Enhancing Threshold Group Action Signature Schemes: Adaptive Security and Scalability Improvements

Michele Battagliola[1], Giacomo Borin[2,3], Giovanni Di Crescenzo[4], Alessio Meneghetti[5], and Edoardo Persichetti[6]

[1] University Polytechnic of Marche,
battagliola.michele@proton.me
[2] University of Zurich,
[3] IBM Research - Zurich,
grass@gbor.in
[4] Peraton Labs
gdicrescenzo@peratonlabs.com
[5] University of Trento,
alessio.meneghetti@unitn.it
[6] Florida Atlantic University,
epersichetti@fau.edu

Abstract. Designing post-quantum digital signatures is a very active research area at present, with several protocols being developed, based on a variety of mathematical assumptions. Many of these signatures schemes can be used as a basis to define more advanced schemes, such as ring or threshold signatures, where multiple parties are involved in the signing process. Unfortunately, the majority of these protocols only considers a static adversary, that must declare which parties to corrupt at the beginning of the execution. However, a stronger security notion can be achieved, namely security against adaptive adversaries, that can corrupt parties at any times.

In this paper we tackle the challenges of designing a post-quantum adaptively secure threshold signature scheme: starting from the GRASS signature scheme, which is only static secure, we show that it is possible to turn it into an adaptive secure threshold signature that we call GRASS+. In particular, we introduce two variants of the classical GAIP problem and discuss their security. We prove that our protocol is adaptively secure in the Random Oracle Model, if the adversary corrupts only $\frac{t}{2}$ parties. We are also able to prove that GRASS+ achieves full adaptive security, with a corruption threshold of t, in the Black Box Group Action Model with Random Oracle. Finally, we improve the performance of the scheme by exploiting a better secret sharing, inspired from the work of Desmedt, Di Crescenzo, and Burmester from ASIACRYPT'94.

Keywords: Post-Quantum Cryptography, Digital Signature, Threshold Signatures, Group Action

1 Introduction

A (t, n)-threshold digital signature scheme is a protocol designed to distribute the privilege to sign messages among n parties, such that any subset of at least t of them is able to sign. Moreover, we require that any subset with t or fewer parties is not able to sign any message. Recently, driven by both the NIST calls for Post-Quantum Standardization [38, 39] and the call for Multi-Party Threshold Schemes [15], many researchers have started to investigate post-quantum threshold digital signature schemes. In this regard, we can mention [8, 18, 24, 27] that present threshold signatures from isogeny assumptions, in particular based on the CSi-FiSh group action; a threshold signature based on Raccoon [41]; a framework for hash-based threshold signatures [36] and a group-action-based threshold signature that uses the group action as black box [6]. Unfortunately, all these works only consider static security, and they are not secure against an adaptive adversary.

1.1 Static and Adaptive Security

In the static setting, the adversary decides which parties to corrupt at the beginning of the protocol, before any message exchange. This model places a great restriction on the adversary's power: indeed in realistic protocols, malicious entities may corrupt a party at any time, and often they do so after seeing some messages.

Adaptive security is a strictly stronger notion and captures this second case. A naive idea to transform a statically secure scheme into an adaptively secure one is to guess the corrupted parties and aborting if incorrect. As noted in [19] by Canetti et al., the main problem with this approach is that the resulting proof of security results in a tightness loss of $\binom{n}{t-1}$, that grows exponentially in the value of the threshold. To solve this issue, the authors of [19] proposed a method which revolves around secure erasures of the secret state, which, however, is not easily enforced in practice. Other alternatives, like [20, 34] usually rely on heavyweight tools, such as non-committing encryption.

The recent NIST call for multi-party threshold schemes included adaptive security as a main goal, ideally supporting up to 1024 participants. This caused a surge in interest in adaptively secure threshold signature schemes, in particular with regard to threshold Schnorr signatures, with notable examples such as [2, 3, 26] (whose techniques paved the way for our work), and lattice-based signatures like [30, 35].

1.2 Our Contribution

As a first result, we present an improvement of the GRASS key generation algorithm. In particular, GRASS relies on Replicated Secret Sharing, that becomes quite unpractical for large n. We adapt a sharing introduced in [28] and get a new solution that requires a number of rounds only linear in t to perform the

signature procedure. We then analyze the performance and security of the sharing scheme, obtaining new and non-asymptotic upper bounds on the number of shares compared to the one introduced in [28]. This new method significantly reduces the number of shares that each party needs to store, even if it still does not achieve the same efficiency as Linear Secret Sharing Schemes.

Next, we present our main result: the design of an adaptively secure post-quantum threshold signature scheme. To do so, we introduce and study two new problems: Chain-GAIP and Graph GAIP (Problems 2 and 3), that translate the classic One More Discrete Logarithm (OMDL) problem in the context of group actions.

We modify the signature algorithm of the GRASS signature scheme [6] to achieve adaptive security, by inserting online-extractable ZKPs[7], and we reduce the adaptive security of the full n-out-of-n threshold scheme to the hardness of n-Chain GAIP, first against $\frac{n-1}{2}$ corruptions in the random oracle model and then against $n-1$ corruptions in the Black Box Group Action Model from [12]. Finally, we discuss when it is possible to extend these results to the more general t-out-of-n schemes and under which assumptions.

1.3 Outline

We begin in Section 2, where we provide all the necessary preliminary definitions and notions used in the paper. In Section 3 we introduce two new conjectured hard problems, Chain-GAIP and Graph GAIP, that are used to prove the security of our protocol. Then, in Section 4 we present and study the new key generation algorithm. In Section 5 we present the new signature protocol and next we prove its security in Section 6.

2 Preliminaries

In this work we use the symbol $\xleftarrow{\$}$ to denote sampling from uniform distribution, while the symbol $\leftarrow$ denotes that the right value is assigned to the left variable. We use again the symbol $\leftarrow$ to denote that we assign to the left variable the output of a (potentially randomized) algorithm. We denote the security parameter as λ. For any positive integer n we define $[n] := \{0, ..., n-1\}$.

2.1 Cryptographic Group Actions

A *group action* $(G, X, \star)$ can be described as a function, as shown below, where X is a set and G a group.

$$\begin{aligned} \star : G \times X &\to X \\ (g, x) &\mapsto g \star x \end{aligned}$$

[7] ZKP(s) stands for Zero Knowledge Proof(s)

A group action's only requirement is to be *compatible* with the group; using multiplicative notation for G and denoting with e its identity element, this means that for all $x \in X$ we have $e \star x = x$ and that moreover for all $g, h \in G$, it holds that $h \star (g \star x) = (h \cdot g) \star x$. The orbit of a set element is the set $\mathcal{O}(x) := \{g \star x \mid g \in G\}$. We say that two set elements $x', x'' \in X$ are *linked* if we know a group element $g \in G$ such that $x'' = g \star x'$ or $x' = g \star x''$. A group action is also said to be:

- *Transitive*, if for every $x, y \in X$, there exists $g \in G$ such that $y = g \star x$;
- *Faithful*, if there does not exist a $g \in G$ such that $x = g \star x$ for all $x \in X$, other than the identity;
- *Free*, if an element $g \in G$ is equal to identity whenever there exists an $x \in X$ such that $x = g \star x$;
- *Regular*, if it is free and transitive.

The adjective *cryptographic* is added to indicate that the group action in question has additional properties that are relevant to cryptography. For instance, a cryptographic group action should be *one-way*, i.e., given randomly chosen $x, y \in X$, it should be hard to find $g \in G$ such that $g \star x = y$ (if such a g exists). Indeed, the problem of finding such an element is known as the *vectorization* problem, or sometimes *Group Action Inverse Problem (GAIP)*.

Problem 1 (GAIP). Given $x \in X$ and y uniformly distributed in $\mathcal{O}(x)$, compute an element $g \in G$ such that $y = g \star x$.

2.2 Sigma Protocol for GAIP

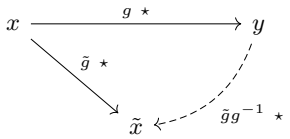


Fig. 1: Sigma protocol for the knowledge of the group action.

Sigma protocols for the knowledge of a solution to the GAIP have been used successfully in many cryptographic protocols and they usually follow the same structure [14]. In this section we summarize the general idea for a generic group action.

Let $(G, X, \star)$ be a group action such that GAIP (Problem 1) is hard and consider $x, y \in X$ and $g \in G$ such that $y = g \star x$. Figure 1 shows how a prover who knows g, can prove its knowledge to a verifier knowing only x, y. The protocol works as follows:

- the prover picks a random $\tilde{g} \in G$ and sends $\tilde{x} = \tilde{g} \star x$ to the verifier,

– the verifier chooses a random bit $b \in \{0, 1\}$ and sends it to the prover,
– the prover sends w to the verifier, where $w = \tilde{g}$ if $b = 0$ or $w = \tilde{g}g^{-1}$ if $b = 1$,
– the verifier accepts if $w \star x = \tilde{x}$ when $b = 0$ or $w \star y = \tilde{x}$ when $b = 1$.

It is easy to see that the above protocol satisfies the classical properties of completeness, special soundness and honest verifier zero-knowledge; thus it is possible to apply the Fiat-Shamir Transform to obtain a secure digital signature. The protocol can then be improved by applying several optimizations, getting different tradeoffs in the signature scheme parameters. More information on the construction of digital signatures from group actions can be found in [14].

Examples of this approach are CSi-Fish [7], MEDS [22], Alteq [11] and LESS [5, 9], with the latter being among the fourteen candidates selected for the second round of the NIST call for post quantum digital signature schemes.

2.3 Threshold Signatures

We briefly summarize here the relevant notions for threshold signature schemes. In a nutshell, a (t, n)-threshold signature is a multi-party protocol that allows any t parties out of a total of n to compute a signature that may be verified against a common public key. We assume that each user has access to a secure, reliable and authenticated private channel with each of the other users, without worrying about specific design and peculiarities of the channel.

Usually, threshold signature protocols involve a key-generation protocol that constructs the key pair $(\mathtt{pk}, \mathtt{sk})$ as well as shares of the private key $\mathtt{sk}_i$, and a multiparty signature protocol TSign, such that any set of t parties who agree on a common message $\mathtt{mes}$ is able to compute a signature, which is verifiable against the public key via the procedure Ver. KeyGen can be executed by a trusted party or by the n parties alone collaborating. In this "decentralized" case, the parties get access to the additional exchanged information.

Even if it is possible to have a more general definition, we tailor the syntax of our definition to both the inherited Σ-protocol-like and sequential round-robin structures of our protocol, which are the same as the original [6]. By sequential round-robin, we mean that the parties take turns to produce the final output, instead of working simultaneously.

Definition 1. *(Threshold Signatures) A threshold signature scheme* TSign *with two round-robin consists of polynomial time algorithms*

$$\mathsf{TSign} = (\mathsf{Setup}, \mathsf{KeyGen}, (\mathsf{TSign}_1, \mathsf{Fin}_1), (\mathsf{TSign}_2, \mathsf{Fin}_2), \mathsf{Ver}),$$

defined as follows:

– $\mathsf{Setup}(1^\lambda) \to \mathtt{par}$*: on input the security parameter* 1^λ*, it outputs the public parameters* $\mathtt{par}$*.*
– $\mathsf{KeyGen}(\mathtt{par}, 1^n, 1^t) \to (\mathtt{pk}, \{\mathtt{sk}_i\}_{[n]})$*: a probabilistic algorithm that takes as input the public parameters, the number of signers and the threshold* t *and outputs the common public key* $\mathtt{pk}$ *and a share* $\mathtt{sk}_i$ *of the private key for each signer.*

– *(*$\mathsf{TSign}_1, \mathsf{Fin}_1), (\mathsf{TSign}_2, \mathsf{Fin}_2)$ *are two pairs of algorithms where* TSign_i *represents one round-robin each and* Fin_i *represents a final broadcast that "finalizes" the round-robin. Both* TSign_1 *and* TSign_2 *are done by each party in the signing set* $\mathcal{S}_{\mathsf{sig}}$ *following the round-robin order and are defined as follows:*

$$
\begin{aligned}
(\mathtt{pm}_{i,1}, \mathtt{st}_{i,1}) &\leftarrow \mathsf{TSign}_1(\mathcal{S}_{\mathsf{sig}}, \mathtt{mes}, \mathtt{sk}_i, \{\mathtt{pm}_{j,1}\}_{j\in\mathcal{S}_{\mathsf{sig}}, j<i}) \\
\mathtt{dcmt}_i &\leftarrow \mathsf{Fin}_1(\mathcal{S}_{\mathsf{sig}}, \mathtt{mes}, \mathtt{sk}_i, \{\mathtt{pm}_{j,1}\}_{j\in\mathcal{S}_{\mathsf{sig}}}) \\
(\mathtt{pm}_{i,2}, \mathtt{st}_{i,2}) &\leftarrow \mathsf{TSign}_2(\mathcal{S}_{\mathsf{sig}}, \mathtt{mes}, \mathtt{sk}_i, \mathtt{st}_{i,1}\{\mathtt{pm}_{j,2}\}_{j\in\mathcal{S}_{\mathsf{sig}}, j<i}, \{\mathtt{pm}_{k,1}, \mathtt{dcmt}_k\}_{k\in\mathcal{S}_{\mathsf{sig}}}) \\
\sigma_i &\leftarrow \mathsf{Fin}_2(\mathcal{S}_{\mathsf{sig}}, \mathtt{mes}, \mathtt{sk}_i, \{\mathtt{pm}_{j,1}\}_{j\in\mathcal{S}_{\mathsf{sig}}}, \{\mathtt{pm}_{j,2}\}_{j\in\mathcal{S}_{\mathsf{sig}}}, \{\mathtt{dcmt}_j\}_{j\in\mathcal{S}_{\mathsf{sig}}})
\end{aligned}
$$

where $\mathtt{pm}_{i,1}$, $\mathtt{pm}_{i,2}$ *are* public messages *broadcast by party* i, $\mathtt{st}_{i,1}, \mathtt{st}_{i,2}$ *are the states of party* i *at the end of each round and the final signature* σ *can be computed deterministically from all the partial signature* σ_i.

– $\mathsf{Ver}(\mathtt{pk}, \mathtt{mes}, \sigma) \to 0/1$*: a deterministic algorithm that takes as input the public key* $\mathtt{pk}$*, a message* $\mathtt{mes}$ *and a signature* σ *and outputs* 1 *if* σ *is valid, else it outputs* 0.

We require that the threshold signature scheme is correct*, i.e., for all security parameters* λ*, all* $1 \leq t \leq n$*, all* $\mathcal{S}_{\mathsf{sig}} \subseteq [n]$ *such that* $|\mathcal{S}_{\mathsf{sig}}| \geq t$*, all messages* $\mathtt{mes}$ *if* $\mathsf{KeyGen}(\mathtt{par}, n, t) \to (\mathtt{pk}, \{(\mathtt{pk}_i, \mathtt{sk}_i)\}_{[n]})$ *then the algorithms* $\mathsf{TSign}_1, \mathsf{Fin}_1, \mathsf{TSign}_2$ *and* Fin_2 *return a valid signature* σ *such that* $\mathsf{Ver}(\mathtt{pk}, \mathtt{mes}, \sigma) = 1$.

Informally speaking, first all the parties engage in a round-robin protocol. Each party i, on input the signing set, its secret key, the message, and all the output from the previous rounds, outputs some public messages $\mathtt{pm}_{i,1}$ and its secret state $\mathtt{st}_{i,1}$. Then a second round-robin is done. There, again, each party i, on input the signing set, its secret key and state, the message, and all the outputs from the first round-robin and from the previous rounds, outputs some public messages $\mathtt{pm}_{i,2}$ and its secret state $\mathtt{st}_{i,2}$. After each round-robin, there is a final broadcast, where each party, on input all the public data, outputs some public data, in particular the second finalization protocol outputs the signature.

Remark 1. The above definition of threshold signature is very complex and notation heavy. The main reason why we need such a definition is to better define security, allowing the adversary maximum freedom in opening parallel sessions. Indeed, when defining the security game for adaptive security, we allow the adversary to freely query the oracle on each of the above four algorithms in any order. Since each algorithm corresponds to one message sent, this simulates the possibility of opening parallel executions.

Adaptive Security. An important notion for threshold signature security is adaptive security, where the adversary is able to corrupt parties dynamically, in contrast to static security, where it is required to declare all the corrupted parties at the beginning of the execution. Here we adapt the definition given in [26] to suit the case of threshold signatures having a round-robin structure.

Definition 2 (Adaptive EUF-CMA). *Let* TSign *be a threshold signature scheme and* $\mathcal{A}$ *be an adversary playing the adaptive EUF-CMA game defined in Figure 2. Let define the advantage of* $\mathcal{A}$ *as:*

$$\boldsymbol{Adv}_{\mathcal{A}}^{\mathsf{a-euf-cma}}(\lambda, \mathsf{frac}) = \Pr(\mathsf{Exp}_{\mathcal{A}}^{\mathsf{a-euf-cma}}(\lambda, \mathsf{frac}) = 1)$$

We say that TSign *is unforgeable against adaptive chosen message attacks with* frac *corruptions if and only if* $\boldsymbol{Adv}_{\mathcal{A}}^{\mathsf{a-euf-cma}}$ *is negligible for every probabilistic polynomial time adversary* $\mathcal{A}$*.*

Informally speaking in the adaptive EUF-CMA game the adversary can interact with the following oracles:

- $\mathcal{O}^{\mathsf{Corrupt}}$: if the total number of corruptions is lower than frac, then the $\mathcal{O}^{\mathsf{Corrupt}}$ returns the private key and all the internal states of the chosen party.
- $\mathcal{O}^{\mathsf{TSign}_i}$: the adversary asks the oracle to perform one round of the round-robin. First the oracle checks that all the previous parties in the round-robin have done their turn (in the case of TSign_2 it also checks whether Fin_1 was executed or not); if so, it executes TSign_i, publishes the public data and stores the private data.
- $\mathcal{O}^{\mathsf{Fin}_i}$: the adversary asks the oracle to perform the finalization protocol. First the oracle checks that the previous round-robin is finished; if so, it performs the finalization algorithm Fin_i on behalf of all the honest parties and sends the result to the adversary. In case of Fin_1, if the adversary refuses (or fails) to execute Fin_1, then the oracle will stop the execution during the TSign_2.

2.4 Black Box Group Action Model

As will become clearer later in Section 6, to prove the adaptive security of GRASS we need to perform a rewind in order to extract all the secrets we need. Unfortunately, this implies that the adversary can corrupt at most $\frac{t-1}{2}$ parties, otherwise we would incur the risk of needing more queries than allowed to the oracle we use in our security assumption, since the adversary could corrupt different parties after the rewind. In order to extract all the group actions after a single forgery, we need to use the Black Box Group Action Model (BBGAM) introduced in [12], which generalizes the Generic Group Model to consider group actions.[8]

In this model, direct computation over set elements is possible only by querying an oracle to do all the computation. In particular, each party is provided with a starting set of set elements $x_0, ..., x_r \in X$ and three oracles:[9]

- $\mathcal{O}^{\mathsf{Eq}}(x, y)$, with $x, y \in X$ that returns 1 if $x = y$, 0 otherwise.
- $\mathcal{O}^{\mathsf{Act}}(g, x)$ that, on input $g \in G$ and a previously seen set element $x \in X$, returns the set element $g \star x$.

[8] We do not consider [29] since they restrict themselves to the abelian group case.

[9] To be very formal, the parties do not have access directly to set elements but instead have access to handles. For the sake of simplicity, with abuse of notation, we will simply write x instead of $\langle x \rangle$, even when referring to handles.

$\mathsf{Exp}^{\mathsf{a-euf-cma}}_{\mathcal{A}}(\lambda, \mathsf{frac})$

$\mathtt{par} \leftarrow \mathsf{Setup}(1^\lambda)$
$\mathtt{Q}_{\mathtt{mes}} \leftarrow \emptyset, \mathtt{Q}_{\mathtt{st}} \leftarrow \emptyset$
$(\mathtt{cor}, \mathtt{st}_{\mathcal{A}}) \leftarrow \mathcal{A}(\mathtt{par})$
if $|\mathtt{cor}| > \mathsf{frac} \vee \mathtt{cor} \not\subset [n]$
  return $\bot$
$\mathtt{hon} \leftarrow [n] \setminus \mathtt{cor}$
$(\mathtt{pk}, \{(\mathtt{pk}_i, \mathtt{sk}_i)\}) \leftarrow \mathsf{KeyGen}(\mathtt{par}, n, t)$
$\mathtt{st}_{\mathcal{A}} \leftarrow (\mathtt{pk}, \{\mathtt{pk}_i, \mathtt{sk}_i\}_{\mathtt{cor}}, \mathtt{st}_{\mathcal{A}})$
$(\mathtt{mes}^*, \sigma^*) \leftarrow \mathcal{A}^{\mathcal{O}^{\mathsf{TSign}_i}, \mathcal{O}^{\mathsf{Fin}_i}, \mathcal{O}^{\mathsf{Corrupt}}}(\mathtt{st}_{\mathcal{A}})$
return $\mathtt{mes}^* \notin \mathtt{Q}_{\mathtt{mes}} \wedge \mathsf{Ver}(\mathtt{pk}, \mathtt{mes}^*, \sigma^*)$

$\mathcal{O}^{\mathsf{TSign}_1}(k, \mathtt{mes}, \mathtt{ssid})$

$\mathtt{Q}_{\mathtt{mes}} \leftarrow \mathtt{Q}_{\mathtt{mes}} \cup \mathtt{mes}$
if $(k \notin \mathtt{hon}) \vee (\mathtt{Q}_{\mathtt{st}}[k, \mathtt{ssid}, 1] \neq \bot)$
  return $\bot$
for $j \in \mathcal{S}_{\mathsf{sig}}, j < k$
  if $\mathtt{Q}_{\mathtt{pm}}[j, \mathtt{ssid}, 1] = \bot$
    return $\bot$
$(\mathtt{pm}^{\mathtt{ssid}}_{k,1}, \mathtt{st}^{\mathtt{ssid}}_{k,1}) \leftarrow \mathsf{TSign}_1(\mathcal{S}_{\mathsf{sig}}, \mathtt{mes}, \mathtt{sk}_k, \{\mathtt{pm}^{\mathtt{ssid}}_{j,1}\}_{j \in \mathcal{S}_{\mathsf{sig}}, j<i})$
$\mathtt{Q}_{\mathtt{st}}[k, \mathtt{ssid}, 1] \leftarrow \mathtt{st}^{\mathtt{ssid}}_{k,1}$
$\mathtt{Q}_{\mathtt{pm}}[k, \mathtt{ssid}, 1] \leftarrow \mathtt{pm}^{\mathtt{ssid}}_{k,1}$
return $\mathtt{pm}^{\mathtt{ssid}}_{k,1}$

$\mathcal{O}^{\mathsf{Corrupt}}(k)$

if $k \notin \mathtt{hon} \vee |\mathtt{cor}| = \mathsf{frac}$
  return $\bot$
$\mathtt{cor} \leftarrow \mathtt{cor} \cup \{k\}$
$\mathtt{hon} \leftarrow \mathtt{hon} \setminus \{k\}$
  // Retrieve all state for party k
$\mathtt{st}_k \leftarrow \mathtt{Q}_{\mathtt{st}}[k, \cdot, \cdot]$
return $(\mathtt{sk}_k, \mathtt{st}_k)$

$\mathcal{O}^{\mathsf{TSign}_2}(k, \mathtt{mes}, \mathtt{ssid}, \{\mathtt{pm}_{j,1}\}_{j \in \mathcal{S}_{\mathsf{sig}}})$

if $(k \notin \mathtt{hon}) \vee (\mathtt{ssid} = \bot)$
  $\vee (\mathtt{Q}_{\mathtt{Fin}}[\mathtt{ssid}] = \bot) \vee (\mathtt{Q}_{\mathtt{pm}}[k, \mathtt{ssid}, 2] \neq \bot)$
  return $\bot$
for $j \in \mathcal{S}_{\mathsf{sig}}, j < k$
  if $\mathtt{Q}_{\mathtt{pm}}[j, \mathtt{ssid}, 2] = \bot$
    return $\bot$
$(\mathtt{pm}^{\mathtt{ssid}}_{k,2}, \mathtt{st}^{\mathtt{ssid}}_{k,2}) \leftarrow \mathsf{TSign}_2(\mathcal{S}_{\mathsf{sig}}, \mathtt{mes}, \mathtt{sk}_k, \{\mathtt{pm}^{\mathtt{ssid}}_{j,2}\}_{j<i}, \{\mathtt{pm}^{\mathtt{ssid}}_{j,2}\}_{j \in \mathcal{S}_{\mathsf{sig}}})$
$\mathtt{Q}_{\mathtt{pm}}[k, \mathtt{ssid}, 2] \leftarrow \mathtt{pm}^{\mathtt{ssid}}_{k,2}$
$\mathtt{Q}_{\mathtt{st}}[k, \mathtt{ssid}, 2] \leftarrow \mathtt{st}^{\mathtt{ssid}}_{k,2}$
return $(\mathtt{pm}^{\mathtt{ssid}}_{k,2})$

$\mathcal{O}^{\mathsf{Fin}_1}(\mathtt{mes}, \mathtt{ssid}, \mathcal{S}_{\mathsf{sig}})$

for $j \in \mathcal{S}_{\mathsf{sig}}$
  if $\mathtt{Q}_{\mathtt{pm}}[j, \mathtt{ssid}, 1] = \bot$
    return $\bot$
for $i \in \mathtt{hon} \cap \mathcal{S}_{\mathsf{sig}}$
  $\mathtt{dcmt}^{\mathtt{ssid}}_i \leftarrow \mathsf{Fin}_1(\mathcal{S}_{\mathsf{sig}}, \mathtt{mes}, \mathtt{sk}_i, \{\mathtt{pm}^{\mathtt{ssid}}_{j,1}\}_{j \in \mathcal{S}_{\mathsf{sig}}})$
$\{\mathtt{dcmt}^{\mathtt{ssid}}_j\}_{j \in \mathtt{cor}} \leftarrow \mathcal{A}(\{\mathtt{dcmt}^{\mathtt{ssid}}_i\}_{i \in \mathtt{hon}})$
$\mathtt{Q}_{\mathtt{Fin}}[\mathtt{ssid}] = 1$
for $j \in \mathtt{cor}$
  if $\mathtt{dcmt}^{\mathtt{ssid}}_j = \bot$
    $\mathtt{Q}_{\mathtt{Fin}}[\mathtt{ssid}] = \bot$
return

$\mathcal{O}^{\mathsf{Fin}_2}(\mathtt{mes}, \mathtt{ssid}, \mathcal{S}_{\mathsf{sig}})$

for $j \in \mathcal{S}_{\mathsf{sig}}$
  if $\mathtt{Q}_{\mathtt{pm}}[j, \mathtt{ssid}, 2] = \bot$
    return $\bot$
for $i \in \mathtt{hon} \cap \mathcal{S}_{\mathsf{sig}}$
  $\sigma_i \leftarrow \mathsf{Fin}_1(\mathcal{S}_{\mathsf{sig}}, \mathtt{mes}, \mathtt{sk}_i, \{\mathtt{pm}^{\mathtt{ssid}}_{j,1}, \mathtt{pm}^{\mathtt{ssid}}_{j,2}, \mathtt{dcmt}^{\mathtt{ssid}}_i\}_{j \in \mathcal{S}_{\mathsf{sig}}})$
return $\{\sigma_i\}_{i \in \mathtt{hon}}$

Fig. 2: Security game for adaptive EUF-CMA.

– $\mathcal{O}^{\mathsf{sample}}(s)$ that, on input a seed s, returns a random element $x \in X$, potentially on a different orbit from the previously returned set elements. For some group action we can even not allow this oracle.

The core idea behind the model is that the only way for the adversary to derive additional meaningful set elements is to query the action oracle $\mathcal{O}^{\mathsf{Act}}$, i.e., compute a group action on a defined set element. In this way, given a set element $\bar{x}$ resulted from one of the party computations, we can always look at all the previous oracle queries. Thus, we can find a group element $\bar{g}$ such that $\bar{x} = \bar{g} \star x_i$ for $i = 0, ..., r$ or, if $\mathcal{O}^{\mathsf{sample}}$ is allowed, $\bar{x} = \bar{g} \star x$ for some x uniformly distributed in X, which is obtained from $\mathcal{O}^{\mathsf{sample}}$. Note that for our setting we consider the case in which all the set elements $x_0, ..., x_r$ are in the same orbit. Moreover, if the adversary tries to cheat by sending elements from a different orbit, the malicious behavior will be caught thanks to the ZKPs used.

The BBGAM in Practice. The reliance of the model clearly depends on the group actions we want to model. To justify it we need to argue that, with respect to a particular group action, the only way to obtain a new set element is by applying a group action on a known set element or (eventually) by random sampling (i.e., using $\mathcal{O}^{\mathsf{sample}}$).

It is well known that this model can safely be applied to the group actions based on isogenies of supersingular elliptic curve, like [21]. In fact, it is a major open problem in isogeny based cryptography to compute a valid supersingular elliptic curve not starting from an isogeny applied on a known one, this is known as the *hashing to the supersingular elliptic curve graph problem* [13], so it may even make sense to not even allow the use of $\mathcal{O}^{\mathsf{sample}}$. More detailed discussion can be found e.g., from [40, Section 4.3].

For the other group actions, like the ones used for LESS [5], MEDS [22] and ALTEQ [11], it is always possible to generate a random set element by sampling a random linear code or tensor, but this lies in the same orbit only with negligible probability, thus giving no advantage to the adversary (moreover, this behavior would be quickly caught by the honest parties, since the protocol uses ZKPs to prevent the usage of set elements that are not linked to other ones). This can be verified with simple computations; e.g., for LESS, the number of monomial maps, i.e., of group elements (that is an upper bound on the orbits sizes), is $(q-1)^n n!$, which for the regime of interest ($n > 200$) is negligible with respect to the number of possible linear codes, $\approx q^{\frac{n^2}{4}}$. We assume this is the case for the group actions considered in this work.

Even if, to the best of our knowledge, it is not possible to use these random elements to get additional information, we must include the possibility of sampling random elements in the set X since there is no efficient way to test if two set elements lie in the same orbit, i.e., to solve the decisional version of GAIP. In the case of LESS, it is even known that GAIP reduces to the problem of deciding if two codes are equivalent [4, 10].

Again, as far as we know, starting from a linear code or an alternating tensor in the literature, there are no other meaningful ways to obtain new objects

related to them. For example, one could try to flip or reorder some entries, but these would yield set elements which, from a group action point of view, unrelated to the starting ones. We believe that, for now, these observations justify the use of the model for group actions cited above. Note that for $[m\times m, k]$-matrix codes, in principle it would be possible to obtain a new code by transposing the matrices in the code. This new code preserves part of the algebraic relations between group elements and set elements, so, even if we do not know how to exploit this to get a meaningful attack, we have to admit that for the group action used in MEDS, the use of the BBGAM is a stronger assumption than for other group action frameworks.

3 Chain and Graph Versions of GAIP

In this Section we introduce two new problems, to be used in the security proof, that aim to generalize the *One More Discrete Logarithm* Problem [37].

3.1 Chain-GAIP

Problem 2 (n-Chain-GAIP). Consider $x \in X$ and $g_0, ..., g_{n-1} \in G$ chosen uniformly at random. Let $x_{i+1} = g_i \star x_i$ for all $i = 0, ..., n-1$, where we have defined x_0 as x. Given $\{x_i\}_{i=0,...,n}$ and access to an oracle that, on input (x_i, x_{i+1}) returns g_i for at most $n-1$ queries, find g such that $x_n = g \star x_0$.

As we can expect, our reduction incurs a security loss which is proportional to n, the number of elements in the chain.

Proposition 1. *If there exists probabilistic polynomial-time algorithm $\mathcal{A}_{\mathsf{Chain}}$ that solves the n-Chain-GAIP problem with probability ϵ and in time T, then there exists a probabilistic polynomial-time algorithm $\mathcal{A}_{\mathsf{GAIP}}$ that solves GAIP in the same time T and with probability ϵ/n.*

Proof. Given as input a GAIP challenge $(x, y) \in X \times X$ the algorithm $\mathcal{A}_{\mathsf{GAIP}}$:

1. samples $n-1$ group elements $g_0, ..., g_{\ell-1}, g_{\ell+1}, .., g_{n-1} \in G$, with $\ell \in \{0, ..., n-1\}$ chosen uniformly at random,
2. sets $x_0 = x, x_n = y$. Then *completes* the chain by defining

$$\begin{cases} x_{i+1} = g_i \star x_i \text{ for } 0 < i < \ell \\ x_i = g_i^{-1} \star x_{i+1} \text{ for } \ell < i < n \end{cases} . \tag{1}$$

3. $\mathcal{A}_{\mathsf{GAIP}}$ sends the chain to $\mathcal{A}_{\mathsf{Chain}}$ and acts as oracle, returning g_i when queried on (x_i, x_{i+1}) for $i \neq \ell$ and failing otherwise.

It is immediate to see that as long as $\mathcal{A}_{\mathsf{Chain}}$ solves the n-Chain-GAIP instance without querying $(x_{\ell-1}, x_\ell)$, then $\mathcal{A}_{\mathsf{GAIP}}$ can return the n-Chain GAIP solution as a GAIP solution. Under the claim that the chain x_i is a valid n-Chain-GAIP instance, with a probability distribution independent of ℓ, the algorithm solves

GAIP with probability ϵ/n. Let idx be the index of the pair that is not queried in an execution of $\mathcal{A}_{\mathsf{Chain}}$[10] interacting with the n-Chain-GAIP oracle, since, under our claim, ℓ is independent of the distribution from item 2 we have

$$\Pr[\mathcal{A}_{\mathsf{Chain}} \text{ succeeds } \wedge \mathsf{idx} = \ell] = \epsilon \cdot \frac{1}{n} .$$

This property is preserved also when interacting with $\mathcal{A}_{\mathsf{GAIP}}$ (item 3) since the distribution of group elements is the same, thus $\mathcal{A}_{\mathsf{GAIP}}$ can extract a valid solution for $(x_{\mathtt{ch}}, y_{\mathtt{ch}})$ with probability ϵ/n.

To prove Proposition 1 we only need to prove the above claim on the distributions of the elements from item 2. Since the group elements $g_1, ..., g_{\ell-1}, g_{\ell+1}, .., g_n$ are uniformly distributed in G then also both g_i^{-1} are uniformly distributed on G. Finally, since the GAIP solution g with $y = g \star x$ is uniform on G so it is the secret group element g_ℓ linking $x_\ell, x_{\ell+1}$.

As said, the security loss with respect to GAIP is at most proportional to n. Luckily, since n is the number of parties, it is expected to be polynomial, so this does not cause any major issue with the scheme. Since it is clear that any solver to GAIP is a solver to n-Chain GAIP (just need to directly tackle the instance and discard the chain related information), the cost of solving n-Chain GAIP with respect to the cost of solving GAIP lies somewhat in between 1 and $1/n$.

However, from a heuristic standpoint, it is reasonable to believe that any algorithm able to solve GAIP should not gain any benefit from knowing other group action that are sampled independently. For example, we may argue that a solver could try to solve these n GAIP *intermediate* instances independently, to then stop at the first success, thus linearly increasing the probability of success, and ask the remaining $n - 1$ instances to the oracle. Anyway this does not decrease the overall cost of the attack since it still requires n parallel independent executions.

3.2 Graph-GAIP

To prove the adaptive security of schemes with more complicated sharing mechanisms, we need to introduce a second problem. Instead of having a single chain, we consider a graph, where the vertices are set elements and the edges are the actions that relate them. Formally we have the following:

Problem 3 (N-Graph-GAIP). Let $x \in X$ and $g \in G$ be chosen uniformly and define $y = g \star x$. Consider N set elements $x_1, ..., x_N \in \mathcal{O}(x)$ sampled uniformly at random and the graph $\mathcal{G} = (V, E)$ with vertices $V = \{x, y\} \cup \{x_1, ..., x_N\}$ and no edges $E = \emptyset$. Find g such that $y = g \star x$ having access to a solver oracle $\mathcal{O}^{\mathcal{G}}$ that on input two set elements $x', y' \in V$:

- if x' is linked to x (resp. to y) in the graph $\mathcal{G}$ and y' is linked to y (resp. to x) in the graph $\mathcal{G}$, return $\perp$;

[10] wlog we can suppose that it always perform $n - 1$ queries

– else add $\{x', y'\}$ to E and return $g' \in G$ such that $y' = g' \star x'$.

What essentially oracle $\mathcal{O}^{\mathcal{G}}$ does is providing links, i.e., solutions to the group action inversion problem, for all the possible pairs of set elements provided at the start, as long as they cannot be composed to get a link between x and y, that is equivalent to get a solution for the GAIP on x, y. It is important that the set elements are fixed from the start, otherwise the adversary could perform a trivial attack by creating two random set element $\tilde{x} = \tilde{g} \star x$ and $\tilde{y} = \tilde{g} \star y$, asking a solution g' for $\tilde{x}, \tilde{y}$ and find a solution $\tilde{g}^{-1} g' \tilde{g}$.

As for n-Chain GAIP we can get reductions to GAIP by trying to generate the set elements ourselves and trying to guess the queries in advance, however in this case the security loss would be exponential in the number N of set elements, compromising the utility of the reduction. This is because, differently from n-Chain GAIP there is no restriction on the allowed queries to the oracle, i.e., no fixed topology of the graph $\mathcal{G}$ but for having two disconnected components[11] containing x and y. However, as before, we have no reason to believe that in practice the additional information provided to a solver can be used meaningfully to solve the GAIP between x and y.

4 Improved Secret Sharing

In this section we extend the secret sharing scheme from [28] to work for group actions, show an improved analysis of the scheme, and discuss its main properties of interest for our threshold signature protocol.

In [28], the authors present a multiplicative secret sharing scheme for the (t, n)-threshold access structure, for any $1 \leq t \leq n$, and any (non-Abelian) group with efficiently computable operation and inverses, and show its application to threshold cryptography (specifically, threshold zero-knowledge proofs of knowledge of secrets encoded as graph isomorphisms). Here, we adapt this scheme to work for any group action, show an improved analysis of the size of the shares. The resulting scheme, denoted as $(\mathsf{RecKeyGen}, \mathsf{Recover})$, is detailed in Figure 3.

Algorithm $\mathsf{RecKeyGen}$ needs to be run by a Trusted Third Party Dealer, that recursively shares the secret key $\mathtt{sk} = g$ associated to the public key $\mathtt{pk} = (x, y)$, i.e., such that $y = g \star x$, between n users $\mathcal{P}$ with a t-out-of-n sharing. Whenever g', x', LBL are sent to the user P, this party will store g', x' under the label LBL. When a set $\mathcal{T}$ of t users wants to recover the secret, each user runs algorithm $\mathsf{Recover}$ which uses the LBL labels to find the correct share to use.

Informal Idea. The core idea is to use a recursive strategy. To do a 1-out-of-n sharing the dealer just forward g', x to all the n users and for an n-out-of-n

[11] i.e., two disjoint non-empty subsets of the graph such that no vertex in the first set is linked to a vertex in the second one

$\mathsf{RecKeyGen}(g, x, y, \mathcal{P}, t, \mathsf{LBL}) \to \{(g_i, x_i, \mathsf{LBL}_i) : i\}$

1 : $n \leftarrow |\mathcal{P}|$
2 : $x_0 \leftarrow x$
3 : parse $\mathcal{P} \to \{P_0, ..., P_{n-1}\}$
4 : **if** $t = 1$
5 : **for** $P \in \mathcal{P}$:
6 : **Send** g, x, LBL to P
7 : **if** $t = n > 1$
8 : $g_i \xleftarrow{\$} G, i = 1, ..., n-1$
9 : $g_0 \leftarrow (g_1 \cdot ... \cdot g_{n-1})^{-1} \cdot g$
10 : **for** $i = 0, ..., n-1$
11 : $x_{i+1} \leftarrow g_i \star x_i$
12 : **Send** g_i, x_i, LBL to P_i
13 : **if** $1 < t < n$
14 : $c \leftarrow \lfloor n/2 \rfloor$
15 : $\mathcal{P}_{\mathsf{left}} \leftarrow \{P_0, ..., P_{c-1}\}$
16 : $\mathcal{P}_{\mathsf{right}} \leftarrow \{P_c, ..., P_{n-1}\}$
17 : **for** $\ell = \max\left(0, t - \left\lceil \frac{n}{2} \right\rceil\right), ..., \min\left(t, \left\lfloor \frac{n}{2} \right\rfloor\right)$
18 : **if** $\ell = 0$
19 : $\mathsf{RecKeyGen}(g, x, y, \mathcal{P}_{\mathsf{left}}, t, \mathsf{LBL} \| (\mathcal{P}_{\mathsf{left}}, t))$
20 : **elseif** $\ell = t$
21 : $\mathsf{RecKeyGen}(g, x, y, \mathcal{P}_{\mathsf{right}}, t, \mathsf{LBL} \| (\mathcal{P}_{\mathsf{right}}, t))$
22 : **else**
23 : $g_1 \xleftarrow{\$} G$
24 : $x_1 \leftarrow g_1 \star x_0$
25 : $g_2 \leftarrow g \cdot g_1^{-1}$
26 : $\mathsf{RecKeyGen}(g_1, x, x_1, \mathcal{P}_{\mathsf{left}}, t - \ell, \mathsf{LBL} \| (\mathcal{P}_{\mathsf{left}}, t - \ell))$
27 : $\mathsf{RecKeyGen}(g_2, x_1, y, \mathcal{P}_{\mathsf{right}}, \ell, \mathsf{LBL} \| (\mathcal{P}_{\mathsf{right}}, \ell))$

$\mathsf{Recover}(P, \mathcal{T}, \mathcal{P}, \mathsf{LBL}) \to g_i, x_i$

1 : $n, t \leftarrow |\mathcal{P}|, |\mathcal{T}|$
2 : assert $P \in \mathcal{T}$
3 : parse $\mathcal{P} \to \{P_0, ..., P_{n-1}\}$
4 : **if** $t = 1$ **or** $t = n$
5 : **get** share received with LBL
6 : **if** $1 < t < n$
7 : $c \leftarrow \lfloor n/2 \rfloor$
8 : $\mathcal{P}_{\mathsf{left}} \leftarrow \{P_0, ..., P_{c-1}\}$
9 : $\mathcal{P}_{\mathsf{right}} \leftarrow \{P_c, ..., P_{n-1}\}$
10 : $\ell \leftarrow \mathcal{T} \cap \mathcal{P}_{\mathsf{right}}$
11 : **if** $\ell = 0$
12 : $\mathsf{Recover}(P, \mathcal{T}, \mathcal{P}_{\mathsf{left}}, \mathsf{LBL} \| (\mathcal{P}_{\mathsf{left}}, t))$
13 : **if** $\ell = t$
14 : $\mathsf{Recover}(P, \mathcal{T}, \mathcal{P}_{\mathsf{right}}, \mathsf{LBL} \| (\mathcal{P}_{\mathsf{right}}, t))$
15 : **else**
16 : **if** $P \in \mathcal{P}_{\mathsf{right}}$
17 : $\mathsf{Recover}(P, \mathcal{T} \cap \mathcal{P}_{\mathsf{right}}, \mathcal{P}_{\mathsf{right}}, \mathsf{LBL} \| (\mathcal{P}_{\mathsf{right}}, \ell))$
18 : **if** $P \in \mathcal{P}_{\mathsf{left}}$
19 : $\mathsf{Recover}(P, \mathcal{T} \cap \mathcal{P}_{\mathsf{left}}, \mathcal{P}_{\mathsf{left}}, \mathsf{LBL} \| (\mathcal{P}_{\mathsf{left}}, t - \ell))$

Fig. 3: Recursive algorithms both for the sharing of the secret g between any subset of t users of $\mathcal{P}$ and the recovery of each individual share, given the set of users $\mathcal{P}$ and the subset of allowed users $\mathcal{T}$ in which P belongs. In the first call to each of the two algorithms, LBL is set as $\emptyset$. The $\mathsf{RecKeyGen}$ algorithm is later used as part of the KeyGen algorithm for the signature, while $\mathsf{Recover}$ is used as part of the signing algorithm, to retrieve the correct share.

sharing, the dealer splits g as $g_{n-1} \cdot \cdots \cdot g_0$, sets $x_0 = x$, computes $x_{i+1} = g_i \star x_i$, then forward g_i, x_i to the i-th party P_i, for $i = 1, .., n$.[12]

Otherwise, to do a t-out-of-n sharing, with $t \neq 1, n$, the user set is split in two sides $\mathcal{P}_{\mathsf{left}}$ and $\mathcal{P}_{\mathsf{right}}$ of approximately the same size $\lfloor \frac{n}{2} \rfloor$ and $\lceil \frac{n}{2} \rceil$. Then for any ℓ such that $\max(0, t - \lceil \frac{n}{2} \rceil) \leq \ell \leq \min(t, \lfloor \frac{n}{2} \rfloor)$, split g as $g_2 \cdot g_1$, then:

- perform recursively an ℓ-out-of-$|\mathcal{P}_{\mathsf{right}}|$ sharing of $g_1, x, g_1 \star x$ on the user set $\mathcal{P}_{\mathsf{right}}$;
- perform recursively a $(t - \ell)$-out-of-$|\mathcal{P}_{\mathsf{left}}|$ sharing of $g_2, g_1 \star x, y$ on the user set $\mathcal{P}_{\mathsf{left}}$.

To allow each user to correctly combine their secret shares the dealer also labels each share with a list containing all the recursive steps performed to arrive to the sharing.

If a subset $\mathcal{T}$ of t users want to recover a subset of shares multiplying to g, each user $P \in \mathcal{T}$ essentially repeats the share generation as before to recompute the correct label. The idea is to recover ℓ^* as $|\mathcal{T} \cap \mathcal{P}_{\mathsf{right}}|$; if $P \in \mathcal{P}_{\mathsf{left}}$ then they compute the label of a $(t - \ell^*)$-out-of-$|\mathcal{P}_{\mathsf{left}}|$ sharing on $\mathcal{T} \cap \mathcal{P}_{\mathsf{left}}$ recursively, or, if $P \in \mathcal{P}_{\mathsf{right}}$, then they compute the label of an ℓ^*-out-of-$|\mathcal{P}_{\mathsf{right}}|$ sharing on $\mathcal{T} \cap \mathcal{P}_{\mathsf{right}}$, again recursively. When $t = 1$ or $t = |\mathcal{P}|$ they can recover the share under the computed label.

In the signature, the algorithm $\mathsf{RecKeyGen}$ is used by the trusted dealer to perform the KeyGen algorithm. The algorithm $\mathsf{Recover}$ is executed by each party at the beginning of every signing session to retrieve the correct share to use in that session.

4.1 Correctness and Secrecy

The above protocol preserves the claim in [28, Theorem 4]; i.e., it is a multiplicative (perfect) secret sharing scheme for the t-out-of-n access structure. Specifically, it satisfies correctness (i.e., any participant subset of cardinality $\geq t$ can recover the secret g), secrecy (i.e., any participant subset of size $\leq t - 1$ obtains no information about g), and is multiplicative (i.e., the secret can be reconstructed by a multiplicative expression over a non-Abelian group, to which any t parties can apply one of their shares). All 3 properties are proved by induction over the recursion level.

4.2 Performances

In [28], the authors show how to compute and bound the total number of shares.

Definition 3. *Let t, n be the parameter of the threshold secret sharing defined by Figure 3. We define as $S(t, n)$ the total number of shares.*

[12] This point could technically be avoided and considered a consequence of the latter point.

Remark 2. By the recursive nature of the secret sharing scheme, it immediately holds that

$$\begin{aligned}
S(t,n) &= \sum_{i=0}^{t} S\left(i, \left\lfloor \frac{n}{2} \right\rfloor\right) + S\left(t-i, \left\lceil \frac{n}{2} \right\rceil\right), && \text{for } t \in \left[2, \left\lfloor \frac{n}{2} \right\rfloor\right] \\
S(t,n) &= \sum_{i=t-\left\lfloor \frac{n}{2} \right\rfloor}^{\left\lfloor \frac{n}{2} \right\rfloor} S\left(i, \left\lfloor \frac{n}{2} \right\rfloor\right) + S\left(t-i, \left\lceil \frac{n}{2} \right\rceil\right), && \text{for } t \in \left[\left\lfloor \frac{n}{2} \right\rfloor + 1, n-1\right] \\
S(t,n) &= n && \text{for } t = 1 \text{ or } t = n \\
S(t,n) &= 0 && \text{for } t \leq 0 \text{ or } t > n.
\end{aligned}$$

We revisit the analysis of the scheme in [28], looking for an improved upper bound on the number of shares distributed to all participants. Our main result is the following:

Theorem 1. *For any $t, n \in \mathbb{N}$ such that $t \in [2, n-1]$, it holds that $S(t,n) \leq 2n \cdot \min\{ub_0, ub_1, ub_2\}$, where*

$$ub_0 = \left(\frac{e(a+b)}{a}\right)^a, \quad ub_1 = \left(\frac{e(a+b)}{b}\right)^b, \quad ub_2 = \sqrt{\frac{(a+b)}{2\pi \cdot a \cdot b}} \cdot \frac{(a+b)^{a+b}}{a^a b^b},$$

and where $a = \lceil \log n \rceil$, $b = t-1$, and e is Euler's constant (i.e., $e = 2.781...$).

The proof for these bounds is slightly technical and we inserted it in Appendix C. The total number of shares represents also a bound on the number of group operations during the key generation procedure (Figure 3). However, for our scenario, it is also relevant to bound on the number of shares each user receive, lets call it $U(t,n)$. Using the same techniques in the proof of Theorem 1, we can bound $U(t,n)$ using the bounds ub_0, ub_1, ub_2, divided by the number of parties n. We can also recursively compute $U(t,n)$ in this way:

$$\begin{aligned}
U(t,n) &= 1 && \text{if } t = 1 \text{ or } t = n \\
U(t,n) &= \sum_{\ell=\max\left(1, t-\left\lfloor \frac{n}{2} \right\rfloor\right)}^{\min\left(t, \left\lfloor \frac{n}{2} \right\rfloor\right)} U\left(\ell, \left\lceil \frac{n}{2} \right\rceil\right) && \text{otherwise.}
\end{aligned}$$

We remark that the upper bound given in [28] for the total number of shares can be considered an asymptotic-notation version of our upper bound ub_0, focusing on large values of t (e.g., $t = \omega(\log n)$). Thus, our analysis additionally generalizes their bound to any t, and shows that the constant hidden in their asymptotic notation, is small. In Figure 4 we compared the 3 upper bounds for $n = 100$ and all $t \leq n/2$. We conclude that upper bound ub_1 is lower than ub_0 for small values of t, and upper bound ub_2 is always smaller than both, by a factor of about 10 or more, thus positively answering the first open question from [28] (i.e., on whether their upper bound could be improved). Similar numerical considerations were derived up to $n = 10$K. Now, we analyze the scheme's performance in specific scenarios depending on the relative values of t and n.

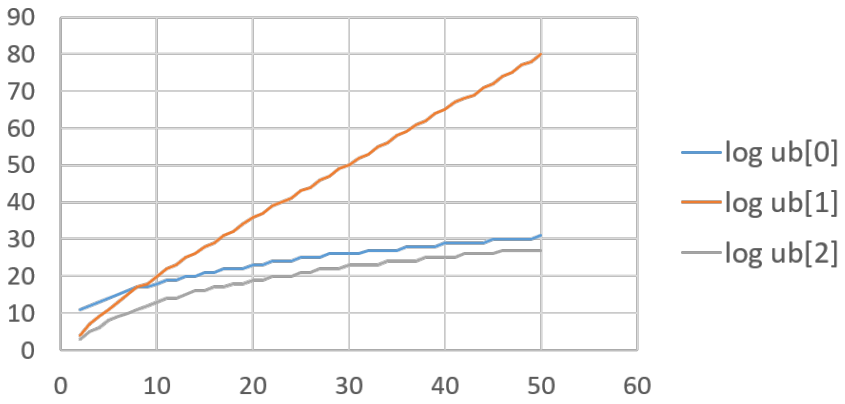


Fig. 4: Comparing (the logarithm of) the 3 upper bounds as a function of $t \leq n/2$, for $n = 100$.

Case t constant (in n). In this case, upper bound ub_1 shows that every participant is given at most a *polylogarithmic* in n (specifically, $O((\log n)^{t-1})$) number of group elements, thus extending the analogue observation done in [28] for $t = 2$, and improving the upper bound in Replicated Secret Sharing [1, 33], which is *polynomial* in n. In Figure 5 (top), we compare exact values of the number of group elements shared among all participants or to any one participant, for both sharing approaches, when $t = 2$.

Case t logarithmic (in n). Here, the upper bound ub_0 shows that every participant is given at most a *polynomial* in n number of group elements. Specifically, when $t = c \log n$, for some constant c, we have that $S(t, n) = O(n^d)$, for a related constant $d = \log(e(c+1))$. This improves the upper bound in Replicated Secret Sharing [1, 33], which is *super-polynomial* in n. In Figure 5 (center), we compare exact values of the number of group elements shared among all participants or to any one participant, for both sharing approaches, when $t = \lfloor \log n \rfloor$.

Case arbitrary t. Here, our upper bounds ub_0 and ub_2 show that participants are given a *slightly super-polynomial* in n number of group elements, thus improving the upper bound in Replicated Secret Sharing [1, 33], which is *exponential* in n. In Figure 5 (bottom), we compare exact values of the number of group elements shared among all participants or to any one participant, for both sharing approaches, when $t = \lfloor n/2 \rfloor$. Sequence Sharing of [16] can be used to produce a multiplicative threshold scheme but would also require an exponential number of group elements to be distributed. The multiplicative threshold scheme in [25] would distribute an (asymptotically) polynomial number of group elements for any t, but the involved constants make it a very impractical scheme. Thus, it still remains of interest to find a multiplicative threshold scheme over arbitrary groups, distributing a practically efficient number of group elements for any t.

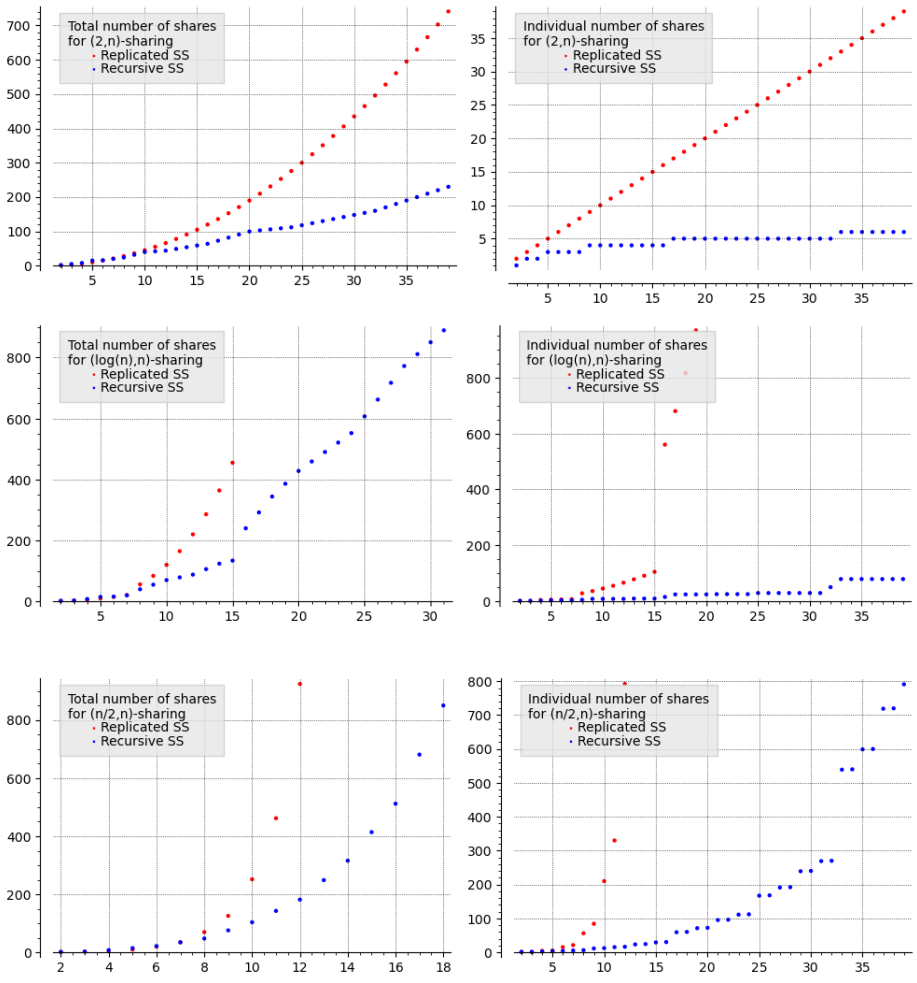


Fig. 5: Total and individual number of shares in the following parameter scenarios: (top) $t = 2$, (center) $t = \lfloor \log(n) \rfloor$, (bottom) $t = \left\lfloor \frac{n}{2} \right\rfloor$, when $n \in [2, 35]$

Comparison with GRASS key generation. This sharing, with respect to the Replicated Secret Sharing [1, 33] used in [6], has the disadvantages of a much more complicated sharing procedure, that was designed with a centralized key generation in mind. In fact, the natural approach to decentralize this key generation procedure requires the participants to agree on $O(S(t,n))$ group values. Moreover, now each share is composed of a group element (even if some of them can be compressed as seed) and a full set element, that for code based group actions are considerably more expensive than single seeds. However, this is balanced by the fact that the number of total shares and individual shares is much lower, as shown by our bounds in Theorem 1 and plots in Figure 5. More importantly, RecKeyGen has the key advantage that the number of interactive rounds necessary for multiparty computation of g now is only t, instead of $\binom{n}{t}$. This can immediately be applied to the GRASS scheme [6], obtaining a scheme with only $2t+1$ interaction rounds between the parties (t for the commitment generation and t for the response), independently of n, greatly reducing the latency of the protocol.

4.3 Simulatability

We investigate simulatability properties of the presented Key Generation scheme, which are important when using it in the context of a threshold signature scheme. This does not follow immediately from [28] since in our protocol the dealer also needs to publish the set elements corresponding to the group elements; however, they can easily be simulated from the shares of the group elements. Also note that the dealer samples new group elements at every recursion, so (with overwhelming probability) no group element is used twice, thus the attacks from [17] cannot be applied.

Proposition 2. *The procedure* RecKeyGen *is simulatable, i.e., given a valid public key* $\mathtt{pk} = (x, y)$*, there exist a simulator* $\mathcal{S}$ *that can simulate the view of* RecKeyGen *applied on* pk *for any corrupted set with cardinality less than the threshold* t*.*

Proof. We prove by induction on n that given x, y set elements in the same orbit we can simulate shards with the same distribution of RecKeyGen (i.e. uniform) in polynomial time without knowledge of a GAIP solution, i.e. g such that $y = g \star x$. For this we exploit the recursive structure of RecKeyGen.

We start the induction from $n = 2$. In $t = 1$ we do not have anything to prove (there are no corrupted parties). If $t = 2$ we sample $g' \in G$ and send it to the corrupted party $P_{\mathtt{cor}}$, then if $P_{\mathtt{cor}}$ is the first user we send x to $P_{\mathtt{cor}}$, otherwise we send $x' = g'^{-1} \star y$. Since the other share is implicitly defined as a product of g and g' (or its inverse), as in RecKeyGen, the shares have the same uniform distribution. It is clear that when $t = n$ (for any n) we can repeat the same simulation strategy with longer chains: let i^* be the honest player, the simulator knows all the g_i for $i \neq i^*$, so it can compute all the set elements for $i \leq i^*$ by computing $x_i = g_i \star x_{i-1}$, with $X_0 = x$ and all the set elements for $i > i^*$ by computing $x_i = g_i^{-1} \star x_{i+1}$, with $X_n = y$.

Now we prove this for n assuming that we have a simulation strategy for any $n' < n$. The cases $t = 1, n$ are already solved above, so let's focus on the case $t \neq 1, n$, thus we need to iterate through ℓ as it is done in line 17 of RecKeyGen (Figure 3). If $\ell = 0, t$ we can just simulate the sharing on a smaller set of users($\mathcal{P}_{\mathsf{left}}$ or $\mathcal{P}_{\mathsf{right}}$) using the simulation strategy that we have by the induction hypothesis since $\lceil \frac{n}{2} \rceil < n$.

Otherwise, we define $\mathtt{cor}_{\mathsf{left}} = \mathtt{cor} \cap \mathcal{P}_{\mathsf{left}}$ and $\mathtt{cor}_{\mathsf{right}} = \mathtt{cor} \cap \mathcal{P}_{\mathsf{right}}$. Now since $\mathtt{cor} < t$ necessarily $\mathtt{cor}_{\mathsf{left}} < t - \ell$ or $\mathtt{cor}_{\mathsf{right}} < \ell$, wlog suppose that $\mathtt{cor}_{\mathsf{right}} < \ell$, the other case follows in the same way. We sample $g_1 \in G$ and define $x_1 = g_1 \star x$. Again, since the other share g_2 is implicitly defined as $g \cdot g_1^{-1}$ all the generated shares are uniformly distributed as in RecKeyGen. Thus, we can simulate the ℓ-out-of-$|\mathcal{P}_{\mathsf{right}}|$ sharing on x_1, y using the simulation strategy, given by the induction hypothesis since $|\mathcal{P}_{\mathsf{right}}| = \lceil \frac{n}{2} \rceil < n$. Then we share g_1, x, x_1 on $\mathcal{P}_{\mathsf{left}}$ using RecKeyGen (here we do not even need to simulate anything since we know g_1).

5 Protocol Description

In this section we present GRASS+, an improved version of GRASS, presented in [6]. The concept of the protocol remains consistent with the original work, while we add ZKP to achieve better security and the key generation of Section 4.

Each signing session is uniquely identified by a session identifier `ssid`. Let $\mathcal{S}_{\mathsf{sig}}$ be the signing set for the session. Wlog we consider $\mathcal{S}_{\mathsf{sig}} = \{1, ..., t\}$. All the parties involved have multiplicative shares of g such that $g_1 \cdots g_t = g$. Moreover, let (x, y) be the public key and $\{x_j\}_{j=0,\ldots,t}$ the intermediate set elements satisfying $x_i = g_i \star x_{i-1}$, with $x_0 = x$ and $x_t = y$. These set elements are known to the parties (the i-th party knows both x_i and x_{i-1}).

The signature protocol can be read in Figure 6. The signing session involves a double round-robin ($\mathsf{TSign}_1, \mathsf{TSign}_2$), each followed by a finalization protocol ($\mathsf{Fin}_1, \mathsf{Fin}_2$). In the Fin_1 protocol, each party i discloses the value r_i committed during TSign_1. These values are subsequently aggregated, and the result is utilized as a salt in the challenge generation. This approach prevents the adversary from predicting the challenge in advance during the simulation, as explained in Remark 3.

In particular, starting from the first party, each party repeat $\mathsf{rps} = \lambda$ times in parallel the following:

- chooses a random group element (line 8),
- apply it to the set element broadcast by the previous party,
- the resulting set element is then broadcast as well as a ZKP about the knowledge of the group action.

The ZKPs need to be online-extractable, for reasons that will be more clear in the proof, as explained in Remark 4. A possible proof Π_{GAK} obtained using the Unruh [43] transform is shown in Appendix B. Unfortunately, this decrease the performance of the protocol, since Unruh transform basically double the size

of the transcript: indeed, since the challenge space is binary, during each round the prover needs to send the response to one of the two challenges (as in normal Fiat-Shamir) and the hash of the other one.

Then, after all the first round-robin is finished, the parties engage in a second round-robin to compute the signature. In particular, when the challenge bit is 0 all the parties reveal the random group element chosen, while if the challenge bit is 1 the parties engage in an iterative protocol as follows:

- each party retrieve the group element from the previous player (the first player starts with the identity element 1),
- each player multiplies it by the nonce chosen in TSign_1 (on the left) and by the inverse of its private key (on the right).

In this way the response of each player is the group element that links the player public key to the commitment sent during TSign_1. If this equation does not hold then the next party aborts, otherwise the protocol continue until the last player, which produce the final signature.

The verification protocol is the same of the centralized signature, that we report for completeness in Figure 7.

6 Adaptive Security

In our protocol we suppose the existence of a broadcast channel, where the messages are published. To avoid the usage of a broadcast channel, when every party sends a message to another one it is required to sign the message. For further information about this technique see [26].

We divide the proof in three steps: first we consider the full threshold case (i.e. all the parties are needed to sign) and we prove the security for $\frac{n-1}{2}$ corruptions (Section 6.1) and $n-1$ corruptions (Section 6.2). Finally, we show how to generalize (parts of) our proofs to any general t-out-of-n threshold.

6.1 $(n-1)/2$ Corruptions

Theorem 2. *Let $(G, X, \star)$ be a group action such that n-Chain-GAIP (Problem 2) is hard, then the digital signature of Figure 6 is secure against adaptive chosen message attack with up to $\frac{n-1}{2}$ corruptions in the Random Oracle Model.*

Outline of the Proof. The proof follows a standard game-based approach. The goal is to show that if an adversary is able to produce a forgery, then it is possible to build an adversary able to win the n-Chain-GAIP problem (Problem 2). To do so we need to simulate the EUF-CMA game of Figure 2. The strategy is very similar to the one used to prove the honest-verifier zero-knowledge of the base centralized protocol (see [5] for an example, where the group action used is the code equivalence):

- every time a new session query is started the simulator choose a random challenge `ch`;

$\mathsf{TSign}_1(\mathcal{S}_{\mathsf{sig}}, \mathtt{mes}, \mathtt{sk}_i, \{\mathtt{pm}_{j,1}\}_{j\in\mathcal{S}_{\mathsf{sig}}, j<i})$

$\tilde{x}_0^k \leftarrow x$ for all $k \in \{1, ..., \mathsf{rps}\}$
if $i \neq 1$
  for $j \in \mathcal{S}_{\mathsf{sig}}, j < i$
    $\{\tilde{x}_j^k, \}_{k=1,...,\mathsf{rps}}, \pi_i \leftarrow \mathtt{pm}_{j,1}[1, 2]$
  if $\mathsf{V}(\pi_i) = \perp$
    return $\perp$
$r_i \xleftarrow{\$} \{0, 1\}^{\mathsf{rps}}$
$\mathtt{cmt}_i \leftarrow \mathsf{H}(\mathtt{ssid}, r_i)$
for $k = 1, ..., \mathsf{rps}$
  $\tilde{g}_i^k \xleftarrow{\$} \mathbb{G}$
  $\tilde{x}_i^k \leftarrow \tilde{g}_i^k \star \tilde{x}_{i-1}^k$
$\pi_i \leftarrow \Pi_{\mathsf{GAK}}(\tilde{g}_i^k, \tilde{x}_i^k, \tilde{x}_{i-1}^k)$
$\mathtt{pm}_{i,1} \leftarrow (\{\tilde{x}_i^k\}_{k=1,...,\mathsf{rps}}, \pi_i, \mathtt{cmt}_i)$
$\mathtt{st}_{i,1} \leftarrow (\{\tilde{g}_i^k\}_{k=1,...,\mathsf{rps}}, r_i)$
return $(\mathtt{pm}_{i,1}, \mathtt{st}_{i,1})$

$\mathsf{TSign}_2(\mathcal{S}_{\mathsf{sig}}, \mathtt{mes}, \mathtt{sk}_i, \{\mathtt{pm}_{j,1}, r_j\}_{j\in\mathcal{S}_{\mathsf{sig}}}, \{\mathtt{pm}_{j,2}\}_{j\in\mathcal{S}_{\mathsf{sig}}, j<i})$

parse $(\{\tilde{g}_i^k\}_{k=1,...,\mathsf{rps}}) \leftarrow \mathtt{st}_{i,1}[1]$
for $j \in \mathcal{S}_{\mathsf{sig}}$
  parse all $(\{\tilde{x}_j^k\}_{k=1,...,\mathsf{rps}}, \pi_j, \mathtt{cmt}_j) \leftarrow \mathtt{pm}_{j,1}$
  if $\mathsf{V}(\pi_j) = \perp$ **or** $\mathtt{cmt}_j \neq \mathsf{H}(\mathtt{ssid}, r_j)$
    return $\perp$
$r \leftarrow \bigoplus_{j\in\mathcal{S}_{\mathsf{sig}}} r_j$
$\mathtt{ch} \leftarrow \mathsf{H}(r||\tilde{x}_t^1||...||\tilde{x}_t^{\mathsf{rps}}||\mathtt{mes})$
for $j \in \mathcal{S}_{\mathsf{sig}}, j < i$
  parse all $(\{\mathtt{rsp}_j^k\}_{k=1,...,\mathsf{rps}}) \leftarrow \mathtt{pm}_{j,2}$
  for $k = 1, ..., \mathsf{rps}$
    if $\mathtt{ch}_k = 0$
      if $\mathtt{rsp}_j^k \star \tilde{x}_{j-1}^k \neq \tilde{x}_j^k$ **return** $\perp$
    if $\mathtt{ch}_k = 1$
      if $\mathtt{rsp}_j^k \star x_j \neq \tilde{x}_j^k$ **return** $\perp$
for $k = 1, ..., \mathsf{rps}$
  if $i = 1$
    $\mathtt{rsp}_0^k = 1$
  if $\mathtt{ch}_k = 0$
    $\mathtt{rsp}_i^k \leftarrow \tilde{g}_i^k$
  if $\mathtt{ch}_k = 1$
    $\mathtt{rsp}_i^k \leftarrow \tilde{g}_i^k \cdot \mathtt{rsp}_{i-1}^k \cdot g_i^{-1}$
$\mathtt{pm}_{i,2} \leftarrow \{\mathtt{rsp}_i^k\}_{k=1,...,\mathsf{rps}}$

$\mathsf{Fin}_1(\mathcal{S}_{\mathsf{sig}}, \mathtt{mes}, \mathtt{sk}_i, \{\mathtt{pm}_{j,1}\}_{j\in\mathcal{S}_{\mathsf{sig}}})$

$r_i \leftarrow \mathtt{st}_{i,1}[2]$
return r_i

$\mathsf{Fin}_2(\mathtt{ch}, \mathcal{S}_{\mathsf{sig}}, \mathtt{mes}, \mathtt{sk}_i, \{\mathtt{pm}_{j,1}, \mathtt{pm}_{j,2}, r_j\}_{j\in\mathcal{S}_{\mathsf{sig}}})$

parse all $\{\mathtt{rsp}_i^k\}_{k=1,...,\mathsf{rps}} \leftarrow \mathtt{pm}_{i,2}$
$r \leftarrow \bigoplus_{j\in\mathcal{S}_{\mathsf{sig}}} r_j$
for $k = 1, ..., \mathsf{rps}$
  if $\mathtt{ch}_k = 0$
    $\mathtt{rsp}^k \leftarrow \prod_{j=1}^{t} \mathtt{rsp}_j^k$
  if $\mathtt{ch} = 1$
    $\mathtt{rsp}^k \leftarrow \mathtt{rsp}_{\max(\mathcal{S}_{\mathsf{sig}})}^k$
$\sigma \leftarrow \mathtt{rsp}^1||...||\mathtt{rsp}^{\mathsf{rps}}||\mathtt{ch}||r$
return σ

Fig. 6: Signature protocol for GRASS+.

$\mathsf{Ver}((x,y),\mathtt{mes},\sigma) \to 0/1$

1 : **parse** $(\{\mathtt{rsp}_k\}_{k=1,\dots,\lambda},\mathtt{ch},r) \leftarrow \sigma$
2 : **for** $k = 1,...,\lambda$
3 : **if** $\mathtt{ch}_k = 0$
4 : $\tilde{x}_t^k \leftarrow \mathtt{rsp}_k \star x$
5 : **if** $\mathtt{ch} = 1$
6 : $\tilde{x}_t^k \leftarrow \mathtt{rsp}_k \star y$
7 : **return** $\mathtt{ch} = \mathsf{H}(r||\tilde{x}_t^1||...||\tilde{x}_t^\lambda||\mathtt{mes})$

Fig. 7: Verification algorithm for GRASS+

– the simulator then simulates the signature generation using the chosen challenge. Notice that a single round is basically an execution of the centralized protocol, so the same strategy can be exploited.

To avoid inconsistencies when answering the random oracle queries, we need that the simulator knows the input in challenge computation before the adversary. This is done by adding a random salt as additional input of the hash function, that the simulator can extract from the adversary.

Lastly, the simulator needs to answer to the corruption queries. Unfortunately the strategy adopted makes impossible to reconstruct all the private data, so the simulator needs some additional information. In particular, the simulator needs to be able to extract all the secret nonces used by the adversary, this is possible thanks to the online-extractable ZKP introduced during the execution of TSign_1.

Proof. Consider a probabilistic polynomial-time adversary $\mathcal{A}$ that make up to $\frac{n-1}{2}$ corruption queries, q_s sign queries and q_h quantum call to the random oracle $\mathcal{O}^{\mathsf{H}}$. We build a probabilistic polynomial-time algorithm $\mathcal{S}$ for n-Chain-GAIP that use $\mathcal{A}$ as a subroutine. In the proof $\mathcal{S}$ runs $\mathcal{A}$ two times. In this way, $\mathcal{S}$ makes no more than $n-1$ queries to its oracle $\mathcal{O}^{\mathsf{chain}}$ and aims to output the n groups elements that constitute the chain. If the total number of queries is less than $n-1$ then $\mathcal{S}$ performs the additional queries required to extract the solution.

The idea of the proof is that during the two iterations, $\mathcal{S}$ reprograms the random oracle $\mathcal{O}^{\mathsf{H}}$ to output a different random value on a single input, so that it can extract the action from x to y.

We now describe how $\mathcal{S}$ can simulate the game $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{euf-cma}}$. In particular $\mathcal{S}$ is responsible for simulating the key generation and the response to the oracles $\mathcal{O}^{\mathsf{Corrupt}}, \mathcal{O}^{\mathsf{TSign}_1}$ and $\mathcal{O}^{\mathsf{TSign}_2}$ as well as all the random oracle queries $\mathcal{O}^{\mathsf{H}}$. $\mathcal{S}$ initializes two empty tables to save all the oracle queries and the sign queries done by the adversary. Let Q_{H} and Q_{TSign} such tables. At the beginning of the second iteration of $\mathcal{A}$, $\mathcal{S}$ resets Q_{TSign} to the empty set, while it keeps Q_{H}. Lastly $\mathcal{S}$ initialize a corruption counter $\mathtt{cc} = 0$ that will count the number of corruption

made by the adversary. If $\mathtt{cc}$ ever surpass $\frac{n-1}{2}$ $\mathcal{S}$ aborts. $\mathcal{S}$ resets such counter to 0 at the beginning of the second iteration.

n-Chain-GAIP challenge. $\mathcal{S}$ takes as input an n-Chain-GAIP challenge in the form of $\{x_i\}_{i=0,\dots,n}$. $\mathcal{S}$ has also access to an oracle $\mathcal{O}^{\mathsf{chain}}$ that can answer to up to $n-1$ queries, $\mathcal{S}$ will use it to answer to the corruption queries made by the adversary. $\mathcal{S}$ sets $(x, y) = (x_0, x_n)$ as the public key of the $\mathsf{Exp}_{\mathcal{A}}^{\mathsf{euf-cma}}$ game, with (x_{i-1}, x_i) being the public key of party i and g_i be the corresponding private key.

Simulating the RO Queries. When $\mathcal{A}$ queries the random oracle $\mathcal{O}^{\mathsf{H}}$ on input X, $\mathcal{S}$ checks whether $X \in Q_{\mathsf{H}}$. If this is the case then $\mathcal{S}$ answers with the same output, otherwise $\mathcal{S}$ samples $c \xleftarrow{\$} \{0,1\}^{\mathsf{rps}}$ randomly, saves (X, c) in the table Q_{H} and returns c.

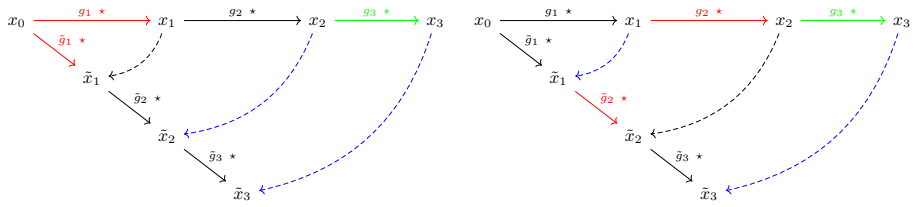
Simulating Signature. $\mathcal{S}$ needs to simulate both $\mathcal{O}^{\mathsf{TSign}_1}$ and $\mathcal{O}^{\mathsf{TSign}_2}$. The simulation proceeds as follows:

- $\mathcal{O}^{\mathsf{TSign}_1}()$. $\mathcal{S}$ do the checks in at the beginning of the game of Figure 2. In particular $\mathcal{S}$ checks whether i is an honest party and whether the session id $\mathtt{ssid}$ is correctly defined (in the case $i = 0$, then $\mathcal{S}$ choose a random session id $\mathtt{ssid}$) or not. Lastly, $\mathcal{S}$ checks whether all the previous player correctly sent their data and the corresponding ZKP. During this step $\mathcal{S}$ can also extract all the previous $\tilde{g}_j$ from the adversary. If all the checks are passed then $\mathcal{S}$ checks whether a challenge $\mathtt{ch}^{\mathtt{ssid}}$ is defined or not. In case it is not defined (meaning that this is the first time $\mathcal{S}$ participate in the signing session $\mathtt{ssid}$), $\mathcal{S}$ choose a random string and sets it as $\mathtt{ch}^{\mathtt{ssid}}$. See Remark 3 for more details. For $k = 1, ..., \mathsf{rps}$ does the following:
  - if $\mathtt{ch}_i^{\mathtt{ssid}} = 0$ then follows the protocol normally,
  - if $\mathtt{ch}_i^{\mathtt{ssid}} = 1$ then $\mathcal{S}$ choose a random $\tilde{g}_i^k \in \mathbb{G}$ and sets $\tilde{x}_i^k = \tilde{g}_i^k \star x_{i-1}$. Then it simulates the ZKPs necessary and compute π_i.
- $\mathcal{O}^{\mathsf{TSign}_2}()$: by construction, the challenge is $\mathtt{ch} = \mathtt{ch}^{\mathtt{ssid}}$, except with negligible probability (see Remark 3). Thanks to the value chosen in the simulation of $\mathcal{O}^{\mathsf{TSign}_1}()$ $\mathcal{S}$ can successfully answer both challenges.

Corruption Query. $\mathcal{S}$ needs to answer the corruption queries made by the adversary. When $\mathcal{A}$ asks for the corruption of party i, $\mathcal{S}$ checks that i is an honest party and if there are less than $\frac{n-1}{2}$ corrupted parties. If both checks are true then $\mathcal{S}$ queries $\mathcal{O}^{\mathsf{chain}}$ on input (x_{i-1}, x_i) to get g_i.

At this point $\mathcal{S}_i$ can successfully retrieve all the $\tilde{g}_i^k$ used in all the active signing session involving i, filling all the $\mathtt{st}_{i,1}$ (excluding those for which i has not yet sent a message). In particular $\mathcal{S}$ needs to reconstruct $\tilde{g}_i^k$ when the challenge is 1, since $\mathcal{S}$ follows the protocol honestly when the challenge is 0. For every session, let $i^* < i$ be the last honest player, controlled by $\mathcal{S}$. In the case where all the player before i are corrupted or i is the first player we set $i^* = 0$. Clearly $\mathcal{S}$ knows all the g_j with $i^* < j < i$, as well as g_i since they are all corrupted

parties. Moreover, $\mathcal{S}$ extracted all the $\tilde{g}_j^k$ from the ZKPs (lines 5 and 10 of TSign_1). Thus, $\mathcal{S}$ knows both the action from x_{i^*} to x_i and the action from $\tilde{x}_{i^*}^k$ to $\tilde{x}_{i-1}^k$. Lastly $\mathcal{S}$ knows both the action from x_i to $\tilde{x}_i^k$ and the action from x_{i^*} to $\tilde{x}_{i^*}^k$ by construction (see the simulation of TSign_1 when the challenge bit is 1). In the case $i^* = 0$ $\mathcal{S}$ can simply set $\tilde{x}_{i^*}^k = x_0$ for all k and thus the action is the identity element. In this way $\mathcal{S}$ can compute the actions from $\tilde{x}_{i-1}^k$ to $\tilde{x}_i^k$.



(a) The previous player is honest. (b) The previous player is corrupted.

Fig. 8: Simulation of a corruption query on player P_2 (in green).

Figure 8a and Figure 8b show schematically how the corruption queries (on P_2, in green) are simulated, depending on whether the previous player is corrupted (in red) or not.

- Figure 8a since P_1 is honest, the simulator knows both the blued arrows and the green arrow (thanks to the $\mathcal{O}^{\mathsf{Corrupt}}$ made to the n-Chain-GAIP oracle). Thus, it can compute $\tilde{g}_3$ by composing the three actions.
- Figure 8b since P_2 is corrupted, the simulator cannot do the same computation of the previous case. However, the simulator knows both the blued arrows, the green arrow (thanks to the $\mathcal{O}^{\mathsf{Corrupt}}$ made to the n-Chain-GAIP oracle) and both the red arrows (thanks to the online extractable ZKP and a previous corruption query). Hence, it can compute $\tilde{g}_3$ by composing the five actions.

Solving n-Chain-GAIP. After the simulation the adversary outputs a forgery. At this point the simulator rewind and change the challenge corresponding to the forgery made by the adversary, thus extracting the action form x to y. $\mathcal{S}$ the use the same action to win the n-Chain-GAIP game.

Remark 3. The simulator chooses the challenge for each session after receiving a query for it for the first time. Then, for all the session, the simulator continues to use the same challenge chosen. To reprogram the random oracle in such a way $\mathtt{ch}^{\mathtt{ssid}}$ is as expected the Simulator just need to collect the set elements $\tilde{x}_t^k$ for $k = 1, ..., \mathsf{rps}$, sent after the last execution of TSign_1, and all the random salt values r_i. The adversary need to commit on these salt values, so the simulator can extract them from the random oracle queries, but they are all revealed only

in Fin_1 after the $\tilde{x}_t^k$ are produced. Thus, the simulator learns them ahead of time and can reprogram the random oracle accordingly.

The only way that the simulation fails is if a query with the same input has already been sent to the random oracle, however this happens with negligible probability since at least one of the salt values is chosen uniformly at random.

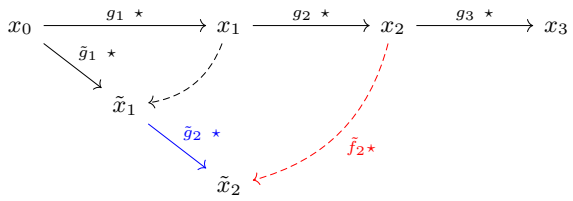


Fig. 9: The simulation failure without the ZKP of TSign_1. If the simulator computes $\tilde{x}_2$ using the red arrow and then the adversary, that controls P_1, corrupts P_2, the simulator is not able to retrieve the blue arrow.

Remark 4. The main difference from the original static secure signature [6] is the non-interactive ZKP in TSign_1. Unfortunately, this is necessary to allow the simulator to correctly answer to the corruption queries. Indeed, let us consider a $(3, 3)$ signature where the adversary controls P_1, while P_2, P_3 are honest (see Figure 9). Notice that since the threshold is 3, the adversary can perform a second corruption query at any time. Moreover, suppose that the challenge is 1. According to the simulator strategy, the simulator compute $\tilde{x}_2$ starting from x_2 (red arrow) instead of $\tilde{x}_1$ (blue arrow).

Now, suppose that after receiving $\tilde{x}_2$ the adversary perform a corruption query on P_2. The simulator then get g_2 from the n-Chain-GAIP oracle, however it is unable to retrieve $\tilde{g}_2$, and thus the simulation fails. Notice that the simulator knows $\tilde{f}_2$ by construction and g_1, g_2 thanks to the n-Chain-GAIP oracle. In this way, the oracle can compute $\tilde{g}_1 \cdot \tilde{g}_2$, however it cannot compute neither $\tilde{g}_1$ nor $\tilde{g}_2$, otherwise it would be able to break the GAIP problem.

6.2 $n - 1$ Corruptions

Theorem 3. *Let $(G, X, \star)$ be a group action such that n-Chain-GAIP (Problem 2) is hard. Then the digital signature of Figure 6, with $\mathsf{rps} = \lambda + \log(n)$ is secure against adaptive chosen message attack with up to $n - 1$ corruptions in the Random Oracle and Black Box Group Action Model.*

Proof. The simulation strategy is the same as the previous proofs, the only difference is that the simulator needs to extract the solution for the n-Chain-GAIP after a single forgery.

For this proof we use the same notation of the previous proof, so $x_0 = x$, $x_n = y$, $x_1, ..., x_{n-1}$ are the intermediate set elements and $g_0, .., g_{n-1}$ the secret group elements from n-Chain-GAIP. Let $\mathtt{mes}, \sigma$ the forgery sent by the adversary, from which we can recover the commitment set elements $\tilde{x}^1, ..., \tilde{x}^{\mathsf{rps}}$, the salt r and the responses $\mathtt{resp} = (\mathtt{resp}^1, ..., \mathtt{resp}^{\mathsf{rps}})$. Recall that $\mathtt{ch} \leftarrow \mathsf{H}(r||\tilde{x}_t^1||...||\tilde{x}_t^{\mathsf{rps}}||\mathtt{mes})$. We summarise the input of the random oracle as

$$\mathtt{cmt} = (r, \tilde{x}^1, ..., \tilde{x}^{\mathsf{rps}}, \mathtt{mes}) \ . \tag{2}$$

Since we are in the BBGAM for each $\tilde{x}^i$ we can "follow" the adversary queries and the computations done by the simulator to recursively find a group element $\bar{g}$ and an index j such that $\tilde{x}^i = \bar{g} \star x_j$, i.e. to find a link to one of the known elements $x_0, x_1, ..., x_n = y$, let $\mathsf{idx}(\tilde{x}^i) := j$. Observe that the validity of the transcript implies that the commitment elements $\tilde{x}^i$ are all part of the same orbit $\mathcal{O}(x) \ni x, y$. Thus, it is not possible for the adversary to link one of the commitment elements to a random set element got using $\mathcal{O}^{\mathsf{sample}}$ since the probability that a random set element lies in the same orbit of a given element is negligible (as discussed in Section 2.4).

Let IDX be $(\mathsf{idx}(\tilde{x}^1), ..., \mathsf{idx}(\tilde{x}^{\mathsf{rps}}))$. Thanks to our simulation strategy ch has been sampled uniformly at random and its distribution is independent of the one the adversary used to generate cmt. In fact, thanks to the salt, $\mathcal{A}$ get access to it only *after* querying the random oracle on cmt.

Using this independence we want to show that we extract a solution to n-Chain-GAIP with non-negligible probability. Now, for any $\ell \in \{0, ..., n-1\}$ we can define the vector $\mathsf{split}_\ell(\mathsf{IDX})$ such that its i-th entry is 1 if $\mathsf{idx}(\tilde{x}^i) > \ell$ and 0 otherwise. Now consider the union set of these n vectors derived from :

$$\mathsf{S}(\mathsf{IDX}) = \{\mathsf{split}_\ell(\mathsf{IDX}) \ \text{ for } \ell = 0, .., n-1\} \ .$$

This set is a deterministic function of cmt, thus its distribution is independent of ch, thus

$$\Pr[\mathtt{ch} \in \mathsf{S}(\mathsf{IDX})] = \frac{n}{2^{\mathsf{rps}}} = \frac{n}{2^{\lambda + \log(n)}} = 2^{-\lambda} \ ,$$

i.e. with overwhelming probability $\mathsf{split}_\ell(\mathsf{IDX}) \neq \mathtt{ch}$ for all ℓ.

We can finally use the forgery information to solve the n-Chain-GAIP. To perform the simulation we have queried $n-1$ solutions g_i to the n-Chain-GAIP oracle (if less we perform the missing one at random), define ℓ such that the only missing pair is $(x_\ell, x_{\ell+1})$. We show now how to extract the missing secret g_ℓ.

By the previous discussion $\mathsf{split}_\ell(\mathsf{IDX}) \neq \mathtt{ch}$. Let i be one of the indexes where they differ.

Consider the case in which $\mathtt{ch}_i = 1$. Thus, $\tilde{x}^i$ is linked to both y (thanks to resp) and x_j, with $j = \mathsf{idx}(\tilde{x}^i)$ (thanks to the BBGAM). We can combine this information to get a group element g' such that $y = g' \star x_j$. Also, $\mathtt{ch}_i = 1$ implies that the i-th entry of split_ℓ is 0, i.e. that $j = \mathsf{idx}(\tilde{x}^i) \leq \ell$. Thus, by previous queries to n-Chain-GAIP oracle, we known $g_0, ..., g_{j-1}$ such that $x_j = (g_{j-1} \cdots g_0) \star x_0$ and we can solve n-Chain GAIP since

$$y = g \star x_0 \text{ with } g = g'(g_{j-1} \cdots g_0)^{-1} \ .$$

If $\mathtt{ch}_i = 0$ we can adapt the same strategy using instead that $\tilde{x}^i$ is linked to both x and x_j for $j > \ell$. Thus, with probability $1 - \frac{1}{2^\lambda}$ we can solve the n-Chain GAIP problem, without rewinding.

6.3 Adapting the Proof for any Threshold

Replicated Secret Sharing. It is easy to see that the above technique can be easily adapted to any threshold t when the key generation is done via replicated secret sharing, as in the original work [6].

Proof (Sketch). Indeed, it is enough to consider a $\binom{n}{t}$-Chain-GAIP instance having the total number of elements equal to the total number of shards. Then, the simulator can simulate the key distribution by sharing the $\{x_i\}$ among all the parties, according to the replicated secret sharing algorithm (notice that it is not important to know all the action g_i during this step). Then the simulation can be carried out as before, except that when the adversary asks for a corruption query the oracle needs to retrieve all the secret key, by doing multiple query to the oracle. This is not a problem since by doing at most $t - 1$ corruptions the simulator will not ask all the group elements to the oracle.

While the above proof sketch is enough to prove the adaptive security of GRASS against $t - 1$ corruptions, there is a performance issue: indeed using replicated secret sharing implies that the total number of shares is $\binom{n}{t}$, and thus the Chain-GAIP challenge should have $\binom{n}{t}$ actions. It is immediate to see from the proof to Theorem 3, that the security loss is thus $\binom{n}{t}$. This is exactly the same security loss that can be achieved by guessing the corrupted set of parties and abort if the guess is incorrect. However, it is important to notice that, while the "naive" approach inherently incurs in such a security loss, our proof is instead tight with respect to the new Chain-GAIP problem. Thus, if the total number of shares is reduced using a better algorithm, then the security loss can be reduced as well, leading to a tighter security compared to simply guessing the adversary corruptions.

Lastly, note that this performance issue impacts also other aspects of the protocol, thus we need anyway $\binom{n}{t}$ to be small.

New Secret Sharing. Given the considerations of the previous paragraph, the first idea to solve the issue would be to use the newly introduced secret sharing of Section 4. However, this is not as straightforward as it could seem. Indeed, the secret sharing of Section 4 does not output a single chain of group actions, but many of them, with many intersections. Thus, we cannot reduce the security of the signature to the n-Chain-GAIP. In particular, to prove the security with the improved secret sharing, we reduce it to N-Graph-GAIP, introduced in Problem 3.

Theorem 4. *Let $(G, X, \star)$ be a group action such that N-Graph-GAIP (Problem 3) is hard. Then the digital signature of Figure 6 with the secret sharing of Section 4 is secure against adaptive chosen message attack with up to $\frac{n-1}{2}$ corruptions in the Random Oracle and Black Box Group Action Model.*

Proof. The proof is the same as the full threshold case (Theorem 2), where instead of receiving an n-Chain-GAIP challenge, the simulator receives as input an N-Graph-GAIP (Problem 3) challenge. Then the simulator set the public key according to it, distributing all the shards to the users following the secret sharing of Section 4 and start the interaction with the adversary. The number of set elements N involved in N-Graph-GAIP is set in such a way to have enough shares for the sharing in Section 4, thus it is bounded by the number of shards $S(t, n)$.

Every time the adversary asks for the corruption of party P_i, the simulator retrieves all the public key of P_i and queries the oracle on them. Notice that since the adversary performs at most t corruptions, then the simulator never asks for an invalid query (i.e., a query that links x, y) by construction, so the simulator can answer all the queries correctly.

The remaining part of the simulation is the same, since the signature algorithm is equal to the full threshold case.

7 Conclusion

In this paper we improved the GRASS signature scheme [6] in two ways: our main result is a post-quantum threshold signature scheme secure against adaptive attacks. This schemes can be instantiated with a variety of cryptographic group actions, in particular [5, 7, 11, 22]. Additionally, as a second result, we were able to improve the key generation protocol.

Even if the asymptotic complexity remains the same, concrete results show a considerable reduction in the number of shares that each party needs to store, allowing a wider choice for the parameters. Improving the secret sharing, possibly reaching a constant number of shares, is still an open problem. In this context, a possible research area is to adapt classical techniques from MPC, like Beaver Triples, to achieve better performance. Another open problem is to define an efficient decentralized version of the new Key Generation procedure.

It is also important to note that improving the secret sharing and the key generation algorithm does not impact only the performance, but also the security of the scheme. As discussed in Section 6, the security loss is often proportional to the number of shares used, so a more efficient protocol would lead to a tighter security reduction.

Unfortunately, our protocol still suffers the same performance issues of the original GRASS and other group-based threshold signature schemes like [24], since we require a double round-robin[13], moreover we introduced expensive ZKPs that are necessary for achieving adaptive security. Trying to reduce the computational overhead while still reaching the higher level of security is a very important challenge to tackle, and possible improvements can be obtained both by compressing the ZKPs (since the Unruh transform is very expensive) or by finding a way to reduce the number of ZKPs needed.

[13] It is worth noting that the usage of one round-robin is optimal, as shown in [23].

Acknowledgements

This collaboration was initiated during the "Post-Quantum Group-Based Cryptography" workshop at the American Institute of Mathematics (AIM), April 29 - May 3, 2024.

This work has been partially supported by project SERICS (PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union - NextGenerationEU and by the European Union FSE-REACT-EU, PON Research and Innovation 2014-2020 DM1062/2021.

The first author is supported by the Italian Ministry of University's PRIN 2022 program under the "Mathematical Primitives for Post Quantum Digital Signatures" (P2022J4HRR) and "POst quantum Identification and eNcryption primiTives: dEsign and Realization (POINTER)" (2022M2JLF2) projects funded by the European Union - Next Generation EU. The second author is supported by SNSF Consolidator Grant CryptonIs 213766. The third author's work was supported the Defense Advanced Research Projects Agency (DARPA), contract number HR001120C0156. The fourth author acknowledges support from Ripple's University Blockchain Research Initiative.

References

[1] Alessandro Baccarini, Marina Blanton, and Chen Yuan. "Multi-party replicated secret sharing over a ring with applications to privacy-preserving machine learning". In: *Cryptology ePrint Archive* (2020).

[2] Renas Bacho et al. *HARTS: High-Threshold, Adaptively Secure, and Robust Threshold Schnorr Signatures*. Cryptology ePrint Archive, Paper 2024/280. 2024. URL: https://eprint.iacr.org/2024/280.

[3] Renas Bacho et al. "Twinkle: Threshold Signatures from DDH with Full Adaptive Security". In: *Advances in Cryptology – EUROCRYPT 2024*. Ed. by Marc Joye and Gregor Leander. Cham: Springer Nature Switzerland, 2024, pp. 429–459. ISBN: 978-3-031-58716-0.

[4] Magali Bardet, Ayoub Otmani, and Mohamed Saeed-Taha. "Permutation Code Equivalence is Not Harder Than Graph Isomorphism When Hulls Are Trivial". In: *2019 IEEE International Symposium on Information Theory (ISIT)*. Paris, France: IEEE Press, 2019, 2464–2468. DOI: 10.1109/ISIT.2019.8849855. URL: https://doi.org/10.1109/ISIT.2019.8849855.

[5] Alessandro Barenghi et al. "LESS-FM: fine-tuning signatures from the code equivalence problem". In: *Post-Quantum Cryptography: 12th International Workshop, PQCrypto 2021, Daejeon, South Korea, July 20–22, 2021, Proceedings 12*. Springer. 2021, pp. 23–43.

[6] Michele Battagliola et al. "Cutting the GRASS: Threshold GRoup Action Signature Schemes". In: *Topics in Cryptology – CT-RSA 2024: Cryptographers' Track at the RSA Conference 2024, San Francisco, CA, USA, May 6–9, 2024, Proceedings*. Berlin, Heidelberg: Springer-Verlag, 2024, 460–489. ISBN: 978-3-031-58867-9. DOI: 10.1007/978-3-031-58868-6_18. URL: https://doi.org/10.1007/978-3-031-58868-6_18.

[7] Ward Beullens, Thorsten Kleinjung, and Frederik Vercauteren. "CSI-FiSh: efficient isogeny based signatures through class group computations". In: *ASIACRYPT 2019*. Springer. 2019.

[8] Ward Beullens et al. "CSI-RAShi: distributed key generation for CSIDH". In: *International Conference on Post-Quantum Cryptography*. Springer. 2021, pp. 257–276.

[9] Jean-François Biasse et al. "LESS is More: Code-Based Signatures Without Syndromes". In: *AFRICACRYPT 2020*. Springer International Publishing, 2020.

[10] Jean-François Biasse and Giacomo Micheli. "A Search-to-Decision Reduction for the Permutation Code Equivalence Problem". In: *2023 IEEE International Symposium on Information Theory (ISIT)*. 2023, pp. 602–607. DOI: 10.1109/ISIT54713.2023.10206940.

[11] Markus Bläser et al. *The ALTEQ signature scheme: Algorithm specifications and supporting documentation*. 2023. URL: https://pqcalteq.github.io/ALTEQ_spec_2023.09.18.pdf.

[12] Dan Boneh, Jiaxin Guan, and Mark Zhandry. "A Lower Bound on the Length of Signatures Based on Group Actions and Generic Isogenies". In: *Annual International Conference on the Theory and Applications of Cryptographic Techniques*. Springer. 2023, pp. 507–531.

[13] Jeremy Booher et al. "Failing to hash into supersingular isogeny graphs". In: *The Computer Journal* (2024), bxae038.

[14] Giacomo Borin et al. *A Guide to the Design of Digital Signatures based on Cryptographic Group Actions*. 2023.

[15] L Brandao and Rene Peralta. *Nist first call for multi-party threshold schemes*. 2023. DOI: https://doi.org/10.6028/NIST.IR.8214C.ipd.

[16] Ernest F. Brickell, Giovanni Di Crescenzo, and Yair Frankel. "Sharing Block Ciphers". In: *Information Security and Privacy, 5th Australasian Conference, ACISP 2000, Brisbane, Australia, July 10-12, 2000, Proceedings*. Ed. by Ed Dawson, Andrew J. Clark, and Colin Boyd. Vol. 1841. Lecture Notes in Computer Science. Springer, 2000, pp. 457–470. DOI: 10.1007/10718964_37. URL: https://doi.org/10.1007/10718964_37.

[17] Alessandro Budroni et al. *Don't Use It Twice! Solving Relaxed Linear Code Equivalence Problems*. Cryptology ePrint Archive, Paper 2024/244. 2024. URL: https://eprint.iacr.org/2024/244.

[18] Fabio Campos and Philipp Muth. "On actively secure fine-grained access structures from isogeny assumptions". In: *International Conference on Post-Quantum Cryptography*. Springer. 2022, pp. 375–398.

[19] Ran Canetti et al. "Adaptive Security for Threshold Cryptosystems". In: *Advances in Cryptology — CRYPTO' 99*. Ed. by Michael Wiener. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 98–116. ISBN: 978-3-540-48405-9.

[20] Ran Canetti et al. "Adaptively secure multi-party computation". In: *Proceedings of the Twenty-Eighth Annual ACM Symposium on Theory of Computing*. STOC '96. Philadelphia, Pennsylvania, USA: Association for

Computing Machinery, 1996, 639–648. ISBN: 0897917855. DOI: 10.1145/237814.238015. URL: https://doi.org/10.1145/237814.238015.

[21] Wouter Castryck et al. "CSIDH: an efficient post-quantum commutative group action". In: *ASIACRYPT 2018*. Springer.

[22] Tung Chou et al. "Take your meds: Digital signatures from matrix code equivalence". In: *International Conference on Cryptology in Africa*. Springer. 2023.

[23] Daniele Cozzo and Emanuele Giunta. "Round-Robin is Optimal: Lower Bounds for Group Action Based Protocols". In: *Theory of Cryptography*. Ed. by Guy Rothblum and Hoeteck Wee. Cham: Springer Nature Switzerland, 2023, pp. 310–335. ISBN: 978-3-031-48624-1.

[24] Daniele Cozzo and Nigel P. Smart. "Sashimi: Cutting up CSI-FiSh Secret Keys to Produce an Actively Secure Distributed Signing Protocol". In: *Post-Quantum Cryptography*. Ed. by Jintai Ding and Jean-Pierre Tillich. Cham: Springer International Publishing, 2020, pp. 169–186. ISBN: 978-3-030-44223-1.

[25] Giovanni Di Crescenzo and Yair Frankel. "Existence of Multiplicative Secret Sharing Schemes with Polynomial Share Expansion". In: *Proceedings of the Tenth Annual ACM-SIAM Symposium on Discrete Algorithms, 17-19 January 1999, Baltimore, Maryland, USA*. Ed. by Robert Endre Tarjan and Tandy J. Warnow. ACM/SIAM, 1999, pp. 895–896. URL: http://dl.acm.org/citation.cfm?id=314500.315074.

[26] Elizabeth Crites, Chelsea Komlo, and Mary Maller. "Fully Adaptive Schnorr Threshold Signatures". In: *Advances in Cryptology – CRYPTO 2023: 43rd Annual International Cryptology Conference, CRYPTO 2023, Santa Barbara, CA, USA, August 20–24, 2023, Proceedings, Part I*. Santa Barbara, CA, USA: Springer-Verlag, 2023, 678–709. ISBN: 978-3-031-38556-8. DOI: 10.1007/978-3-031-38557-5_22. URL: https://doi.org/10.1007/978-3-031-38557-5_22.

[27] Luca De Feo and Michael Meyer. "Threshold schemes from isogeny assumptions". In: *PKC 2020*. Springer. 2020.

[28] Yvo Desmedt, Giovanni Di Crescenzo, and Mike Burmester. "Multiplicative non-abelian sharing schemes and their application to threshold cryptography". In: *Advances in Cryptology—ASIACRYPT'94: 4th International Conferences on the Theory and Applications of Cryptology Wollongong, Australia, November 28–December 1, 1994 Proceedings 4*. Springer. 1995, pp. 19–32.

[29] Julien Duman et al. "Generic models for group actions". In: *IACR International Conference on Public-Key Cryptography*. Springer. 2023, pp. 406–435.

[30] Thomas Espitau, Shuichi Katsumata, and Kaoru Takemure. "Two-round threshold signature from algebraic one-more learning with errors". In: *Annual International Cryptology Conference*. Springer. 2024, pp. 387–424.

[31] Marc Fischlin. "Communication-Efficient Non-interactive Proofs of Knowledge with Online Extractors". In: *Advances in Cryptology – CRYPTO*

2005. Ed. by Victor Shoup. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 152–168. ISBN: 978-3-540-31870-5.

[32] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics: A Foundation for Computer Science, 2nd Ed.* Addison-Wesley, 1994. ISBN: 0-201-55802-5. URL: https://www-cs-faculty.stanford.edu/\%7Eknuth/gkp.html.

[33] Mitsuru Ito, Akira Saito, and Takao Nishizeki. "Secret sharing scheme realizing general access structure". In: *Electronics and Communications in Japan* (1989).

[34] Stanisław Jarecki and Anna Lysyanskaya. "Adaptively Secure Threshold Cryptography: Introducing Concurrency, Removing Erasures". In: *Advances in Cryptology — EUROCRYPT 2000*. Ed. by Bart Preneel. Berlin, Heidelberg: Springer Berlin Heidelberg, 2000, pp. 221–242. ISBN: 978-3-540-45539-4.

[35] Shuichi Katsumata, Michael Reichle, and Kaoru Takemure. "Adaptively secure 5 round threshold signatures from MLWE/MSIS and DL with rewinding". In: *Annual International Cryptology Conference*. Springer. 2024, pp. 459–491.

[36] Irakliy Khaburzaniya et al. "Aggregating and Thresholdizing Hash-based Signatures using STARKs". In: *Proceedings of the 2022 ACM on Asia Conference on Computer and Communications Security*. ASIA CCS '22. Nagasaki, Japan: Association for Computing Machinery, 2022, 393–407. ISBN: 9781450391405. DOI: 10.1145/3488932.3524128. URL: https://doi.org/10.1145/3488932.3524128.

[37] Jonas Nick, Tim Ruffing, and Yannick Seurin. "MuSig2: Simple Two-Round Schnorr Multi-signatures". In: *Advances in Cryptology – CRYPTO 2021*. Ed. by Tal Malkin and Chris Peikert. Cham: Springer International Publishing, 2021, pp. 189–221. ISBN: 978-3-030-84242-0.

[38] NIST. *Call for Additional Digital Signature Schemes for the Post-Quantum Cryptography Standardization Process*. URL: https://csrc.nist.gov/projects/pqc-dig-sig/standardization/call-for-proposals. 2023.

[39] NIST. *Post-Quantum Cryptography Standardization*. URL: https://csrc.nist.gov/Projects/Post-Quantum-Cryptography. 2017.

[40] Emmanuela Orsini and Riccardo Zanotto. "Simple Two-Message OT in the Explicit Isogeny Model". In: *IACR Communications in Cryptology* 1.1 (Apr. 9, 2024). ISSN: 3006-5496. DOI: 10.62056/a39qgy4e-.

[41] Rafael del Pino et al. "Threshold Raccoon: Practical Threshold Signatures from Standard Lattice Assumptions". In: *Advances in Cryptology – EUROCRYPT 2024*. Ed. by Marc Joye and Gregor Leander. Cham: Springer Nature Switzerland, 2024, pp. 219–248. ISBN: 978-3-031-58723-8.

[42] Herbert E. Robbins. "A Remark on Stirling's Formula". In: *American Mathematical Monthly* 62 (1955), pp. 26–29. URL: https://api.semanticscholar.org/CorpusID:122180319.

[43] Dominique Unruh. "Non-Interactive Zero-Knowledge Proofs in the Quantum Random Oracle Model". In: *EUROCRYPT (2)*. Springer, 2015, pp. 755–

784. DOI: 10.1007/978-3-662-46803-6_25. URL: https://www.iacr.org/archive/eurocrypt2015/90560105/90560105.pdf.

A Secret Sharing

We briefly recall the definition and main properties of the type of secret sharing scheme that is sufficient for this paper's needs.

Let $\mathcal{P} = \{P_1, \ldots, P_n\}$ denote a set of n participants. The (t, n)-*threshold access structure* over $\mathcal{P}$ is the set of subsets of $\mathcal{P}$ of size at least t. A *secret sharing scheme for the* (t, n)*-threshold access structure* is a pair of efficient algorithms (Share, Recover) with the following syntax and requirements. On input a value s, called the *secret*, algorithm Share returns n (sets of) values $sh_1, \ldots, sh_n$, called *shares*. On input a subset of shares, algorithm Recover returns a value in s'. The *correctness* requirement says that if an authorized subset of shares (i.e., at least t out of n) is input to Recover, then $s' = s$. The *secrecy* requirement says that the distribution of the secret is independent on the distribution of any unauthorized subset of shares (i.e., at most $t - 1$ out of n).

A secret sharing scheme for the (t, n)-threshold access structure with values in a group G is *multiplicative* (see, e.g., Definition 2 in [28]) if in the reconstruction phase the secret can be written as the group product of t values in G, each of these being locally computed by a different participant, as a function of the share obtained at the end of the distribution phase. A major research question in the area of secret sharing schemes is minimizing the size of the shares.

B Online Extractable Proofs

In this section we provide the online-extractable ZKP used in the signature protocol. Our protocol is essentially the Unruh transform [43] of the standard proof for group action knowledge. Other alternatives are possible, such as using the Fischilin transform [31].

Let g be the secret group action and x, y public set elements, with $y = g \star x$. Let $\mathsf{H}, \mathsf{H}_{\mathtt{ch}}$ two hash functions, with $\mathsf{H}_{\mathtt{ch}} : \{0,1\}^* \to \{0,1\}^\lambda$. Figure 10 shows how the prover, who knows g, can convince the verifier about it.

The verify procedure V consist in computing all the $\tilde{x}_i$ and $\mathtt{cmt}_{i,j}$ for $j = 0, 1$ $i = 1, ..., \lambda$ and then checking the correctness by computing $\mathtt{ch}$. In particular, for each challenge bit $\mathtt{ch}_i$ the verifier compute the following

- if $\mathtt{ch}_i = 0$ then $\tilde{x}_i = \mathtt{rsp}_{i,0} \star x$ and $\mathtt{cmt}_{i,0} = \mathsf{H}(\mathtt{rsp}_{i,0})$,
- if $\mathtt{ch}_i = 1$ then $\tilde{x}_i = \mathtt{rsp}_{i,1} \star y$ and $\mathtt{cmt}_{i,0} = \mathsf{H}(\mathtt{rsp}_{i,1})$,

then check whether $\mathtt{ch} = \mathsf{H}_{\mathtt{ch}}(\tilde{x}_1||\mathtt{cmt}_{1,0}||\mathtt{cmt}_{1,1}||...||\tilde{x}_\lambda||\mathtt{cmt}_{\lambda,0}||\mathtt{cmt}_{\lambda,1})$ or not.

The correctness, honest-verifier zero-knowledge and online-extractability properties follow directly from the security of the base protocol and the properties of the Unruh transform. The protocol efficiency can be improved by using well-known optimizations like seed trees and fixed-weight challenges (see e.g. a survey in [14]) and by storing the commitments $\mathtt{cmt}_{i,j}$ in more efficient structures, like Merkle Trees. In the interest of clarity, we omit these optimizations from the description.

$\Pi_{\mathsf{GAK}}(g, x, y)$

for $i = 1, ..., \lambda$
  $\tilde{g}_i \xleftarrow{\$} G$
  $\tilde{x}_i = \tilde{g}_i \star x$
  $\mathtt{rsp}_{i,0} = \tilde{g}_i$
  $\mathtt{rsp}_{i,1} = \tilde{g}_i \cdot g^{-1}$
  $\mathtt{cmt}_{i,0} = \mathsf{H}(\mathtt{rsp}_{i,0})$
  $\mathtt{cmt}_{i,1} = \mathsf{H}(\mathtt{rsp}_{i,1})$
$\mathtt{ch} = \mathsf{H}_{\mathtt{ch}}(\tilde{x}_1||\mathtt{cmt}_{1,0}||\mathtt{cmt}_{1,1}||...||\tilde{x}_\lambda||\mathtt{cmt}_{\lambda,0}||\mathtt{cmt}_{\lambda,1})$
for $i = 1, ..., \lambda$
  if $\mathtt{ch} = 0$
    $\mathtt{rsp}_i = (\mathtt{rsp}_{i,0}, \mathtt{cmt}_{i,1})$
else
    $\mathtt{rsp}_i = (\mathtt{rsp}_{i,1}, \mathtt{cmt}_{i,0})$
$\pi = (\mathtt{ch}, \{\mathtt{rsp}_i\})$

Fig. 10: Online extractable ZKP for group action knowledge

C Improved Analysis of the Threshold Scheme

The formal proof of Theorem 1 is organized as follows. We start by deriving in Section C.1 upper bounds on binomial coefficients from some of their well-known properties. Then, in Section C.2, we recall the definition of the total number $S(t, n)$ of shares distributed in the threshold scheme from [28], and some analysis from the same paper. Finally, in Section C.3 we show some new analysis and conclude the proof. The analysis includes a lemma characterizing the nested sums with dependent indices in terms of a binomial coefficient of the 'combinations with repetitions' type.

C.1 Preliminary Facts

Let e denote Euler's number (i.e., 2.781...). We start by recalling a well-known property of binomial coefficients and an upper bound for them.

Fact 1. For any $n, k \in \mathbb{N}$, such that $k \leq n$, it holds that

1. $\binom{n}{k} = \binom{n}{n-k}$,
2. $\binom{n}{k} \leq \left(\frac{en}{k}\right)^k$.

By applying both items in Fact 1, we derive the following

Fact 2. For any $a, b \in \mathbb{N}$, it holds that

$$\binom{a+b}{a} \leq \min\left(\left(\frac{e(a+b)}{b}\right)^b, \left(\frac{e(a+b)}{a}\right)^a\right).$$

The following fact recalls bounds on the factorial, as from [42], based on Stirling's factorial approximation.

Fact 3. [42] For any $n \in \mathbb{N}$, it holds that

$$\sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n \cdot \left(e^{1/(12n+1)}\right) < n! < \sqrt{2\pi n} \cdot \left(\frac{n}{e}\right)^n \cdot \left(e^{1/12n}\right).$$

Based on Fact 3, we obtain the following alternative upper bound on binomial coefficients.

Fact 4. For any $a, b \in \mathbb{N}$, it holds that

$$\binom{a+b}{a} \leq \sqrt{\frac{(a+b)}{2\pi \cdot a \cdot b}} \cdot \frac{(a+b)^{a+b}}{a^a b^b}.$$

Proof. The bound follows by starting with the definition of binomial coefficients, i.e.,

$$\binom{a+b}{a} = \frac{(a+b)!}{a! \cdot b!},$$

then applying the upper bound on $(a+b)!$ derived from Fact 3, and the lower bounds on $a!$ and $b!$ also derived from Fact 3, and then performing algebraic simplifications. □

We will also use the following parallel summation equality for binomial coefficients (see, e.g., Eq. (5.9) in [32]).

Fact 5. For any $n, k \in \mathbb{N}$, it holds that

$$\sum_{r \leq n} \binom{k+r}{r} = \binom{k+n+1}{n}.$$

C.2 Definitions and Analysis from [28]

To analyze the number of shares (i.e., group elements) distributed in the threshold scheme from [28], we recall definitions and facts of interest from that paper, as well as the definitions of Section 4.

Let $S(t,n)$ denote the number of group elements distributed to all n parties in the scheme. Then, it holds that

$$\begin{aligned}
S(t,n) =& \sum_{i=0}^{t} S\left(i, \left\lfloor \frac{n}{2} \right\rfloor\right) + S\left(t-i, \left\lceil \frac{n}{2} \right\rceil\right), && \text{for } t \in \left[2, \left\lfloor \frac{n}{2} \right\rfloor\right] \\
S(t,n) =& \sum_{i=t-\left\lfloor \frac{n}{2} \right\rfloor}^{\left\lfloor \frac{n}{2} \right\rfloor} S\left(i, \left\lfloor \frac{n}{2} \right\rfloor\right) + S\left(t-i, \left\lceil \frac{n}{2} \right\rceil\right), && \text{for } t \in \left[\left\lfloor \frac{n}{2} \right\rfloor + 1, n-1\right] \\
S(t,n) =& n && \text{for } t = 1 \text{ or } t = n \\
S(t,n) =& 0 && \text{for } t \leq 0 \text{ or } t > n.
\end{aligned}$$

The following preliminary fact helps determining a common upper bound for the above two equations for $S(t,n)$.

Fact 6. [28] For any $t, n \in \mathbb{N}$, such that $t \in [\lfloor \frac{n}{2} \rfloor + 1, n-1]$, it holds that

$$\sum_{i=t-\left\lfloor \frac{n}{2} \right\rfloor}^{\left\lfloor \frac{n}{2} \right\rfloor} S\left(i, \left\lfloor \frac{n}{2} \right\rfloor\right) + S\left(t-i, \left\lceil \frac{n}{2} \right\rceil\right) \leq \sum_{i=0}^{t} S\left(i, \left\lfloor \frac{n}{2} \right\rfloor\right) + S\left(t-i, \left\lceil \frac{n}{2} \right\rceil\right).$$

Proof. The proof follows by observing that the right-hand-side sum contains at least all elements in the left-hand-side sum. □

The following lemma characterizes $S(t,n)$ in terms of nested sums with dependent indices.

Lemma 1. [28] For any $t, n \in \mathbb{N}$, such that $t \in [2, n-1]$, it holds that

$$S(t,n) \leq 2n \cdot \sum_{i_1=0}^{t} \sum_{i_2=0}^{i_1} \cdots \sum_{i_{\lceil \log n \rceil}=0}^{i_{\lceil \log n \rceil - 1}} 1.$$

Proof. For $t \in [2, n-1]$, it holds that

$$\begin{aligned}
S(t,n) &\leq \sum_{i=0}^{t} S(i, \lfloor n/2 \rfloor) + S(t-i, \lceil n/2 \rceil), \\
&\leq 2 \cdot \sum_{i=0}^{t} S(i, \lceil n/2 \rceil), \\
&\leq 2^2 \cdot \sum_{i_1=0}^{t} \sum_{i_2=0}^{i_1} S(i_2, \lceil n/4 \rceil), \\
&\leq \ldots \\
&\leq 2^{\lceil \log n \rceil} \cdot \sum_{i_1=0}^{t} \sum_{i_2=0}^{i_1} \cdots \sum_{i_{\lceil \log n \rceil}=0}^{i_{\lceil \log n \rceil - 1}} S(i_{\lceil \log n \rceil}, 1), \\
&\leq 2n \cdot \sum_{i_1=0}^{t} \sum_{i_2=0}^{i_1} \cdots \sum_{i_{\lceil \log n \rceil}=0}^{i_{\lceil \log n \rceil - 1}} 1,
\end{aligned}$$

where the first inequality follows by Fact 6, the second inequality by term grouping, the next inequalities by repeated applications of Fact 6 and term grouping, and the last inequality follows from the definition of $S(t,n)$, for $n = 1$. This proves the lemma. □

C.3 Proof of Theorem 1

The following lemma characterizes the nested sums with dependent indices in the statement of Lemma 1 in terms of a binomial coefficient of the 'combinations with repetitions' type.

Lemma 2. For any $t, m \in \mathbb{N}$, it holds that

$$\sum_{i_1=0}^{t} \sum_{i_2=0}^{i_1} \cdots \sum_{i_m=0}^{i_{m-1}} 1 = \binom{t+m-1}{m}.$$

Proof. We prove that the equality in the lemma statement holds, by double induction over variables t and m.

When $t' = m' = 1$, both the left-hand side and the right-hand side are $= 1$.

Assuming the equality holds for $t' \leq t$ and $m' \leq m$, to show that the equality holds for $t' = t+1$ and $m' = m$, we can write

$$\begin{aligned}
\sum_{i_1=0}^{t+1} \sum_{i_2=0}^{i_1} \cdots \sum_{i_m=0}^{i_{m-1}} 1 &= \sum_{i_1=0}^{t} \sum_{i_2=0}^{i_1} \cdots \sum_{i_m=0}^{i_{m-1}} 1 + \sum_{i_2=0}^{t+1} \cdots \sum_{i_m=0}^{i_{m-1}} 1 \\
&= \binom{t+m-1}{m} + \sum_{i_2=0}^{t+1} \cdots \sum_{i_m=0}^{i_{m-1}} 1 \\
&= \quad \cdots \\
&= \binom{t+m-1}{m} + \binom{t+m-2}{m-1} + \cdots + \binom{t+1}{2} + t + 1 \\
&= \sum_{i=0}^{m} \binom{t-1+i}{i} \;=\; \binom{t+m}{m} \;=\; \binom{(t+1)+m-1}{m},
\end{aligned}$$

where the 2nd equality follows a single application of the induction hypothesis, the 4th equality follows from multiple applications of the induction hypothesis, and the 6th equality follows from Fact 5.

Now, assuming the equality holds for $t' \leq t$ and $m' \leq m$, to show that the equality holds for $t' = t$ and $m' = m+1$, we can write

$$\begin{aligned}
\sum_{i_1=0}^{t}\sum_{i_2=0}^{i_1}\cdots\sum_{i_{m+1}=0}^{i_m} 1 &= \sum_{i_2=0}^{0}\cdots\sum_{i_{m+1}=0}^{i_m} 1 + \cdots + \sum_{i_2=0}^{t}\cdots\sum_{i_{m+1}=0}^{i_m} 1 \\
&= \binom{m-1}{m} + \cdots + \binom{m-1+t}{m} \\
&= \binom{m}{m} + \cdots + \binom{m+t-1}{m} \\
&= \binom{m}{0} + \cdots + \binom{m-1+t}{t-1} \\
&= \sum_{i=0}^{t-1}\binom{m+i}{i} \;=\; \binom{m+t}{t-1} \;=\; \binom{t-1+(m+1)}{m+1},
\end{aligned}$$

where the 2nd equality follows by $t+1$ applications of the induction hypothesis, and the 6th equality follows from Fact 5. □

We can finally conclude the proof of Theorem 1, by using lemmas and facts so far established in this section to show the upper bounds on the number of group elements distributed in the threshold scheme.

Specifically, denoting $\lceil \log n \rceil$ as ℓ, we have that

$$S(t,n) \leq 2n \cdot \sum_{i_1=0}^{t}\sum_{i_2=0}^{i_1}\cdots\sum_{i_\ell=0}^{i_{\ell-1}} 1 \leq 2n \cdot \binom{t-1+\ell}{t-1} \leq 2n \cdot \min\{ub_0, ub_1, ub_2\},$$

where the 1st inequality follows from Lemma 1, the 2nd inequality follows from Lemma 2 for $m = \lceil \log n \rceil$, and the 3rd inequality follows from Fact 1 (bounds ub_0 and ub_1) and Fact 2 (bound ub_2).

This concludes the proof of Theorem 1.