol repetitions. In particular, the computational cost of producing a signature linearly increases with t. For many cryptographic group actions, the bottleneck is given by the evaluation of the action $\star$. Such a computation is performed once every repetition, hence, their total number is linear in t. Similarly to the fixed-weight parameter, the number of repetitions for the centralized signature is fixed for each set of parameters, and when N signatures are concatenated, the single instance of the protocol is repeated $N \cdot t$ times distributed among N users. The number of repetitions for the multi-signature is determined by the number of signers and is given by $(N+1)\omega'$. Applying a similar analysis as above, the repetition rate, i.e., the rate of reduction in the number of iterations between centralized and multi-signature, is given by

$$1 - \frac{(N+1)\omega'}{N \cdot t} \approx 1 - \frac{\omega'}{t} = 1 - \frac{1}{t}\left\lceil \frac{\lambda}{(N+1)H_b(1/(N+1)) + \log_2(s-1)} \right\rceil.$$

Table 2: Minimum number of users N to achieve a signature compression, compression rate, repetition rate and fixed-weight parameter ω' for 50 and 100 users for the LESS parameters sets.

Set	**Min.** N	**Compr. Rate**		**Repet. Rate**		**FW Par.** ω'	
		$N = 50$	$N = 100$	$N = 50$	$N = 100$	$N = 50$	$N = 100$
LESS-1b	5	0.469	0.529	0.922	0.930	19	17
LESS-1i	6	0.351	0.437	0.933	0.942	16	14
LESS-1s	5	0.342	0.396	0.928	0.934	14	13
LESS-3b	11	0.336	0.413	0.962	0.967	28	25
LESS-3s	13	0.259	0.354	0.972	0.975	25	22
LESS-5b	15	0.281	0.365	0.972	0.975	37	33
LESS-5s	11	0.281	0.374	0.963	0.968	33	29

The previous expression compares performance only in terms of iterations of the underlying protocol. In the case of an interactive multi-signature, however, it is also necessary to consider the cost associated with the communication rounds between the parties during the signing process. The analysis of this cost is beyond the scope of this work and is not further analysed.

6.1 LESS

LESS [17,8] is a signature scheme based on the hardness of the *Linear Code Equivalence*. The LESS group action $\star_{\mathsf{LEP}}$ is given by the action of the *monomial group* $\mathrm{Mon}(n, q)$ on the set X of k-dimensional linear codes of length n over $\mathbb{F}_q$, represented by generator matrices in systematic form. A monomial map in $\mathrm{Mon}(n, q)$ is given by the product of a permutation matrix (which permutes the codewords coordinates) with a diagonal matrix with all non-zero elements, i.e. $\mathrm{Mon}(n, q) = S_n \rtimes (\mathbb{F}_q^*)^n$. Hence, the action is defined as

$$\star_{\mathsf{LEP}} \colon \mathrm{Mon}(n, q) \times X \to X, (\mathbf{Q}, \mathbf{G}) \mapsto \mathsf{SF}(\mathbf{GQ}).$$

In the following, we consider the parameters proposed in [7] with respect to NIST security levels I, III, and V as reported in Table 5 in Appendix C. The proposed parameterizations for LESS [7] include the unbalanced challenges with seed tree optimizations and, for some sets only, the use of multiple public keys.

LESS also includes a specific optimization presented in [64] with the introduction of the *Information Set* variant of LEP (IS-LEP), which requires two codes to be equivalent only on an information set.

Concrete Parameters. The elements of the monomial group $\mathrm{Mon}(n, q)$ can be represented with $n(\lceil \log_2(n) \rceil + \lceil \log_2(q-1) \rceil)$ bits. Thanks to the information set optimization, it is only required to transmit $\ell_{\mathsf{G}} = k(\lceil \log_2(n) \rceil + \lceil \log_2(q-1) \rceil)$.

Since in LESS, the parameter k is approximately equal to the security parameter λ, the compression of random elements in the multi-signature can be applied when the number of users N is less than $\left\lfloor \frac{\ell_{\mathsf{G}}}{\lambda} \right\rfloor \approx \lceil \log_2(n) \rceil + \lceil \log_2(q-1) \rceil$.

For each NIST security level I, III, and V, the LESS specification proposes three sets of parameters with a progressive trade-off between signature size and public key size, starting with the *balanced* set that does not use multiple keys, with a gradual increase in the *intermediate* and *short* sets. The increase in the number of public keys also corresponds to a lower fixed-weight parameter ω, leading to lower multi-signature compression for "shorter" parameters. However, for the level I balanced set, we achieve a compression of 30% starting for $N > 15$. In Table 2, some parameters for the multi-signature are shown as the number of users N changes, along with the minimum number of users N to achieve a compression of the signature. More extensive data concerning different N is presented in Appendix C.1.

6.2 MEDS

MEDS [30] is a signature scheme based on the hardness of the *Matrix Code Equivalence*. In the following, we consider the parameters proposed in [29] with respect to NIST security levels I, III, and V as reported in Table 6 in Appendix C.

The MEDS group action $\star_{\mathsf{MCE}}$ is given by the action of $\mathrm{GL}_n(q) \times \mathrm{GL}_m(q)$ on the set X of $k \times nm$ matrices in $\mathbb{F}_q$ representing the k-dimensional $(n \times m)$-matrix codes over $\mathbb{F}_q$:

$$\star_{\mathsf{MCE}} \colon (\mathrm{GL}_n(q) \times \mathrm{GL}_m(q)) \times X \to X, ((\mathbf{L}, \mathbf{S}), \mathbf{G}) \mapsto \mathsf{SF}(\mathbf{G}(\mathbf{L}^T \otimes \mathbf{S})).$$

The proposed parameterizations for MEDS [29] include the unbalanced challenges with seed tree optimizations and the use of multiple public keys.

Concrete Parameters. The elements of the group $\mathrm{GL}_n(q) \times \mathrm{GL}_m(q)$ can be represented with $\ell_{\mathsf{G}} = (n^2 + m^2)\lceil \log_2 q \rceil$ bits.

For each NIST security level I, III, and V, the MEDS specification proposes two sets of parameters with a strong trade-off between signature size and scheme efficiency due to the heavy use of the fixed-weight optimization. For the shorter parameterization, the high number of repetitions allows for an extremely low fixed-weight parameter ω, reducing the compression of the multi-signature. This is particularly observable in the short parameterization of security level I, where there is positive aggregation only from $N = 36$.

The best aggregation rates are obtained with the balanced parameter of security level V, with a compression rate greater than 25% already for $N > 2$ and greater than 50% for $N > 19$.

The best results for the number of repetitions are obtained for level I and III balanced parameters with reductions in the number of repetitions exceeding 95% for any number of users.

In Table 3, some parameters for the multi-signature are given for 50 and 100 users. Additional data is presented in Appendix C.2.

Table 3: Minimum number of users N to achieve a signature compression, compression rate, repetition rate and fixed-weight parameter ω' for 50 and 100 users for the MEDS parameters sets. (S) and (B) stand for short and balanced, respectively.

Set	**Min.** N	**Compr. Rate**		**Repet. Rate**		**FW Par.** ω'	
		$N = 50$	$N = 100$	$N = 50$	$N = 100$	$N = 50$	$N = 100$
MEDS-9923 (S)	36	0.030	0.160	0.986	0.988	16	14
MEDS-13220 (B)	7	0.307	0.405	0.920	0.932	15	13
MEDS-41711 (S)	19	0.174	0.250	0.961	0.965	23	21
MEDS-69497 (B)	4	0.407	0.464	0.860	0.874	22	20
MEDS-134180 (S)	3	0.452	0.509	0.846	0.863	29	26
MEDS-167717 (B)	2	0.576	0.622	0.745	0.775	28	25

Table 4: Minimum number of users N to achieve a signature compression, compression rate, repetition rate and Fixed-Weight parameter ω' for 50 and 100 users for the ALTEQ parameters sets.

Set	**Min.** N	**Compr. Rate**		**Repet. Rate**		**FW Par.** ω'	
		$N = 50$	$N = 100$	$N = 50$	$N = 100$	$N = 50$	$N = 100$
Balanced-I	4	0.394	0.443	0.830	0.844	14	13
Short-I	2	0.351	0.429	0.426	0.495	9	8
Balanced-III	6	0.305	0.374	0.893	0.905	21	19
Short-III	2	0.301	0.355	0.634	0.663	14	13

6.3 ALTEQ

ALTEQ [70] is a signature scheme based on the hardness of the *Alternating Trilinear Form Equivalence* problem. In the following, we consider the parameters proposed in [19] with respect to NIST security levels I and III as reported in Table 7 in Appendix C.

The ALTEQ group action $\star_{\mathsf{ATFE}}$ is given by the action of $\mathrm{GL}_n(q)$ on the set of alternating trilinear forms $\mathrm{ATF}(n,q) = \{\phi\colon \mathbb{F}_q^n \times \mathbb{F}_q^n \times \mathbb{F}_q^n \to \mathbb{F}_q\}$:

$$\star_{\mathsf{ATFE}}\colon \mathrm{GL}_n(q) \times \mathrm{ATF}(n,q) \to \mathrm{ATF}(n,q), (\mathbf{A}, \phi) \mapsto \phi \circ \mathbf{A},$$

where $\phi \circ \mathbf{A}$ is defined as the map sending (x, y, z) to $\phi(\mathbf{A}^T x, \mathbf{A}^T y, \mathbf{A}^T z)$.

The proposed parameterizations for ALTEQ [19] include the unbalanced challenges and the multiple public keys optimizations.

Concrete Parameters. The elements of the group $\mathrm{GL}_n(q)$ can be represented with a string of length $\ell_{\mathsf{G}} = n^2 \lceil \log_2 q \rceil$.

For each NIST security level I and III the ALTEQ specification proposes two sets of parameters with a strong trade-off between signature size and public key size due to heavy use of the Multiple Public Keys optimization. The increase in the number of public keys also corresponds to a lower fixed-weight parameter ω, leading to slightly lower multi-signature compression for "shorter" parameters. On the other hand, the large number of public keys also benefits the multi-signature, with little reduction in compression compared to the MEDS case.

Concerning the compression rate, similar results are achieved by all parameterizations, with Level I sets having slightly higher aggregation for fewer users. Considering the balanced parameters of security level I, the compression rate is greater than 30% for $N > 17$. The best results for the repetitions rates are obtained for the balanced parameters of level III with reductions in the number of repetitions exceeding 80% for any number of users.

In Table 4, the minimum number of users N to achieve compression and some parameters for the multi-signature are shown as the number of users N changes. More data on how these parameters change for different N can be found in Appendix C.3.

Acknowledgments. The first and third authors are members of GNSAGA of INdAM. The first and fourth authors are members of CrypTO, the group of Cryptography and Number Theory of the Politecnico di Torino. The second author acknowledges support from Eustema S.p.A. through the PhD scholarship. This work was partially supported by the QUBIP project (https://www.qubip.eu), funded by the European Union under the Horizon Europe framework programme [grant agreement no. 101119746]. This publication was created with the co-financing of the European Union FSE-REACT-EU, PON Research and Innovation 2014-2020 DM1062/2021. The authors would like to thank the anonymous reviewers for the valuable comments, which helped to improve the overall quality of this work.

References

1. Aardal, M.A., Aranha, D.F., Boudgoust, K., Kolby, S., Takahashi, A.: Aggregating falcon signatures with LaBRADOR. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part I. LNCS, vol. 14920, pp. 71–106. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68376-3_3
2. Alamati, N., De Feo, L., Montgomery, H., Patranabis, S.: Cryptographic group actions and applications. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 411–439. Springer, Cham (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_14
3. Albrecht, M.R., Cini, V., Lai, R.W.F., Malavolta, G., Thyagarajan, S.A.K.: Lattice-based SNARKs: Publicly verifiable, preprocessing, and recursively composable - (extended abstract). In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 102–132. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_4
4. Alper, H.K., Burdges, J.: Two-round trip schnorr multi-signatures via delinearized witnesses. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol.

12825, pp. 157–188. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0_7
5. Attema, T., Fehr, S.: Parallel repetition of $(k_1, \ldots, k_\mu)$-special-sound multi-round interactive proofs. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part I. LNCS, vol. 13507, pp. 415–443. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15802-5_15
6. Bagherzandi, A., Cheon, J.H., Jarecki, S.: Multisignatures secure under the discrete logarithm assumption and a generalized forking lemma. In: Ning, P., Syverson, P.F., Jha, S. (eds.) ACM CCS 2008. pp. 449–458. ACM Press (Oct 2008). https://doi.org/10.1145/1455770.1455827
7. Baldi, M., Barenghi, A., Beckwith, L., Biasse, J., Esser, A., Gaj, K., Mohajerani, K., Pelosi, G., Persichetti, E., Saarinen, M.O., Santini, P., Wallace, R.: LESS — Linear Equivalence Signature Scheme. Tech. rep., National Institute of Standards and Technology (2023), available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures
8. Barenghi, A., Biasse, J.F., Persichetti, E., Santini, P.: LESS-FM: Fine-tuning signatures from the code equivalence problem. In: Cheon, J.H., Tillich, J.P. (eds.) Post-Quantum Cryptography - 12th International Workshop, PQCrypto 2021. pp. 23–43. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-81293-5_2
9. Battagliola, M., Borin, G., Meneghetti, A., Persichetti, E.: Cutting the GRASS: Threshold GRoup action signature schemes. In: Oswald, E. (ed.) CT-RSA 2024. LNCS, vol. 14643, pp. 460–489. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58868-6_18
10. Battagliola, M., Longo, R., Pintore, F., Signorini, E., Tognolini, G.: Security of fixed-weight repetitions of special-sound multi-round proofs. Cryptology ePrint Archive, Report 2024/884 (2024), https://eprint.iacr.org/2024/884
11. Bellare, M., Davis, H., Günther, F.: Separate your domains: NIST PQC KEMs, oracle cloning and read-only indifferentiability. In: Canteaut, A., Ishai, Y. (eds.) EUROCRYPT 2020, Part II. LNCS, vol. 12106, pp. 3–32. Springer, Cham (May 2020). https://doi.org/10.1007/978-3-030-45724-2_1
12. Bellare, M., Namprempre, C., Neven, G.: Unrestricted aggregate signatures. In: Arge, L., Cachin, C., Jurdzinski, T., Tarlecki, A. (eds.) ICALP 2007. LNCS, vol. 4596, pp. 411–422. Springer, Berlin, Heidelberg (Jul 2007). https://doi.org/10.1007/978-3-540-73420-8_37
13. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Juels, A., Wright, R.N., De Capitani di Vimercati, S. (eds.) ACM CCS 2006. pp. 390–399. ACM Press (Oct / Nov 2006). https://doi.org/10.1145/1180405.1180453
14. Beullens, W., Dobson, S., Katsumata, S., Lai, Y.F., Pintore, F.: Group signatures and more from isogenies and lattices: Generic, simple, and efficient. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part II. LNCS, vol. 13276, pp. 95–126. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07085-3_4
15. Beullens, W., Katsumata, S., Pintore, F.: Calamari and Falafl: Logarithmic (linkable) ring signatures from isogenies and lattices. In: Moriai, S., Wang, H. (eds.) ASIACRYPT 2020, Part II. LNCS, vol. 12492, pp. 464–492. Springer, Cham (Dec 2020). https://doi.org/10.1007/978-3-030-64834-3_16
16. Beullens, W., Kleinjung, T., Vercauteren, F.: CSI-FiSh: Efficient isogeny based signatures through class group computations. In: Galbraith, S.D., Moriai, S. (eds.) ASIACRYPT 2019, Part I. LNCS, vol. 11921, pp. 227–247. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-34578-5_9

17. Biasse, J.F., Micheli, G., Persichetti, E., Santini, P.: LESS is more: Code-based signatures without syndromes. In: Nitaj, A., Youssef, A.M. (eds.) AFRICACRYPT 20. LNCS, vol. 12174, pp. 45–65. Springer, Cham (Jul 2020). https://doi.org/10.1007/978-3-030-51938-4_3
18. Bläser, M., Chen, Z., Duong, D.H., Joux, A., Nguyen, T.N., Plantard, T., Qiao, Y., Susilo, W., Tang, G.: On digital signatures based on group actions: QROM security and ring signatures. In: Saarinen, M.J., Smith-Tone, D. (eds.) Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part I. pp. 227–261. Springer, Cham (Jun 2024). https://doi.org/10.1007/978-3-031-62743-9_8
19. Bläser, M., Duong, D.H., Narayanan, A.K., Plantard, T., Qiao, Y., Sipasseuth, A., Tang, G.: ALTEQ. Tech. rep., National Institute of Standards and Technology (2023), available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures
20. Boneh, D., Gentry, C., Lynn, B., Shacham, H.: Aggregate and verifiably encrypted signatures from bilinear maps. In: Biham, E. (ed.) EUROCRYPT 2003. LNCS, vol. 2656, pp. 416–432. Springer, Berlin, Heidelberg (May 2003). https://doi.org/10.1007/3-540-39200-9_26
21. Boschini, C., Takahashi, A., Tibouchi, M.: MuSig-L: Lattice-based multi-signature with single-round online phase. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 276–305. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_10
22. Boudgoust, K., Takahashi, A.: Sequential half-aggregation of lattice-based signatures. In: Tsudik, G., Conti, M., Liang, K., Smaragdakis, G. (eds.) ESORICS 2023, Part I. LNCS, vol. 14344, pp. 270–289. Springer, Cham (Sep 2023). https://doi.org/10.1007/978-3-031-50594-2_14
23. Brogle, K., Goldberg, S., Reyzin, L.: Sequential aggregate signatures with lazy verification from trapdoor permutations - (extended abstract). In: Wang, X., Sako, K. (eds.) ASIACRYPT 2012. LNCS, vol. 7658, pp. 644–662. Springer, Berlin, Heidelberg (Dec 2012). https://doi.org/10.1007/978-3-642-34961-4_39
24. Castryck, W., Lange, T., Martindale, C., Panny, L., Renes, J.: CSIDH: An efficient post-quantum commutative group action. In: Peyrin, T., Galbraith, S. (eds.) ASIACRYPT 2018, Part III. LNCS, vol. 11274, pp. 395–427. Springer, Cham (Dec 2018). https://doi.org/10.1007/978-3-030-03332-3_15
25. Chailloux, A., Etinski, S.: On the (in)security of optimized stern-like signature schemes. DCC **92**(3), 803–832 (2024). https://doi.org/10.1007/s10623-023-01329-y
26. Chen, M., Leroux, A., Panny, L.: SCALLOP-HD: Group action from 2-dimensional isogenies. In: Tang, Q., Teague, V. (eds.) PKC 2024, Part II. LNCS, vol. 14603, pp. 190–216. Springer, Cham (Apr 2024). https://doi.org/10.1007/978-3-031-57725-3_7
27. Chen, Y.: DualMS: Efficient lattice-based two-round multi-signature with trapdoor-free simulation. In: Handschuh, H., Lysyanskaya, A. (eds.) CRYPTO 2023, Part V. LNCS, vol. 14085, pp. 716–747. Springer, Cham (Aug 2023). https://doi.org/10.1007/978-3-031-38554-4_23
28. Chen, Z., Duong, D.H., Nguyen, N.T., Qiao, Y., Susilo, W., Tang, G.: On digital signatures based on isomorphism problems: QROM security and ring signatures. Cryptology ePrint Archive, Report 2022/1184 (2022), https://eprint.iacr.org/2022/1184
29. Chou, T., Niederhagen, R., Persichetti, E., Ran, L., Randrianarisoa, T.H., Reijnders, K., Samardjiska, S., Trimoska, M.: MEDS — Matrix Equivalence Digital Signature. Tech. rep., National Institute of Standards and Technology (2023), available at https://csrc.nist.gov/Projects/pqc-dig-sig/round-1-additional-signatures

30. Chou, T., Niederhagen, R., Persichetti, E., Randrianarisoa, T.H., Reijnders, K., Samardjiska, S., Trimoska, M.: Take your MEDS: Digital signatures from matrix code equivalence. In: El Mrabet, N., De Feo, L., Duquesne, S. (eds.) AFRICACRYPT 23. LNCS, vol. 14064, pp. 28–52. Springer, Cham (Jul 2023). https://doi.org/10.1007/978-3-031-37679-5_2
31. Chou, T., Persichetti, E., Santini, P.: On linear equivalence, canonical forms, and digital signatures. Cryptology ePrint Archive, Report 2023/1533 (2023), https://eprint.iacr.org/2023/1533
32. Couveignes, J.M.: Hard homogeneous spaces. Cryptology ePrint Archive, Report 2006/291 (2006), https://eprint.iacr.org/2006/291
33. Cover, T.M., Thomas, J.A.: Elements of information theory. Hoboken, NJ: Wiley-Interscience (2006)
34. Damgård, I., Orlandi, C., Takahashi, A., Tibouchi, M.: Two-round n-out-of-n and multi-signatures and trapdoor commitment from lattices. In: Garay, J. (ed.) PKC 2021, Part I. LNCS, vol. 12710, pp. 99–130. Springer, Cham (May 2021). https://doi.org/10.1007/978-3-030-75245-3_5
35. De Feo, L., Fouotsa, T.B., Kutas, P., Leroux, A., Merz, S.P., Panny, L., Wesolowski, B.: SCALLOP: Scaling the CSI-FiSh. In: Boldyreva, A., Kolesnikov, V. (eds.) PKC 2023, Part I. LNCS, vol. 13940, pp. 345–375. Springer, Cham (May 2023). https://doi.org/10.1007/978-3-031-31368-4_13
36. De Feo, L., Galbraith, S.D.: SeaSign: Compact isogeny signatures from class group actions. In: Ishai, Y., Rijmen, V. (eds.) EUROCRYPT 2019, Part III. LNCS, vol. 11478, pp. 759–789. Springer, Cham (May 2019). https://doi.org/10.1007/978-3-030-17659-4_26
37. Devadas, L., Goyal, R., Kalai, Y., Vaikuntanathan, V.: Rate-1 non-interactive arguments for batch-NP and applications. In: 63rd FOCS. pp. 1057–1068. IEEE Computer Society Press (Oct / Nov 2022). https://doi.org/10.1109/FOCS54457.2022.00103
38. Don, J., Fehr, S., Majenz, C., Schaffner, C.: Security of the Fiat-Shamir transformation in the quantum random-oracle model. In: Boldyreva, A., Micciancio, D. (eds.) CRYPTO 2019, Part II. LNCS, vol. 11693, pp. 356–383. Springer, Cham (Aug 2019). https://doi.org/10.1007/978-3-030-26951-7_13
39. Drijvers, M., Edalatnejad, K., Ford, B., Kiltz, E., Loss, J., Neven, G., Stepanovs, I.: On the security of two-round multi-signatures. In: 2019 IEEE Symposium on Security and Privacy. pp. 1084–1101. IEEE Computer Society Press (May 2019). https://doi.org/10.1109/SP.2019.00050
40. El Bansarkhani, R., Buchmann, J.: Towards lattice based aggregate signatures. In: Pointcheval, D., Vergnaud, D. (eds.) AFRICACRYPT 14. LNCS, vol. 8469, pp. 336–355. Springer, Cham (May 2014). https://doi.org/10.1007/978-3-319-06734-6_21
41. El Bansarkhani, R., Sturm, J.: An efficient lattice-based multisignature scheme with applications to bitcoins. In: Foresti, S., Persiano, G. (eds.) CANS 16. LNCS, vol. 10052, pp. 140–155. Springer, Cham (Nov 2016). https://doi.org/10.1007/978-3-319-48965-0_9
42. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) CRYPTO'86. LNCS, vol. 263, pp. 186–194. Springer, Berlin, Heidelberg (Aug 1987). https://doi.org/10.1007/3-540-47721-7_12
43. Fukumitsu, M., Hasegawa, S.: A lattice-based provably secure multisignature scheme in quantum random oracle model. In: Nguyen, K., Wu, W., Lam, K.Y., Wang, H. (eds.) ProvSec 2020. LNCS, vol. 12505, pp. 45–64. Springer, Cham (Nov / Dec 2020). https://doi.org/10.1007/978-3-030-62576-4_3

44. Gentry, C., O'Neill, A., Reyzin, L.: A unified framework for trapdoor-permutation-based sequential aggregate signatures. In: Abdalla, M., Dahab, R. (eds.) PKC 2018, Part II. LNCS, vol. 10770, pp. 34–57. Springer, Cham (Mar 2018). https://doi.org/10.1007/978-3-319-76581-5_2
45. Gentry, C., Peikert, C., Vaikuntanathan, V.: Trapdoors for hard lattices and new cryptographic constructions. In: Ladner, R.E., Dwork, C. (eds.) 40th ACM STOC. pp. 197–206. ACM Press (May 2008). https://doi.org/10.1145/1374376.1374407
46. Goldreich, O., Micali, S., Wigderson, A.: Proofs that yield nothing but their validity and a methodology of cryptographic protocol design (extended abstract). In: 27th FOCS. pp. 174–187. IEEE Computer Society Press (Oct 1986). https://doi.org/10.1109/SFCS.1986.47
47. Hohenberger, S., Koppula, V., Waters, B.: Universal signature aggregators. In: Oswald, E., Fischlin, M. (eds.) EUROCRYPT 2015, Part II. LNCS, vol. 9057, pp. 3–34. Springer, Berlin, Heidelberg (Apr 2015). https://doi.org/10.1007/978-3-662-46803-6_1
48. Itakura, K., Nakamura, K.: A public-key cryptosystem suitable for digital multisignature. NEC research and development **71**, 1–8 (1983)
49. Ji, Z., Qiao, Y., Song, F., Yun, A.: General linear group action on tensors: A candidate for post-quantum cryptography. In: Hofheinz, D., Rosen, A. (eds.) TCC 2019, Part I. LNCS, vol. 11891, pp. 251–281. Springer, Cham (Dec 2019). https://doi.org/10.1007/978-3-030-36030-6_11
50. Katz, J., Lindell, Y.: Introduction to modern cryptography: principles and protocols. Chapman and hall/CRC (2007)
51. Leroux, A., Roméas, M.: Updatable encryption from group actions. In: Saarinen, M.J., Smith-Tone, D. (eds.) Post-Quantum Cryptography - 15th International Workshop, PQCrypto 2024, Part II. pp. 20–53. Springer, Cham (Jun 2024). https://doi.org/10.1007/978-3-031-62746-0_2
52. Lysyanskaya, A., Micali, S., Reyzin, L., Shacham, H.: Sequential aggregate signatures from trapdoor permutations. In: Cachin, C., Camenisch, J. (eds.) EUROCRYPT 2004. LNCS, vol. 3027, pp. 74–90. Springer, Berlin, Heidelberg (May 2004). https://doi.org/10.1007/978-3-540-24676-3_5
53. Ma, C., Weng, J., Li, Y., Deng, R.H.: Efficient discrete logarithm based multi-signature scheme in the plain public key model. DCC **54**(2), 121–133 (2010). https://doi.org/10.1007/s10623-009-9313-z
54. Maurer, U.M., Renner, R., Holenstein, C.: Indifferentiability, impossibility results on reductions, and applications to the random oracle methodology. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 21–39. Springer, Berlin, Heidelberg (Feb 2004). https://doi.org/10.1007/978-3-540-24638-1_2
55. Maxwell, G., Poelstra, A., Seurin, Y., Wuille, P.: Simple schnorr multi-signatures with applications to bitcoin. DCC **87**(9), 2139–2164 (2019). https://doi.org/10.1007/s10623-019-00608-x
56. Meneghetti, A., Signorini, E.: History-free sequential aggregation of hash-and-sign signatures. In: Oswald, E. (ed.) CT-RSA 2024. LNCS, vol. 14643, pp. 187–223. Springer, Cham (May 2024). https://doi.org/10.1007/978-3-031-58868-6_8
57. National Institute of Standards and Technology (NIST): SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions (2015). https://doi.org/https://doi.org/10.6028/NIST.FIPS.202
58. Neven, G.: Efficient sequential aggregate signed data. In: Smart, N.P. (ed.) EUROCRYPT 2008. LNCS, vol. 4965, pp. 52–69. Springer, Berlin, Heidelberg (Apr 2008). https://doi.org/10.1007/978-3-540-78967-3_4

59. Nick, J., Ruffing, T., Seurin, Y.: MuSig2: Simple two-round Schnorr multi-signatures. In: Malkin, T., Peikert, C. (eds.) CRYPTO 2021, Part I. LNCS, vol. 12825, pp. 189–221. Springer, Cham, Virtual Event (Aug 2021). https://doi.org/10.1007/978-3-030-84242-0_8
60. Nick, J., Ruffing, T., Seurin, Y., Wuille, P.: MuSig-DN: Schnorr multi-signatures with verifiably deterministic nonces. In: Ligatti, J., Ou, X., Katz, J., Vigna, G. (eds.) ACM CCS 2020. pp. 1717–1731. ACM Press (Nov 2020). https://doi.org/10.1145/3372297.3417236
61. Nicolosi, A., Krohn, M.N., Dodis, Y., Mazières, D.: Proactive two-party signatures for user authentication. In: NDSS 2003. The Internet Society (Feb 2003)
62. Ohta, K., Okamoto, T.: Multi-signature schemes secure against active insider attacks. IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences **82**(1), 21–31 (1999)
63. Okamoto, T.: A digital multisignature scheme using bijective public-key cryptosystems. ACM Transactions on Computer Systems (TOCS) **6**(4), 432–441 (1988)
64. Persichetti, E., Santini, P.: A new formulation of the linear equivalence problem and shorter LESS signatures. In: Guo, J., Steinfeld, R. (eds.) ASIACRYPT 2023, Part VII. LNCS, vol. 14444, pp. 351–378. Springer, Singapore (Dec 2023). https://doi.org/10.1007/978-981-99-8739-9_12
65. Pham, M.T.T., Duong, D.H., Li, Y., Susilo, W.: Threshold ring signature scheme from cryptographic group action. In: Zhang, M., Au, M.H., Zhang, Y. (eds.) ProvSec 2023. LNCS, vol. 14217, pp. 207–227. Springer, Cham (Oct 2023). https://doi.org/10.1007/978-3-031-45513-1_12
66. Ransom, R.: Constant-time verification for cut-and-choose-based signatures. Cryptology ePrint Archive, Report 2020/1184 (2020), https://eprint.iacr.org/2020/1184
67. Reijnders, K., Samardjiska, S., Trimoska, M.: Hardness estimates of the code equivalence problem in the rank metric. DCC **92**(3), 833–862 (2024). https://doi.org/10.1007/s10623-023-01338-x
68. Stolbunov, A.: Cryptographic schemes based on isogenies. Phd thesis, Norwegian University of Science and Technology (2012)
69. Syta, E., Tamas, I., Visher, D., Wolinsky, D.I., Jovanovic, P., Gasser, L., Gailly, N., Khoffi, I., Ford, B.: Keeping authorities "honest or bust" with decentralized witness cosigning. In: 2016 IEEE Symposium on Security and Privacy. pp. 526–545. IEEE Computer Society Press (May 2016). https://doi.org/10.1109/SP.2016.38
70. Tang, G., Duong, D.H., Joux, A., Plantard, T., Qiao, Y., Susilo, W.: Practical post-quantum signature schemes from isomorphism problems of trilinear forms. In: Dunkelman, O., Dziembowski, S. (eds.) EUROCRYPT 2022, Part III. LNCS, vol. 13277, pp. 582–612. Springer, Cham (May / Jun 2022). https://doi.org/10.1007/978-3-031-07082-2_21
71. Waters, B., Wu, D.J.: Batch arguments for NP and more from standard bilinear group assumptions. In: Dodis, Y., Shrimpton, T. (eds.) CRYPTO 2022, Part II. LNCS, vol. 13508, pp. 433–463. Springer, Cham (Aug 2022). https://doi.org/10.1007/978-3-031-15979-4_15
72. Zhang, X., Steinfeld, R., Esgin, M.F., Liu, J.K., Liu, D., Ruj, S.: Loquat: A SNARK-friendly post-quantum signature based on the legendre PRF with applications in ring and aggregate signatures. In: Reyzin, L., Stebila, D. (eds.) CRYPTO 2024, Part I. LNCS, vol. 14920, pp. 3–38. Springer, Cham (Aug 2024). https://doi.org/10.1007/978-3-031-68376-3_1

A Proof for EUF-CMA of centralized signature (Theorem 1)

In this section, we prove the EUF-CMA security of a variant of the centralized signature based on group actions. The signature scheme is then used in the security proof of Section 4 to prove the security of the multi-signature scheme.

In the following, we adopt the notation of [5] to define an interactive proof and its main properties. In particular, we recall the following standard definitions.

Definition 7 (Interactive Proof). *An interactive proof $(\mathcal{P}, \mathcal{V})$ for a binary relation R is an interactive protocol between two probabilistic machines, a prover $\mathcal{P}$ and a polynomial time verifier $\mathcal{V}$. Both $\mathcal{P}$ and $\mathcal{V}$ take as public input a statement x and, additionally, $\mathcal{P}$ takes as private input a witness $w \in R(x)$. We denote the protocol instance as $(\mathcal{P}(w), \mathcal{V})(x)$. As the output of the protocol, $\mathcal{V}$ either returns* `accept` *or* `reject`*. Accordingly, we say the corresponding transcript (i.e., the set of all messages exchanged in the protocol execution) is accepting or rejecting.*

Definition 8 (Knowledge Soundness). *An interactive proof $(\mathcal{P}, \mathcal{V})$ for relation R is knowledge sound with knowledge error $\kappa\colon \{0,1\}^* \to [0,1]$ if there exists a positive polynomial q and an algorithm $\mathcal{E}$, called a knowledge extractor, with the following properties: The extractor $\mathcal{E}$, given input x and rewindable oracle access to a (potentially dishonest) prover $\mathcal{P}^*$, runs in an expected number of steps that is polynomial in $|x|$ and outputs a witness $w \in R(x)$ with probability*

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

where $\varepsilon(x, \mathcal{P}^*) := \Pr[(\mathcal{P}^*, \mathcal{V})(x) = \texttt{accept}]$.

It can be shown [5] that to satisfy the previous definition, it is sufficient to restrict to *deterministic* provers $\mathcal{P}^*$.

Definition 9 (Public-Coin). *An interactive proof $(\mathcal{P}, \mathcal{V})$ is public-coin if all of $\mathcal{V}$'s random choices are made public.*

We refer to a 3-round public-coin interactive proof as a Σ-protocol.

A common strategy to prove the knowledge soundness of a 3-round protocol is showing that it enjoys (general) *special soundness*. Informally, given enough accepting transcripts with a fixed commitment, it is possible to extract a witness.

Definition 10 (k-out-of-N Special-Soundness). *Let $k, N \in \mathbb{N}$. A 3-round public-coin protocol $(\mathcal{P}, \mathcal{V})$ for relation R, with challenge set of cardinality $N \geq k$, is k-out-of-N special-sound if there exists a polynomial time algorithm that, on input a statement x and k accepting transcripts $(a, c_1, z_1), \ldots, (a, c_k, z_k)$ with common first message a and pairwise distinct challenges $c_1, \ldots, c_k$, outputs a witness $w \in R(x)$. We also say $(\mathcal{P}, \mathcal{V})$ is k-special-sound and, if $k = 2$, it is simply said to be special-sound.*

It is well known that k-out-of-N special soundness implies knowledge soundness with knowledge error $(k-1)/N$.

Algorithm 5: Group Action Σ-Protocol with n ephemeral keys

`Setup`: Group G acting on X via $\star$. Choose $x_0 \in X, n \in \mathbb{N}$. Set $\hat{x}_0 = x_0, \hat{g}_0 = e$.
`Private key`: $g_1 \in G$.
`Public key`: $x_1 \leftarrow g_1 \star x_0$. Set $\hat{x}_1 = x_1$.

Prover(g_1, x_1)		**Verifier**(x_1)
for $j \leftarrow 2, \ldots, n$ **do**		
$\quad \hat{g}_j \leftarrow\!\!\$\ G$		
$\quad \hat{x}_j \leftarrow \hat{g}_j \star x_0$		
$\tilde{g} \leftarrow\!\!\$\ G$		
$\tilde{x} \leftarrow \tilde{g} \star x_0$	$\xrightarrow{\hat{x}_2, \ldots, \hat{x}_n, \tilde{x}}$	
	$\xleftarrow{\mathsf{ch}}$	$\mathsf{ch} \leftarrow\!\!\$\ [n]$
$z \leftarrow \tilde{g}\hat{g}_{\mathsf{ch}}^{-1}$	$\xrightarrow{z}$	
		Accept if $z \star \hat{x}_{\mathsf{ch}} = \tilde{x}$

A.1 Base Sigma Protocol Variant

We start by modifying the base Sigma protocol from a cryptographic group action $\star\colon G \times X \to X$. In the base protocol, to prove the knowledge of a secret $g_1 \in G$ such that $x_1 = g_1 \star x_0$, the prover proceeds as follows: first, it samples a random $\tilde{g} \in G$ and computes the commitment as $\tilde{x} \leftarrow \tilde{g} \star x_0$. Then, they exhibit a path from either x_0 or x_1 to $\tilde{x}$ based on the challenge of the verifier. This protocol is correct, 2-special sound and perfect honest-verifier zero-knowledge (HVZK).

Given $n > 1$, we denote with $\Pi[n]$ the variant Σ-protocol where the prover additionally samples $n-1$ ephemeral keys $\hat{x}_2, \ldots, \hat{x}_n \in X$ during the commitment phase. The verifier can then request a path to $\tilde{x}$ from x_0, x_1 or one of the ephemeral keys. The protocol is detailed in Algorithm 5. This procedure artificially increases the challenge space to the set $\{0, \ldots, n\}$ and increases the knowledge error to $(n-1)/n$. Clearly, this is a step back from the basic protocol, but it effectively represents a protocol where all but one key is potentially under the control of an adversary.

Proposition 1. *$\Pi[n]$ is complete, $(n+1)$-special-sound and honest-verifier zero-knowledge.*

Proof. We prove each property separately.

Completeness In an honest execution, the verifier receives $z = \tilde{g}\hat{g}_{\mathsf{ch}}^{-1}$. Completeness easily follows since $z \star \hat{x}_{\mathsf{ch}} = \tilde{g}\hat{g}_{\mathsf{ch}}^{-1}\hat{g}_{\mathsf{ch}} \star x_0 = \tilde{x}$.

$(n+1)$-Special Soundness Suppose the extractor has access to $n+1$ accepting transcript $(\mathsf{com}, \mathsf{ch}_1, z_1), \ldots, (\mathsf{com}, \mathsf{ch}_{n+1}, z_{n+1})$ with $\mathsf{com} = (\hat{x}_2, \ldots, \hat{x}_n, \tilde{x})$ and pairwise distinct challenges $\mathsf{ch}_1, \ldots, \mathsf{ch}_{n+1}$. It follows that there exists $i, j \in [n+1]$ such that $\mathsf{ch}_i = 1$ and $\mathsf{ch}_j = 0$. Then it can run the special-soundness extractor from the identification scheme of Section 2.3 on $(\mathsf{com}, \mathsf{ch}_i, z_i), (\mathsf{com}, \mathsf{ch}_j, z_j)$, extracting the witness g_1 as $z_i^{-1} z_j$.

Honest-Verifier Zero-Knowledge A polynomial time simulator can be obtained as follows. On input a public key $x_1 \in X$, the simulator randomly samples $\mathsf{ch} \leftarrow\!\!\$ \{0, \ldots, n\}, z \leftarrow\!\!\$ G$, and $\hat{g}_2, \ldots, \hat{g}_n \leftarrow\!\!\$ G$. Then it computes $\hat{x}_j \leftarrow \hat{g}_j \star x_0$ and $\tilde{x} \leftarrow z \star \hat{x}_{\mathsf{ch}}$. Finally, it returns the transcript $(\mathsf{com} = (\hat{x}_2, \ldots, \hat{x}_n, \tilde{x}), \mathsf{ch}, z)$. Notice that in the real distribution, $\hat{x}_2, \ldots, \hat{x}_n$ and $\tilde{x}$ are uniformly distributed in the orbit of x_0, ch is uniformly distributed in $\{0, \ldots, n\}$, and $z \in G$ is uniquely determined by $z \star \hat{x}_{\mathsf{ch}} = \tilde{x}$. This is the same in the simulated distribution.

□

Since $\Pi[n]$ is $(n+1)$-out-of-$(n+1)$ special-sound Σ-protocol, we can take its parallel repetition and still obtain a proof of knowledge. In particular, by applying [5, Theorem 2], we obtain that the t-fold parallel repetition of $\Pi[n]$ is knowledge-sound with knowledge error $n^t/(n+1)^t$. Moreover, the same approach can be applied to show that the t-fold parallel repetition of $\Pi[n]$, with a single sample of the ephemeral keys $\hat{x}_2, \ldots, \hat{x}_n$, is still knowledge-sound with knowledge error $n^t/(n+1)^t$.

A.2 Variant with Variable Ephemeral Keys

Next, we show that the full variant Π of Section 3.1 is a proof of knowledge. On a high level, we show that any dishonest prover $\mathcal{P}^*$ attacking Π can be used to build a prover $\mathcal{P}_n^*$ against $\Pi[n]^{t(n)}$ with the same success probability of $\mathcal{P}^*$. Since $\Pi[n]^{t(n)}$ is a proof of knowledge, it is possible to extract the witness from $\mathcal{P}_n^*$.

In Π, the challenge space Ch is taken together with a family of surjective maps $f_n \colon \mathsf{Ch} \to [0,n]^{t(n)}$ such that $f_n^{-1}(U([0,n]^{t(n)})) \sim U(\mathsf{Ch})$. Let $\mathcal{P}^*$ be a deterministic prover attacking Π on input x_1. More precisely, on input $c \in \mathsf{Ch}$, $\mathcal{P}^*$ outputs a fixed first message a and its response z. We define $V \colon \mathsf{Ch} \times \{0,1\}^* \to \{\mathtt{accept}, \mathtt{reject}\}$ the function that runs the verification check that the verifier performs on the transcript (a, c, z). $\mathcal{P}^*$ is successful on input c if $V(c, \mathcal{P}^*(c)) = \mathtt{accept}$. Since $\mathcal{P}^*$ is deterministic, the number of ephemeral keys n is fixed and known from the first message a. Then, such a prover naturally induces a (probabilistic) prover $\mathcal{P}_n^*$ attacking $\Pi[n]^{t(n)}$. More precisely, on input $c' \in [0,n]^{t(n)}$, $\mathcal{P}_n^*$ randomly samples $c \leftarrow\!\!\$ f_n^{-1}(c')$, runs $z \leftarrow \mathcal{P}^*(c)$, and outputs z. Since $\Pi[n]^{t(n)}$ is knowledge-sound, we can use the protocol extractor on $\mathcal{P}_n^*$ and estimate its success probability with respect to the success probability of $\mathcal{P}^*$.

We know that the knowledge error for $\Pi[n]^{t(n)}$ is $\kappa_n^{t(n)}$ with $\kappa_n = n/(n+1)$. Then, given access to $\mathcal{P}_n^*$, the extractor for $\Pi[n]^{t(n)}$ succeeds with probability at least $(\varepsilon(x_1, \mathcal{P}_n^*) - \kappa^{t(n)})/\mathsf{poly}(|x_1|)$. To conclude, it is enough to show that the success probability of $\mathcal{P}_n^*$ is the same as for $\mathcal{P}^*$:

$$\begin{aligned}\varepsilon(x_1, \mathcal{P}_n^*) &= \Pr_{c' \leftarrow\$ [0,n]^{t(n)}}[V(c, \mathcal{P}_n^*(c') = \texttt{accept}] \\ &= \Pr_{c' \leftarrow\$ [0,n]^{t(n)}}\left[V(c, \mathcal{P}^*(c)) = \texttt{accept} \mid c \leftarrow\$\ f_n^{-1}(c')\right] \\ &= \Pr_{c \leftarrow\$ \mathsf{Ch}}[V(c, \mathcal{P}^*(c)) = \texttt{accept}] = \varepsilon(x_1, \mathcal{P}^*).\end{aligned}$$

Therefore, the probability that the extractor on $\mathcal{P}^*$ is successful is at least

$$\frac{\varepsilon(x_1, \mathcal{P}^*) - \kappa_n^{t(n)}}{\mathsf{poly}(|x_1|)}.$$

The previous holds for any prover $\mathcal{P}^*$ with n ephemeral keys. Moreover, by taking $n = N$, it holds for any dishonest prover against Π. Therefore, Π is a knowledge-sound Sigma protocol.

A.3 Application of the Fiat-Shamir Transform

We apply the Fiat-Shamir transform [42] to obtain a signature scheme from the Sigma protocol Π of Section 3.1. To prove the EUF-CMA security of the signature scheme, we apply [38, Theorem 22]. Since we already showed that Π is a proof of knowledge, it is enough to prove that Π has perfect unique responses and is perfect HVZK.

Perfect Unique Responses. A widely adopted assumption used to define digital signatures on group actions $\star : G \times X \to X$ states that the group of stabilizers of any element of the set X is the trivial group. Moreover, in [28, Lemma 1] it is proven that the sigma protocol in Algorithm 1 supports perfect unique responses if and only if the group of stabilizers is trivial. It is easy to see that since the sigma protocol Algorithm 1 supports perfect unique responses, also the sigma protocol variants that we present support perfect unique responses.

Perfect HVZK. Immediatly follows from the perfect HVZK of $\Pi[n]$ where the simulator also randomly sample $n \leftarrow\$ [1, N]$.

B On the family of functions $\{f_n\}$

In Section 3.1, we require the existence of a family of functions $\{f_n \colon \mathsf{Ch} \to [0,n]^{t(n)} \mid n \in [1,N]\}$ with certain properties. Later, we argue that $\{f_n\}$ can be used to build multiple random oracles $\mathsf{H}^{(n)}$ on $[0,n]^{t(n)}$, for $n \in [1,N]$, from a single random oracle H on $\mathsf{Ch} \subseteq [0,N]^{t(N)}$. In this section, we show how such family of functions can be constructed, and how each $\mathsf{H}^{(n)}$ is indifferentiable from a random oracle.

B.1 A concrete construction

Here we give an example instantiation of the family of function $\{f_n\}$ from Section 3.1. Recall that N and n are positive integers, $\mathsf{Ch} \subseteq [0,N]^{t(N)}$ and $f_n \colon \mathsf{Ch} \to [0,n]^{t(n)}$ is a family of maps such that $f_n^{-1}\left(U([0,n]^{t(n)})\right) \sim U(\mathsf{Ch})$ for any $n \in [1,N]$. To achieve perfect indistinguishability from the uniform distribution, we introduce an additional bound $\bar{N}$ on n.

Set $N = \mathrm{lcm}(\{n+1 : n \leq \bar{N}\}) - 1$, then define the map f_n as

$$\begin{aligned} f_n : [0,N]^{t(N)} &\to [0,n]^{t(n)}\,, \\ \left(\mathsf{ch}_1, \ldots, \mathsf{ch}_{t(N)}\right) &\mapsto \left(\mathsf{ch}_1 \mod (n+1), \ldots, \mathsf{ch}_{t(n)} \mod (n+1)\right), \end{aligned}$$

i.e., takes the first $t(n)$ values modulo $n+1$ and discards the other ones.

Note that for all $n \leq \bar{N}$, the function f_n is surjective, and for each $\mathsf{ch}' \in [0,n]^{t(n)}$ the preimage $f_n^{-1}(\mathsf{ch}')$ is a subset of $[0,N]^{t(N)}$ of cardinality

$$\left(\frac{N+1}{n+1}\right)^{t(n)} \cdot N^{t(N)-t(n)}$$

since the first $t(n)$ components take values satisfying $\mathsf{ch}'_i = \mathsf{ch}_i \mod (n+1)$, and all the other components can take any possible value in $[0,N]$. Therefore, all the subsets have the same cardinality.

This means that sampling uniformly at random an element in $[0,N]^{t(N)}$ is equivalent to: (1) sampling one of the subsets S (sampling an element ch' in $[0,n]^{t(n)}$ and computing $f_n^{-1}(\mathsf{ch}') = S$), then (2) sampling uniformly at random an element from S.

B.2 Indifferentiability from a Random Oracle

In this section, we show that given a random oracle H on $[0,N]^{t(N)}$ and a family of functions $\{f_n\}$ as described in Section 3.1, each f_n defines a function $\mathsf{H}^{(n)}$ which is *indifferentiable* [54] from a random oracle on $[0,n]^{t(n)}$. Our proof relies on the notion of oracle cloning and read-only indifferentiability introduced in [11]. We start by giving relevant background and definitions, and then we show how to apply [11, Theorem 4.2] to our construction.

Indifferentiability Background. The indifferentiability framework, introduced by Maurer, Renner, and Holenstein in 2004 [54], extends the traditional notion of indistinguishability to scenarios where an adversary has access to both the main cryptographic construction and its internal building blocks. At its core, the framework allows to prove when an idealized functionality (like a random oracle) can be safely replace by a real construction, while maintaining security properties. More formally, we will use the theoretical framework of *function spaces* of [11], as detailed below.

Algorithm 6: Inverse of Query and Answer Translators

$\mathsf{QTI}_{\boldsymbol{f},\boldsymbol{p}}(U)$:
1: $\boldsymbol{W} \leftarrow ()$
2: **for** $i \leftarrow 1, \ldots, \bar{N}$ **do**
3: **if** $\boldsymbol{p}[i]$ prefixes U **then**
4: $\boldsymbol{p}[i] \parallel X \leftarrow U$
5: $\boldsymbol{W} \leftarrow ((i, X))$
6: **return** $\boldsymbol{W}$

$\mathsf{ATI}(U, \boldsymbol{Y})$:
1: **if** $\boldsymbol{Y} = ()$ **then**
2: $V \leftarrow 0^{\mathsf{OL}(N)}$
3: **for** $i \leftarrow 1, \ldots, \bar{N}$ **do**
4: **if** $\boldsymbol{p}[i]$ prefixes U **then**
5: $V \leftarrow \bar{f}_i^{-1}(\boldsymbol{Y}[1])$ // Deterministically sample from $\bar{f}_i^{-1}(\cdot)$
6: **return** V

Functions and Functors. A function space FS is a set of functions with the same domain $\mathsf{Dom}(\mathsf{FS})$ and the same codomain $\mathsf{Rng}(\mathsf{FS})$. In the following, we mainly consider function spaces FS with some domain $\mathcal{D}$ and output of fixed length ℓ, i.e. with $\mathsf{Rng}(\mathsf{FS}) = \{0,1\}^\ell$, denoting them as *single output-length* function spaces $\mathsf{SOL}(\mathcal{D}, \ell)$.

Following [11], we denote by *functor* a map between a starting function space SS and an ending function space ES. In particular, a functor $\mathsf{F}\colon \mathsf{SS} \to \mathsf{ES}$ takes as oracle a function $h \in \mathsf{SS}$ and returns a function $\mathsf{F}[h] \in \mathsf{ES}$. For a random $h \in \mathsf{SS}$, $\mathsf{F}[h]$ is said to be indifferentiable from a random $e \in \mathsf{ES}$, if there exists an efficient *simulator* S having oracle access to e, such that no adversary is able to distinguish between the behaviour of a pair of oracles, which is either $(\mathsf{F}[h], h)$ or $(e, \mathsf{S}[e])$.

The functors that we consider, apply to function spaces FS where functions are tuples $\mathsf{H} = (\mathsf{H}^{(1)}, \ldots, \mathsf{H}^{(n)})$ of n functions. The integer n is called the *arity* of the function space, and we can view each $\mathsf{H} \in \mathsf{FS}$ as a two-argument function, taking as input (i, X) with $i \in [1, n]$ and returnig $\mathsf{H}^{(i)}(X)$.

f-Functor. In the following, the starting space $\mathsf{SS} = \mathsf{SOL}(\{0,1\}^*, \mathsf{OL}(N))$ is a single output-length function space with domain $\{0,1\}^*$ and outputs of length $\mathsf{OL}(N)$, where $\mathsf{OL}(n) = t(n)\lceil \log_2 n \rceil$. The ending space ES is an arity-$\bar{N}$ function spaces that has single output-length function space with outputs of length $\mathsf{OL}(1), \ldots, \mathsf{OL}(\bar{N})$.

Let $\boldsymbol{f} = \{f_n\colon [0,N]^{t(N)} \to [0,n]^{t(n)}\}$ be a family of functions as defined in Appendix B.1, and let $\boldsymbol{p}$ be a vector of *prefix-free* strings, i.e. such that, for any $i \neq j$, $\boldsymbol{p}[i]$ is not a prefix of $\boldsymbol{p}[j]$. To simplify the notation, we consider $\bar{f}_n$ the function that first maps elements in $\{0,1\}^{\mathsf{OL}(N)}$ to $\{0,N\}^{t(N)}$, applies f_n and finally maps back to $\{0,n\}^{t(n)}$. The functor $\mathsf{F}_{\boldsymbol{f},\boldsymbol{p}}\colon \mathsf{SS} \to \mathsf{ES}$ corresponding to $\boldsymbol{f}$ and $\boldsymbol{p}$ is defined by $\mathsf{F}_{\boldsymbol{f},\boldsymbol{p}}[h](n, X) = \bar{f}_n\big(h(\boldsymbol{p}[n] \parallel X)\big)$. In more details, $\mathsf{F}_{\boldsymbol{f},\boldsymbol{p}}$ takes a random oracle h from SS and defines random oracles $\mathsf{H}^{(n)}$ with outputs length $\mathsf{OL}(n)$ for $n \in [1, \ldots, \bar{N}]$. When given as input X, $\mathsf{H}^{(n)}$ prefixes $\boldsymbol{p}[n]$ to X, applies h to $\boldsymbol{p}[n] \parallel X$ to get Y and finally returns $\bar{f}_n(Y)$.

Proving Indifferentiability. We now show that each $\mathsf{H}^{(n)}$ is read-only indifferentiable from a random oracle by applying Theorem 4.2 of [11]. In particular,

we need to show that $\mathsf{F}_{\boldsymbol{f},\boldsymbol{p}}$ is a translating functor which is invertible over the domain of ES [11, Section 4.2]. To show that $\mathsf{F}_{\boldsymbol{f},\boldsymbol{p}}$ is a translating functor, define a query translator $\mathsf{QT} = \mathsf{QT}_{\boldsymbol{f},\boldsymbol{p}}$ by $\mathsf{QT}_{\boldsymbol{f},\boldsymbol{p}}(n, X) = (\boldsymbol{p}[n] \parallel X)$ and an answer translator $\mathsf{AT} = \mathsf{AT}_{\boldsymbol{f},\boldsymbol{p}}$ such that, on input $(n, X) \in \mathsf{Dom}(\mathsf{ES})$ and $\boldsymbol{V} \in \mathsf{Rng}(\mathsf{SS})^*$, returns $\bar{f}_n(\boldsymbol{V}[1])$, i.e. applies $\bar{f}_n$ to the first entry of $\boldsymbol{V}$. It is easy to verify that the translating functor $\mathsf{TF}_{\mathsf{QT},\mathsf{AT}} \colon \mathsf{SS} \to \mathsf{ES}$ coincides with $\mathsf{F}_{\boldsymbol{f},\boldsymbol{p}}$.

We proceed to show that $\mathsf{F}_{\boldsymbol{f},\boldsymbol{p}}$ is invertible by defining inverses for the query and answer translators as detailed in Algorithm 6. To prove that $\mathsf{QTI} = \mathsf{QTI}_{\boldsymbol{f},\boldsymbol{p}}$ and $\mathsf{ATI} = \mathsf{ATI}_{\boldsymbol{f},\boldsymbol{p}}$ are inverses for QT and AT over $\mathsf{Dom}(\mathsf{ES})$, we need to show that they satisfy two conditions. The first is that for all $e \in \mathsf{ES}$ and all $W = (n, X) \in \mathsf{Dom}(\mathsf{ES})$, Equation (2) of [11] holds:

$$\begin{aligned}
\mathsf{TF}_{\mathsf{QT},\mathsf{AT}}[\mathsf{P}[e]_{\mathsf{QTI},\mathsf{ATI}}](n, X) &= \bar{f}_n\big(\mathsf{P}[e]_{\mathsf{QTI},\mathsf{ATI}}(\boldsymbol{p}[n] \parallel X)\big) \\
&= \bar{f}_n\big(\mathsf{ATI}(\boldsymbol{p}[n] \parallel X, (e(n, X)))\big) \\
&= \bar{f}_n\big(\bar{f}_n^{-1}(e(n, X))\big) \\
&= e(n, X).
\end{aligned}$$

The second condition for invertibility on $\mathsf{Dom}(\mathsf{ES})$ requires that $\mathsf{QTI}(U) \neq ()$ for any $U \in \{\mathsf{QT}(W)[i] \mid W \in \mathsf{Dom}(\mathsf{ES})\}$. Since all $W \in \mathsf{Dom}(\mathsf{ES})$ are of the form (n, X), we have that $\mathsf{QT}(W) = (\boldsymbol{p}[n] \parallel X)$ and $\mathsf{QTI}(\boldsymbol{p}[n] \parallel X) \neq ()$.

Finally, to apply Theorem 4.2 of [11] and conclude read-only indifferentiability, we need to prove that $(\mathsf{QTI}, \mathsf{ATI})$ provide *perfect translation indistinguishability*. In particular, if e is randomly chosen from ES we need to show that $\mathsf{P}[e]_{\mathsf{QTI},\mathsf{ATI}}$, on an input $U \in \mathsf{Dom}(\mathsf{SS})$ such that $\mathsf{QTI}(U) \neq ()$, is distributed like the output on U of a random function $h \in \mathsf{SS}$. Let $U \in \mathsf{Dom}(\mathsf{SS})$ such that $\mathsf{QTI}(U) \neq ()$, then U is of the form $\boldsymbol{p}[n] \parallel X$. It follows that $\mathsf{P}[e]_{\mathsf{QTI},\mathsf{ATI}}(U) = \bar{f}_n^{-1}(e(n, X))$. Due to properties of $\bar{f}_n$ (as detailed in the previous section), the output of $\bar{f}_n^{-1}(e(n, X))$ is indistinguishable from the output of $h(\boldsymbol{p}[n] \parallel X)$, with $h \leftarrow\!\!\$\ \mathsf{SS}$.

C Additional Evaluation Data

C.1 LESS

LESS [17,8] is a signature scheme based on the hardness of the *Linear Code Equivalence*. The parameters proposed in [7] with respect to NIST security levels I, III, and V are reported in Table 5. Recently, Chou, Persichetti, and Santini [31] introduced another notion of equivalence for linear codes and proved that it reduces to linear equivalence. This notion uses *canonical forms* to achieve further compression of group elements transmitted in the response. This optimization is not currently part of the LESS specification, and we refer to [31] for more details.

Figure 2b shows the compression rates as the number of users increases and Figure 2c similarly shows the repetitions rates upon varying the number of users. Here the best results are obtained for level III and V parameters. Finally, in Figure 2a presents how the Fixed Weight parameter ω' changes with an increasing number of users N.

Table 5: Proposed parameters for LESS [7] with corresponding signature sizes.

Set	**NIST Cat.**	**Action Params**			**Signature Params**			**|sig|**
		n	k	q	t	ω	s	(KiB)
LESS-1b	I	252	126	127	247	30	2	8.4
LESS-1i					244	20	4	5.8
LESS-1s					198	17	8	5.0
LESS-3b	III	400	200	127	759	33	2	16.8
LESS-3s					895	26	3	13.4
LESS-5b	V	548	274	127	1352	40	2	29.8
LESS-5b					907	37	3	26.6

Table 6: Proposed parameters for MEDS [29] with corresponding signature sizes.

Set	**NIST Cat.**	**Action Params**		**Signature Params**			**|sig|**
		q	(n, m, k)	t	ω	s	(KiB)
MEDS-9923	I	4093	14	1152	14	4	9.7
MEDS-13320				192	20	5	12.7
MEDS-41711	III	4093	22	608	26	4	40.1
MEDS-69497				160	36	5	53.5
MEDS-134180	V	2039	30	192	52	5	129.4
MEDS-167717				112	66	6	161.6

C.2 MEDS

MEDS [30] is a signature scheme based on the hardness of the *Matrix Code Equivalence*. In the following, we consider the parameters proposed in [29] with respect to NIST security levels I, III, and V as reported in Table 6. The proposed parameterizations for MEDS [29] include the unbalanced challenges with seed tree optimizations and the use of multiple public keys. MEDS also proposes a technique to reduce the size of the signature by representing group elements by linear combinations of a known base $(\mathbf{A}_1, \ldots, \mathbf{A}_k)$. In particular, a matrix $\mathbf{A} \in \mathbb{F}_q^{m \times n}$ can be written as $\mathbf{A} = \sum_i \lambda_i \mathbf{A}_i$, requiring only the $\lambda_i \in \mathbb{F}_q$ to be transmitted, obtaining a compression to $k\lceil \log_2 q \rceil$ bits. This optimization is not currently part of the MEDS proposed parameters, and we refer to [29] for more details.

The trend of the fixed-weight parameter ω' for the multi-signature as the number of users changes is shown in Figure 3a. Figure 3b and Figure 3c show, respectively, the compression rates and the repetitions rates as the number of users increases.

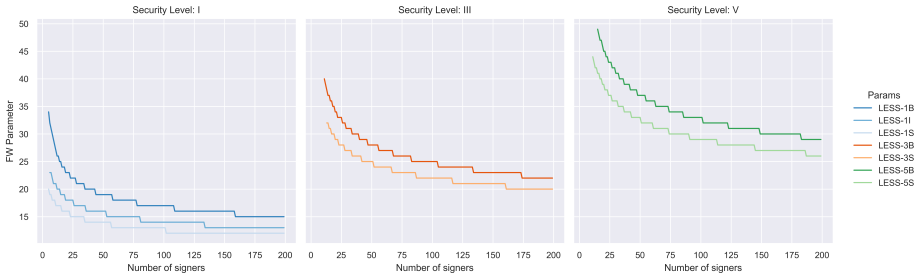
Table 7: Proposed parameters for ALTEQ [19] with corresponding signature sizes.

Set	**NIST Cat.**	**Action Params**		**Centr. Sig. Params**			**|sig|**
		n	q	t	ω	s	(KiB)
Balanced-I	I	13	$2^{32}-5$	84	22	7	15.6
Short-I				16	14	458	9.3
Balanced-III	III	20	$2^{32}-5$	201	28	7	47.9
Short-III				39	20	229	31.8

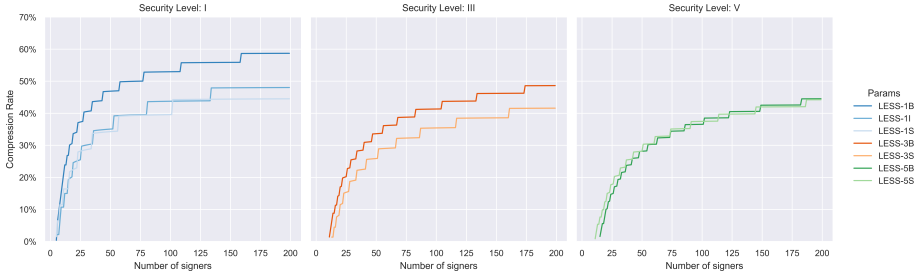
C.3 ALTEQ

ALTEQ [70] is a signature scheme based on the hardness of the *Alternating Trilinear Form Equivalence* problem. In Table 7, we report the parameters proposed in [19] with respect to NIST security levels I and III.

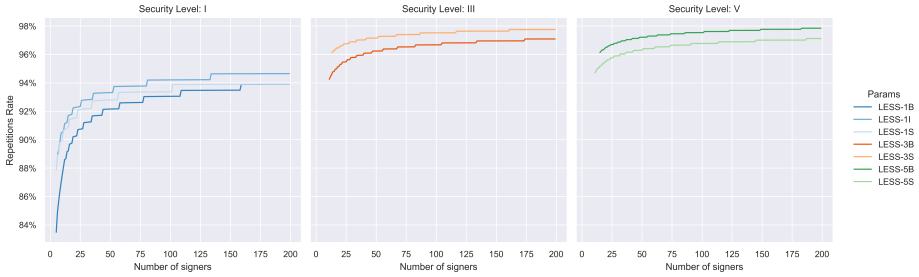
The trend of the fixed-weight parameter ω' for the multi-signature as the number of users changes is shown in Figure 4a while Figure 4b shows the compression rates as the number of users increases. Figure 4c similarly shows the repetitions rates upon varying the number of users.



(a) Fixed-Weight parameters for LESS parameters.

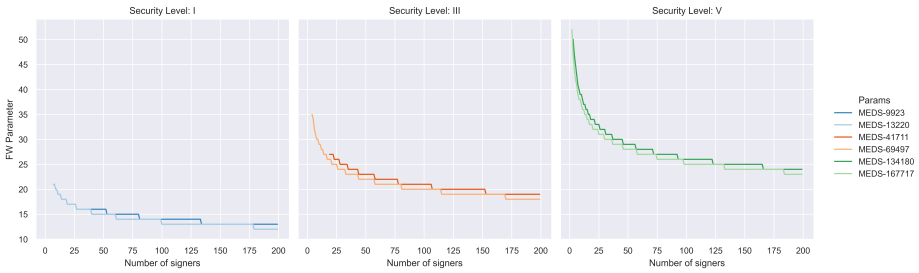


(b) Compression rates for LESS parameters.

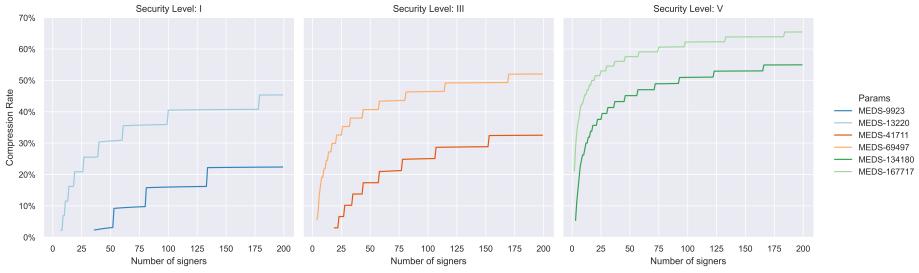


(c) Repetitions rates for LESS parameters.

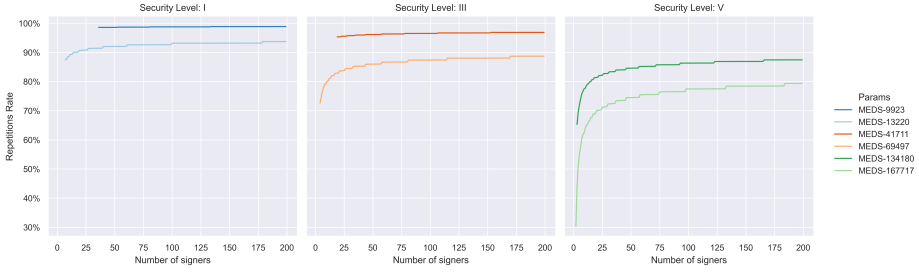
Fig. 2: Multi-signature rates and parameters for LESS.



(a) Fixed-Weight parameters for MEDS parameters.

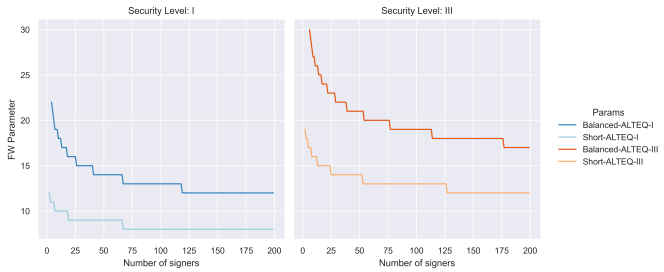


(b) Compression rates for MEDS parameters.

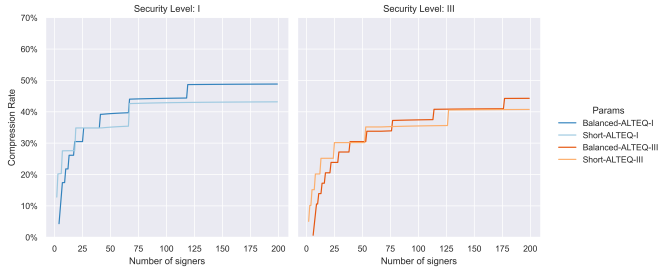


(c) Repetitions rates for MEDS parameters.

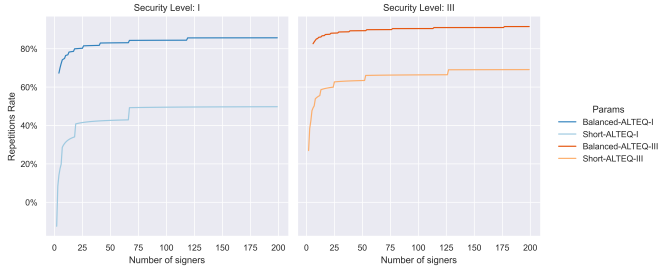
Fig. 3: Multi-signature rates and parameters for MEDS.



(a) Fixed-Weight parameters for ALTEQ parameters.



(b) Compression rates for ALTEQ parameters.



(c) Repetitions rates for ALTEQ parameters.

Fig. 4: Multi-signature rates and parameters for ALTEQ.