Distributed Fiat-Shamir Transform: from Threshold Identification Protocols to Signatures

Michele Battagliola[1][0000−0002−8269−2148] and Andrea Flamini[1][0000−0002−3872−7251]

Università Politecnica delle Marche, Italia, battagliola.michele@proton.me,
Università di Trento, Italia, andrea.flamini@unitn.it

Abstract. The recent surge of distribute technologies caused an increasing interest towards threshold signature protocols, that peaked with the recent NIST First Call for Multi-Party Threshold Schemes.
Since its introduction, the Fiat-Shamir Transform has been the most popular way to design standard digital signature schemes. Many threshold signature schemes are designed in a way that recalls the structure of digital signatures created using Fiat Shamir, by having the signers generate a common commitment, compute the challenge as the hash of it, and then jointly create the response.
In this work we formalize this approach. In particular we introduce the notion of threshold identification scheme and threshold sigma protocol. Next, we introduce the concept of generalized Fiat-Shamir transform, that links the security of the threshold signature with the underlying threshold identification protocol. Our framework seeks to be an alternative, easier way to design concurrently secure threshold digital signatures and we show its potentiality providing an alternative security proof for Sparkle, a recent threshold Schnorr signature, and GRASS, a full threshold signature based on cryptographic group actions.

Keywords: Threshold Signatures · Fiat-Shamir Transform · Threshold Identification Schemes

1 Introduction

Decentralized systems are slowly becoming a desirable alternative to centralized ones, due to the advantages of distributing the management of data, such as avoiding single-points-of-failures or the secure storage of crypto-assets. For them to become a viable alternative, it is necessary to use secure decentralized cryptographic schemes. In particular, digital signature schemes assume a central role in this setting, as hinted by the amount of recent works on multi-user schemes and threshold variants of signature protocols, with a particular focus toward Schnorr, EdDSA and ECDSA [3,4,19,15], and by the recent NIST calls [12,11,10].

A common way to design threshold signatures is to translate a well-established digital signature schemes to the multi-party setting. Their security is then proved

with a reduction to the standard centralized scheme or directly to the hard problem they relies on. In this work, we provide a new framework for designing threshold signature protocols, without relying on already existing centralized signature schemes. To do so, we introduce the concept of threshold identification schemes, the decentralized version of the classical identification schemes, and show their ties with threshold signature algorithms.

Definitions In this work, we employ a game-based approach defining the security of a threshold signature, following the footprint of well established works like [23,15]. This is done to mimic the centralized case and to provide tools for designing and proving the security of threshold signatures in the same way is usually done for centralized signature (i.e. by proving the completeness, the special soundness and the honest verifier zero knowledge of the underlying sigma protocol)[26].

On the other hand, alternative definition such as UC-security exists [13]. Secure evaluation of threshold signature functionality is stronger than unforgeability and it is arguably overly strong in the sense that it often requires particular design decisions that are usually less efficient, such as the use of online-extractable zero-knowledge proofs.

Organization In Section 2 we define the cryptographic preliminaries needed in our paper. Next, in Section 3 we define threshold identification schemes, threshold sigma protocols and a threshold variant of the Fiat-Shamir Transform. Section 4 contains the core of our work: we present necessary security properties of the threshold identification scheme to obtain secure threshold signatures. Lastly, in Section 5 we show a possible application of our paradigm providing an alternative security proof for two recent threshold signatures: Sparkle and GRASS. Finally, in Section 6 we draw our conclusions and suggest some possible research directions arising from our work.

1.1 Our contribution and related works

The concept of distributed identification protocol has very few examples in literature: it was firstly introduced in [7], where M. Ben-Or et al. defined the concept of *multi provers zero knowledge proof*. However, the scope was limited to only two provers that could not communicate after starting an interaction with the verifier. The concept was later revised by Y. Desmedt et al. in [17], who maintained the setting of no communication between the provers, and by Pedersen in [25], who introduced the concept of multiple provers in the context of undeniable signatures. While Pedersen's focus is on robustness, we focus on the security of threshold identification schemes and their relation with threshold signatures. Lastly, M. Keller et al. in [21] introduced the concept of *multiple prover with combiner*: each of these provers communicate with a player, denoted as combiner, that combines the messages in the proof and communicates with the verifier, effectively playing the role of the prover in a standard ZKP.

Finally, C. Baum et al. in [5] introduced the concept of multiple verifiers that cooperate to verify a proof made by a single prover.

In this paper we flip the approach of [5], introducing the notion of *distributed identification protocol*, where the knowledge of the witness is shared among multiple provers cooperating in the production of a proof, which later will be verified by a single verifier. Contrary to the previous works such as [7,17], we allow for communication between the provers and we do not rely on the presence of a combiner handling the communication with the verifier, like in [21]. Instead we focus our attention to protocols where provers communicate and jointly produce the proofs.

We then introduce the distributed Fiat Shamir transform, a way to build threshold signatures starting from distributed identification protocols. Miming the approach proposed by Abdalla et al. in [1], we show that our transformation can be used to obtain unforgeable threshold digital signatures. In particular, we show that if the starting threshold identification protocol satisfies some security definitions, then the threshold signature obtained by applying the distributed Fiat-Shamir transform is unforgeable against active chosen message attacks.

2 Preliminaries

In Section 2.1 we introduce the notation that we use along the paper. In Section 2.2 we introduce the concepts of sigma protocol and identification scheme together with the security notions associated to such schemes. In Section 2.3 we introduce the Fiat-Shamir transform, one of the most common way to design digital signature schemes starting from identification protocols. Finally in Section 2.4 we define the concept of threshold signature scheme and the security notions associated to it.

2.1 Notation and terminology

If S is a set, $s \xleftarrow{\$} S$ means that s is sampled uniformly at random in the set S; we write $[n]$ to represent the set of numbers $\{1, 2, \dots, n\}$; for the sake of readability, when having an index set $J \subseteq \{1, \dots, n\}$ we write $\{a_i\}_J$ in place of $\{a_i\}_{i \in J}$.

With $y \leftarrow A(x_1, x_2, \dots)$ we refer to a deterministic algorithm A taking in input the values $x_1, x_2, \dots$ and returning the value y; if the input of some algorithm is clear from the context we might write $A(\cdot)$ instead of $A(x_1, x_2, \dots)$ and when the input is not explicitly necessary, we might omit it entirely, writing simply A; let A be an algorithm that produces an output y by accessing an oracle $\mathcal{O}$, then we write $y \leftarrow A^{\mathcal{O}}$.

If $A(x_1, x_2, \dots)$ is a probabilistic algorithm then we can use two notations for assigning to a variable y the output of $A(x_1, x_2, \dots)$: $y \xleftarrow{\$} A(x_1, x_2, \dots)$ where the symbol $\xleftarrow{\$}$ emphasizes the probabilistic nature of the algorithm $A(x_1, x_2, \dots)$ or $y \leftarrow A(x_1, x_2, \dots; R)$, where $R \xleftarrow{\$} \mathsf{Coins}(\lambda)$ is drawn from the set of random

coins $\mathsf{Coins}(\lambda)$, namely the set of bit strings of appropriate length which guarantees λ bits of randomness. If the randomness R is given in input to $A(x_1, x_2, \dots)$, the output y is uniquely determined.

Algorithms that start with the the letters T are multi-party algorithms that require communication between the parties. In particular each party has its own input identified with a subscript and the set of participants (usually denoted by J) is an explicit input of the function. For example $\mathsf{TSign}(\{a_i\}_J, \mathtt{m})$ means that the protocol TSign is a multi party protocol, run by party in J, with each party having a private input a_i while $\mathtt{m}$ is a common input. We also assume that the parties involved in the execution of multi-party algorithms have pairwise untappable authenticated communication channels.

We indicate the concatenation of strings $x_1, x_2 \dots, x_n$ as $x_1||x_2|| \dots ||x_n$. We also assume that, given a context, any string x can be uniquely parsed as a the concatenation of substrings.

2.2 Sigma protocols and identification schemes

A sigma protocol [9,16] for a relation $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{Y}$ is a three moves interactive protocol between a prover, holding a witness-statement pair $(w, y) \in \mathcal{R}$, and a verifier, knowing only the statement y. Roughly speaking, Sigma protocols work as follows:

1. In the first step, the prover sends the *first message* of the sigma protocol, often called *commitment*, $\textsc{Cmt} \in \mathcal{X}$ to the verifier.
2. Then the verifier returns a *challenge* $\textsc{Ch}$ consisting of a random string of fixed length $c(\lambda)$ which depends on the security parameter λ.
3. Lastly, the prover provides a *response* $\textsc{Rsp}$ and the verifier verifies it according to $y, \textsc{Cmt}, \textsc{Ch}$ and $\textsc{Rsp}$.

Usually, it is required that a sigma protocol satisfies the standard definitions of completeness, special soundness and honest verifier zero knowledge [9].

From sigma protocols to identification schemes When the relation $\mathcal{R}$ is hard (i.e. given only $y \in \mathcal{Y}$, it is hard to compute $w \in \mathcal{W}$ such that $(w, y) \in \mathcal{R}$) we can use sigma protocols to build canonical identification schemes. Informally speaking, we can imagine a canonical identification scheme as a sigma protocol equipped with a secure key generation algorithm for the relation $\mathcal{R}$ which generates key pairs $(w, y) \in \mathcal{R}$. In this case we say that w is the secret key, denoted by sk, while y is the public key, denoted by pk. From now on we omit to mention the hard relation $\mathcal{R}$ when not necessary.

Definition 1 (Identification scheme)**.** A canonical identification scheme is an interactive protocol between a prover P and a verifier V defined by the tuple

$$\mathcal{ID} = (\mathsf{Setup}(\cdot), \mathsf{Key\text{-}Gen}(\cdot), \mathsf{P}_{\textsc{Cmt}}(\cdot), \mathsf{P}_{\textsc{Rsp}}(\cdot), V(\cdot))$$

- $\mathsf{Setup}(\lambda)$: on input a security parameter λ, it outputs public parameters pp;

- $\mathsf{Key\text{-}Gen}(\mathsf{pp}; R)$: it is a probabilistic *key generation algorithm* that takes as input the public parameters pp and outputs a key pair $(\mathsf{sk}, \mathsf{pk})$;
- $\mathsf{P}_{\textsc{Cmt}}(\mathsf{pk}, \mathsf{pp}; R)$: it is a probabilistic algorithm called *prover commitment* that takes as input a secret key sk and outputs a commitment $\textsc{Cmt} \in \mathcal{X}$;
- $\mathsf{P}_{\textsc{Rsp}}(\mathsf{sk}, \textsc{Cmt}, \textsc{Ch}; R)$: it is a probabilistic algorithm called *prover response* that takes as input a private key sk, a commitment $\textsc{Cmt}$ and a challenge $\textsc{Ch}$ of fixed length $c(\lambda)$ and outputs a response $\textsc{Rsp}$;
- $V(\mathsf{pk}, \textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$: it is a deterministic algorithm, called Verifier, which takes as input a public key, a commitment $\textsc{Cmt}$, a challenge $\textsc{Ch}$ and a response $\textsc{Rsp}$, and outputs `accept` or `reject`.

A triple $(\textsc{Ch}, \textsc{Cmt}, \textsc{Rsp})$ such that $V(\mathsf{pk}, \textsc{Cmt}, \textsc{Ch}, \textsc{Rsp}) = \texttt{accept}$ is called *accepting conversation*.

Particularly interesting are non-trivial identification scheme, where the min-entropy of the first message is high. Formally:

Definition 2 (Min-entropy of $\mathsf{P}_{\textsc{Cmt}}$)**.** Being

$$\alpha(\mathsf{sk}) = \max_{\textsc{Cmt} \in \mathcal{X}} \{\Pr[\mathsf{P}_{\textsc{Cmt}}(\mathsf{pk}, \mathsf{pp}; R) = \textsc{Cmt} : R \xleftarrow{\$} \mathsf{Coins}(\lambda)]\}$$

the probability that $\mathsf{P}_{\textsc{Cmt}}$ outputs the most likely commitment $\textsc{Cmt}$, we define the *min-entropy* function associated to $\mathsf{P}_{\textsc{Cmt}}$ (or the min-entropy of the commitments) as

$$\beta(\lambda) = \min_{\mathsf{sk}} \left\{ \log_2 \frac{1}{\alpha(\mathsf{sk})} \right\}.$$

Definition 3 (Non-triviality)**.** A canonical identification scheme is called *non-trivial* if the min-entropy of the commitments is super-logarithmic in the security parameter λ [1].

An important notion of security for canonical identification schemes is the *security against impersonation under passive attacks (or eavesdropping attacks)*. Informally speaking, an identification scheme is secure against impersonation under passive attacks if no attacker can engage in an accepting conversation with an honest verifier, even after having access to a polynomial number of transcripts of interactions between the real prover and an honest verifier. For a formal definition see [1].

2.3 Fiat-Shamir Transform

Firstly introduced in [18], and then studied in [1,26] the Fiat-Shamir Transform is a widespread heuristic used to design digital signature schemes starting from canonical identification schemes. The general idea is to replace the challenge step with the evaluation of a cryptographic hash function on the concatenation of the message to be signed and the first message of the identification protocol. This transform is proven secure in the Random Oracle Model (ROM), in which

all the pseudo-random functions (usually hash functions) are replaced by random oracles which return truly random values upon invocation. In this paper we always assume the random oracle model.

In [1], Abdalla et al. prove that, if a non-trivial canonical identification scheme is secure against eavesdropping attack, then the digital signature scheme obtained by applying the Fiat-Shamir Transform is unforgeable under chosen message attacks.

2.4 Threshold signature schemes

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. This can be done by sharing the secret key among the multiple parties involved using a secret sharing scheme.

Definition 4 (Secret sharing)**.** A (t, n)-secret sharing scheme $\mathtt{SS}$ for a secret s is a pair of algorithm (D, R) such that share-generation function D takes in input a secret s and some randomness q and outputs a vector of n shares $(s_1, ..., s_n)$. The recovery function R takes input a set of t shares and outputs a value. We require the following properties:

- perfect security: $\mathbb{P}[\mathsf{secret} = s|\{s_i\}_J] = \mathbb{P}[\mathsf{secret} = s'|\{s_i\}_J]$ for every set $J \subseteq \{1, ..., n\}$ such that $|J| < t$.
- completeness: $\mathsf{R}(\{s_i\}_J) = s$ for all J with $|J| \geq t$.

In the following we write $\mathtt{SS}(s, t, n; R) = (s_1, ..., s_n)$ to refer to the algorithm for the creation of the shares of s for the (t, n)-secret sharing $\mathtt{SS}$ and we implicitly suppose $\mathtt{SS}$ is secure according to this definition.

Classically, threshold signature schemes comprise of four algorithms:

$$\mathcal{TDS} = (\mathsf{Setup}(\lambda), \mathsf{Key\text{-}Gen}(\mathsf{pp}, n, t), \mathsf{TSign}(\mathtt{m}, \{\mathsf{sk}_i\}_J), \mathsf{Ver}(\mathsf{pk}, \mathtt{m}, \sigma)).$$

However, since we suppose the presence of a trusted dealer, both the Setup and $\mathsf{Key\text{-}Gen}$ are not considered in our discussion.

- $\mathsf{Setup}(\lambda)$, on input a security parameter λ, it outputs public parameters pp.
- $\mathsf{Key\text{-}Gen}(\mathsf{pp}, n, t, \lambda)$, on input the number of participants n, the threshold t and the security parameter λ, it outputs a public key pk and a secret sharing $\{\mathsf{sk}_i\}_{i\in[n]}$ of the corresponding secret key sk, with participant P_i holding sk_i.
- $\mathsf{TSign}(\mathtt{m}, \{\mathsf{sk}_i\}_J)$ is a multi party protocol run by parties in J. On input an agreed upon message $\mathtt{m}$ and shards sk_i from various players, it outputs a valid signature σ if $|J| \geq t$,
- $\mathsf{Ver}(\mathsf{pk}, \mathtt{m}, \sigma)$, on input a public key pk, a message $\mathtt{m}$ and a signature σ, it outputs `accept` if the signature is valid, `reject` if not.

Informally, after an initial setup, any set of t parties who agrees on a common message $\mathtt{m}$ is able to jointly perform TSign to sign it. The resulting signature is verifiable against the public key pk via the verification algorithm Ver.

Security of threshold digital signatures There are several relevant security notions associated to threshold signatures, for instance security against passive chosen message attacks, against active chosen message attacks or adaptive chosen message attacks, each of them capturing an adversary with different capabilities in the way the attack is performed [9]. In this work we focus on the security against active chosen message attack even if most of our main results and definition could be easily adapted to cover also the passive case.

Definition 5 (Unforgeability under active chosen message attacks)**.** Let $\mathcal{TDS} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TSign}, \mathsf{Ver})$ be a (t, n)-threshold digital signature scheme.

Let $\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},F}(\lambda)$ be the experiment described below, where $\mathcal{F}$ is a forger who can corrupt up to $t-1$ parties of its choice and can perform a polynomial number q_s of queries to a signing oracle $\mathcal{O}_{\mathcal{TDS}}(\cdot)$ with which it interacts in the creation of the signatures of the queried messages, specifying also the set of parties hon the oracle must impersonate in the signing protocol execution.

$\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},F}(\lambda)$:

1 : $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(\lambda)$
2 : $(\{\mathsf{sk}_i\}_{i\in[n]}, \mathsf{pk}) \xleftarrow{\$} \mathsf{Key\text{-}Gen}(\mathsf{pp}, n, t)$
3 : $\mathsf{cor} \xleftarrow{\$} \mathcal{F}(\mathsf{pp}, \mathsf{pk}, n, t)$ // $\mathsf{cor} \subset [n], |\mathsf{cor}| < t$
4 : $(\mathtt{m}^\star, \sigma^\star) \leftarrow \mathcal{F}^{\mathcal{O}_{\mathcal{TDS}}}(\{\mathsf{sk}_i\}_{i\in\mathsf{cor}})$
5 : **return** $\mathsf{Ver}(\mathsf{pk}, \mathtt{m}^\star, \sigma^\star) \wedge m \notin Q$

$\mathcal{O}_{\mathcal{TDS}}(\mathsf{hon}, \mathtt{m})$

1 : **if** $|Q| < q_s$
2 : $\sigma \leftarrow \mathsf{TSign}(\mathtt{m}, \{\mathsf{sk}_i\}_{\mathsf{cor}\cup\mathsf{hon}})$
3 : // $\mathcal{F}$ and $\mathcal{O}_{\mathcal{TDS}}$ interact.
4 : $Q.\mathsf{add}(\mathtt{m}, \sigma)$
5 : // $\mathcal{F}$ records all the messages exchanged
6 : **return** $\perp$

Define the advantage of $\mathcal{F}$ in winning $\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},F}(\lambda)$

$$\mathrm{Adv}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda) = \mathbb{P}(\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda) = 1)$$

We say that $\mathcal{TDS}$ is *existentially unforgeable under active chosen message attacks* if $\mathrm{Adv}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda)(\cdot)$ is negligible for every probabilistic polynomial time forger $\mathcal{F}$.

Identifiable abort. A desirable feature of threshold signatures is the ability for the honest parties to perform identifiable abort. Informally, a threshold signature scheme supports identifiable abort if, in case of abort of the protocol execution, the honest participants can identify at least one participant who misbehaved. Typically this happens by requiring the participants to provide proofs of honest behaviour during the protocol execution or, in some cases, to prove that they behaved honestly after a protocol abort [20]. In this way the honest participants can repeat the protocol execution excluding the identified malicious party and substituting it with another possibly honest party.

Concurrent security Unlike for UC-secure [13] schemes, which automatically guarantee the security of the scheme also against an adversary who opens concurrent session, when we consider a game based security definition for threshold

digital signatures, the most that can be done to claim security against an adversary who open concurrent signing sessions is to show that:

- the reduction that is used to prove the security of the digital signature scheme can simulate in polynomial time the experiment $\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda)$ even if the adversary performs parallel signing queries;
- that the threshold signature scheme is not subject to practical attacks (ROS [8]) that exploit concurrent sessions.

.

3 Distributed Identification Schemes and Fiat-Shamir Transform

In Section 3.1 we generalise the definition of identification scheme to *threshold identification scheme* following the same principles used to define the security of threshold signatures. Then, in Section 3.2 we generalise the definition of sigma protocol to *threshold sigma protocol* and we describe a way to design secure threshold identification schemes derived from it. Finally in Section 3.3 we propose a generalisation of the Fiat-Shamir Transform to the distributed case and we identify two properties that the algorithm $\mathsf{TP}_{\mathrm{CMT}}$ might satisfy. These properties will be useful to characterize threshold identification schemes which can be turned, by applying the Fiat-Shamir transform, into threshold signature schemes secure against active chosen message attacks.

3.1 Threshold (canonical) identification schemes

We generalize Definition 1 and define protocols that allow multiple provers $P_1, ..., P_n$, holding a secret sharing of a secret sk, to prove their joint knowledge of sk. The idea is to replace both the $\mathsf{P}_{\mathrm{CMT}}$ and $\mathsf{P}_{\mathrm{RSP}}$ in the original definition with multi-party protocols that fulfill the same role. In particular $\mathsf{TP}_{\mathrm{CMT}}$ is run by a set J of provers to jointly produce a common first message CMT, then, after receiving a challenge CH, the parties in J jointly run $\mathsf{TP}_{\mathrm{RSP}}$ to produce a response RSP.

Definition 6 (Canonical $(t, n)-$ identification protocol)**.** Let $P_1, \ldots, P_n$ be a set of players, λ the security parameter and $c(\lambda)$ the challenge length. A *threshold identification protocol* is defined by the tuple

$$\mathcal{TID} = (\mathsf{Setup}(\cdot), \mathsf{Key\text{-}Gen}(\cdot), \mathsf{TP}_{\mathrm{CMT}}(\cdot), \mathsf{TP}_{\mathrm{RSP}}(\cdot), V(\cdot))$$

- $\mathsf{Setup}(\lambda)$: on input a security parameter λ, it outputs public parameters pp.
- $\mathsf{Key\text{-}Gen}(n, t, \mathsf{pp}; \mathbf{R})$: it is a probabilistic *key generation algorithm* that takes as input the public parameters pp, the number of participants n and the threshold t, and outputs a public key pk and a secret sharing $\mathtt{SS}(\mathsf{sk}, t, n; \mathbf{R}) = \{\mathsf{sk}_i\}_{[n]}$ of the secret key sk, with each participant P_i holding sk_i;

- $\mathsf{TP}_{\textsc{Cmt}}(\mathsf{pk}, \mathsf{pp}, J; \mathbf{R})$: it is a probabilistic multi-party protocol run by parties in J. On input the public parameters, the public key and the set of participants it outputs a common first message $\textsc{Cmt}$;
- $\mathsf{TP}_{\textsc{Rsp}}(\{\mathsf{sk}_i\}_J, \textsc{Cmt}, \textsc{Ch}; \mathbf{R})$: it is a probabilistic multi party protocol run by parties in J. It takes as input shards sk_i from the various player, a commitment $\textsc{Cmt}$ and a challenge $\textsc{Ch} \in \{0,1\}^{c(\lambda)}$, and outputs a valid response $\textsc{Rsp}$ if $|J| \geq t$;
- $V(\mathsf{pk}, \textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$: it is a centralized protocol called Verifier which takes in input a public key, a first message $\textsc{Cmt}$, a challenge $\textsc{Ch}$ and a response $\textsc{Rsp}$, and outputs `accept` or `reject`.

We require that when a set of t players $P_{i_1}, \ldots, P_{i_t}$ executes the protocol $\mathsf{TP}_{\textsc{Cmt}}(\mathsf{pk}, \mathsf{pp}, J)$, it receives a challenge from V and executes the protocol $\mathsf{TP}_{\textsc{Rsp}}(\mathsf{sk}_{i_1}, \ldots, \mathsf{sk}_{i_t}, \textsc{Cmt}, \textsc{Ch})$, then a verifier V with in input the public key outputs `accept` with probability 1.

In Figure 1 we represent the execution of a threshold identification scheme.

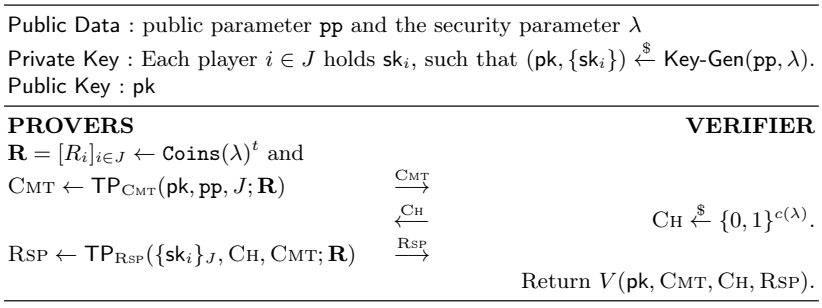

Fig. 1. Threshold identification scheme.

As before, since we are supposing the presence of a trusted dealer, both the Setup and Key-Gen are not considered in our discussion.

From now on, we refer to canonical (t, n)-identification schemes (and (t, n)-digital signatures) as threshold identification schemes (and threshold digital signatures).

Security notions for threshold identification schemes. In order to define security notions for threshold identification schemes, we adapt the security against impersonation under passive attacks (Definition 2.1 of [1]) to the distributed case. In particular we allow the adversary $\mathcal{I}$ (impersonator) to corrupt a subset $\{P_i\}_{i \in \mathsf{cor}}$ of the parties $\{P_i\}_{i \in [n]}$ with $|\mathsf{cor}| < t$. Before performing the impersonation attempt we allow the adversary to interact with an identification oracle $\mathcal{O}_{\mathcal{TID}}$ and execute a polynomial number q_i of queries for the execution of

the threshold identification protocol with the $\mathcal{O}_{\mathcal{TID}}$ controlling a set of honest parties $\{P_i\}_{i\in\mathsf{hon}}, |\mathsf{cor}\cup\mathsf{hon}| = t$ chosen by $\mathcal{I}$ (and the verifier by sampling a random challenge).

Definition 7 (Security against impersonation under active attacks)**.** Let $\mathcal{TID} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TP}_{\text{CMT}}, \mathsf{TP}_{\text{RSP}}, V)$ be a (t,n)-threshold identification scheme. Let $\text{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}(\lambda)$ be the experiment described below, where $\mathcal{I}$ is an impersonator having access to the threshold identification oracle $\mathcal{O}_{\mathcal{TID}}$.

Define the advantage of $\mathcal{I}$ in winning the experiment $\text{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}(\lambda)$ as

$$\text{Adv}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}(\lambda) = \mathbb{P}(\text{Exp}^{\text{a-imp}}_{\mathcal{TDS},\mathcal{I}}(\lambda) = 1)$$

We say that $\mathcal{TID}$ is *secure against impersonation under active attacks* if $\text{Adv}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}(\lambda)(\cdot)$ is negligible for every probabilistic polynomial time impersonator $\mathcal{I}$.

$\text{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}(\lambda)$:

1 : $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(\lambda)$
2 : $(\{\mathsf{sk}_i\}, \mathsf{pk}) \xleftarrow{\$} \mathsf{Key\text{-}Gen}(\mathsf{pp}, n, t)$
3 : $\mathsf{cor} \xleftarrow{\$} \mathcal{I}(\mathsf{pp}, \mathsf{pk}, n, t)$
4 : // $|\mathsf{cor}| < t$
5 : $st || \text{CMT}^\star \xleftarrow{\$} \mathcal{I}^{\mathcal{O}_{\mathcal{TID}}}(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in\mathsf{cor}})$
6 : $\text{CH}^\star \xleftarrow{\$} \{0,1\}^{c(\lambda)}$
7 : // $\mathcal{I}$ can perform other queries
8 : $st' \xleftarrow{\$} \mathcal{I}^{\mathcal{O}_{\mathcal{TID}}}(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in\mathsf{cor}}, \text{CMT}^\star, \text{CH}^\star)$
9 : $\text{RSP}^\star \xleftarrow{\$} \mathcal{I}^{\mathcal{O}_{\mathcal{TID}}}(\mathsf{pk}, \{\mathsf{sk}_i\}_{i\in\mathsf{cor}}, \text{CMT}^\star, \text{CH}^\star)$
10 : **return** $V(\mathsf{pk}, \text{CMT}^\star || \text{CH}^\star || \text{RSP}^\star)$

$\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, \mathsf{hon}, \mathsf{cor})$

1 : **if** $c < q_i$
2 : $c \leftarrow c + 1$
3 : // $\mathcal{O}_{\mathcal{TID}}$ interacts with $\mathcal{I}$
4 : $\text{CMT} \xleftarrow{\$} \mathsf{TP}_{\text{CMT}}(\mathsf{pk}, \mathsf{pp}, \mathsf{cor}\cup\mathsf{hon})$
5 : $\text{CH} \xleftarrow{\$} \{0,1\}^{c(\lambda)}$
6 : $\text{RSP} \xleftarrow{\$} \mathsf{TP}_{\text{RSP}}(\{\mathsf{sk}_i\}_{\mathsf{cor}\cup\mathsf{hon}}, \text{CMT}, \text{CH})$
7 : // $\mathcal{I}$ records all the messages exchanged
8 : **return** $\perp$

3.2 Threshold Sigma Protocols and Zero-Knowledge Property

We adapt Definition 6 to define a threshold Sigma protocol as follows.

Definition 8 (Threshold sigma protocol)**.** Let $\mathcal{R} \subseteq \mathcal{W}\times\mathcal{Y}$ be a relation (depending on the security parameter λ) and $\mathtt{SS}$ a secure (t,n)-secret sharing scheme for elements of $\mathcal{W}$. A *threshold sigma protocol* Σ for $\mathcal{R}$ and $\mathtt{SS}$ is defined by the tuple

$$\Sigma_{\mathcal{R},\mathtt{SS}} = (\mathsf{TP}_{\text{CMT}}(\mathsf{pk}, \mathsf{pp}, J; \mathbf{R}), \mathsf{TP}_{\text{RSP}}(\{w_i\}_J, \text{CMT}, \text{CH}; \mathbf{R}), V(y))$$

where the algorithms $\mathsf{TP}_{\text{CMT}}, \mathsf{TP}_{\text{RSP}}, V$ are defined as for *canonical* threshold identification schemes (Definition 6).

We require that when a set of t players $P_{i_1}, \ldots, P_{i_t}$ executes the protocol $\mathsf{TP}_{\textsc{Cmt}}(y, \mathsf{pp}, J; \mathbf{R})$, they receive a challenge from V and execute the protocol $\mathsf{TP}_{\textsc{Rsp}}(w_{i_1}, \ldots, w_{i_t}, \textsc{Cmt}, \textsc{Ch})$, then a verifier V with in input the statement y outputs `accept` with probability 1.

This definition naturally extends the definition of sigma protocol presented in [9, Section 19.4]. When $\mathcal{R}$ and $\mathtt{SS}$ are clear from the context, or it is not relevant, we will refer to a threshold sigma protocol as Σ instead of $\Sigma_{\mathcal{R},\mathtt{SS}}$.

We now describe properties which are the analogous of honest-verifier zero-knowledge and of (knowledge) soundness defined for sigma protocols [16].

We start with Active Honest-Verifier Zero Knowledge (AHVZK). Roughly speaking, a threshold identification protocol is AHVZK if no information about the private share is leaked during the execution. In particular, we require the existance of a simulator that

- takes in input a random challenge $\textsc{Ch}$;
- simulates the execution of $\mathsf{TP}_{\textsc{Cmt}}$ and $\mathsf{TP}_{\textsc{Rsp}}$, interacting with the adversary, without knowing its own private shards. The interaction needs to be indistinguishable from a real interaction.

Notice that the adjective "active" refers to the fact that the adversary is active in the interaction with the simulator, while "honest verifier" refers to the fact that the challenge is chosen randomly by the simulator (in the same way of honest-verifier zero-knowledge property of sigma protocols). More formally we have the following definition:

Definition 9 (Active Honest-Verifier Zero-Knowledge (AHVZK))**.** Let $\Sigma_{\mathcal{R},\mathtt{SS}}$ be a threshold sigma protocol for a relation $R \subseteq \mathcal{W} \times \mathcal{Y}$, secret sharing $\mathtt{SS}$ and challenge length $c(\lambda)$. Let $(w, y) \in \mathcal{R}$.

We say that Σ is *active honest-verifier zero-knowledge* if, for every adversary $\mathcal{A}_{y,\mathsf{cor}}(\{w_i\}_{i\in\mathsf{cor}})$ corrupting the set of parties $\{P_i\}_{i\in\mathsf{cor}}, \mathsf{cor} \subset [n], |\mathsf{cor}| < t$, there exists an efficient probabilistic algorithm $\mathcal{S}_{y,\mathsf{cor}}(\{w_i\}_{i\in\mathsf{cor}})$ controlling the honest parties in $[n] \setminus \mathsf{cor}$ such that, being $\mathsf{hon} \subset [n] \setminus \mathsf{cor}$ with $|\mathsf{cor} \cup \mathsf{hon}| = t$ chosen by $\mathcal{A}_{y,\mathsf{cor}}(\{w_i\}_{i\in\mathsf{cor}})$, the following two experiments are indistinguishable:

- Experiment 0: $\mathcal{A}_{y,\mathsf{cor}}(\{w_i\}_{i\in\mathsf{cor}})$ interacts with a challenger $\mathcal{C}(\{w_i\}_{i\in[n]}, y)$ engaging in a real execution of the sigma protocol, with $\mathcal{C}$ acting on behalf of the parties in hon and also as a honest verifier;
- Experiment 1: $\mathcal{S}_{y,\mathsf{cor}}(\{w_i\}_{i\in\mathsf{cor}}, \mathsf{hon})$ pick a random challenge $\textsc{Ch} \in \{0,1\}^{c(\lambda)}$, then $\mathcal{A}_{y,\mathsf{cor}}(\{w_i\}_{i\in\mathsf{cor}})$ interacts with the simulator $\mathcal{S}_{y,\mathsf{cor}}(\{w_i\}_{i\in\mathsf{cor}}, \mathsf{hon}, \textsc{Ch})$ in a simulated execution of the sigma protocol, with $\mathcal{S}_{y,\mathsf{cor}}(\{w_i\}_{i\in\mathsf{cor}}, \textsc{Ch})$ acting on behalf of the parties in hon and setting the challenge to $\textsc{Ch}$.

Note that $\mathcal{C}$ acting as an honest verifier samples uniformly at random a challenge after the execution of $\mathsf{TP}_{\textsc{Cmt}}$, instead $\mathcal{S}_{y,\mathsf{cor}}(\{w_i\}_{i\in\mathsf{cor}}, \mathsf{hon})$ samples it in advance.

Note that as a consequence of the definition of active zero knowledge, if $\mathcal{A}$ acts honestly, while interacting with $\mathcal{S}$, then $(\textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$ is an accepting conversation for y.

The special soundness definition is the same as the centralized case [9], we include it for completeness:

Definition 10 (Special Soundness)**.** Let Σ be a threshold sigma protocol for a relation $R \subseteq \mathcal{W} \times \mathcal{Y}$. We say that Σ is special sound if and only if there exists an efficient deterministic algorithm $\mathcal{E}$, called extractor, with the following property: whenever $\mathcal{E}$ is given as input a statement $y \in \mathcal{Y}$, two accepting conversations $(\textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$ and $(\textsc{Cmt}, \textsc{Ch}', \textsc{Rsp}')$, with $\textsc{Ch} \neq \textsc{Ch}'$ $\mathcal{E}$ outputs $w \in \mathcal{W}$ such that $(w, y) \in \mathcal{R}$.

This definition naturally extends to k-special soundness, where k is the number of transcripts with the same commitment and different challenges that must be provided to an extractor to extract the witness w for y.

It is easy to see that given a generic threshold sigma protocol $\Sigma_{\mathcal{R},\mathsf{SS}}$ there exists a generic threshold identification scheme in which the Setup generates the parameters for the relation $\mathcal{R}$, and $\mathsf{Key\text{-}Gen}$ generates the key pair $(\mathsf{sk}, \mathsf{pk})$ and computes $\{\mathsf{sk}_i\}_{i\in[n]} \xleftarrow{\$} \mathsf{SS}(t, n, \mathsf{sk})$.

Theorem 1. *Let $\Sigma_{\mathcal{R},\mathsf{SS}} = (TP_{C_{MT}}, TP_{R_{SP}}, V)$ be a $(t, n)-$ threshold Sigma protocol for a hard relation $\mathcal{R} \subseteq \mathcal{W}\times\mathcal{Y}$, and secret sharing SS, with super-polynomial challenge space $\{0,1\}^{c(\lambda)}$. Let*

$$\mathcal{TID} = (\mathit{Setup}, \mathit{Key\text{-}Gen}, TP_{C_{MT}}, TP_{R_{SP}}, V)$$

be the threshold identification scheme derived from it. If Σ is AHVZK and special sound, then the $\mathcal{TID}$ is secure against active impersonation attacks.

Proof. We want to show that if there exists an adversary $\mathcal{I}$ able to win the $\mathrm{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}$ game with non-negligible advantage, then it is possible to build a simulator $\mathcal{B}(\mathsf{pk})$ that, simulating the challenger of experiment $\mathrm{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}$, is able to extract the secret key sk such that $(\mathsf{sk}, \mathsf{pk}) \in \mathcal{R}$, contradicting the hardness of the relation $\mathcal{R}$.

$\mathcal{B}(\mathsf{pk})$ receives from $\mathcal{I}$ the set of parties it want to corrupt $\mathsf{cor}, |\mathsf{cor}| < t$, then $\mathcal{B}$ sends to $\mathcal{I}$ random shares $\{\mathsf{sk}_i\}_{i\in\mathsf{cor}}$. Since the secret sharing scheme is secure according to Definition 4, this is indistinguishable from an execution of a real secret sharing, since $\mathcal{I}$ controls less than t parties.

Simulation of identification queries. $\mathcal{B}$ is able to simulate the oracle $\mathcal{O}_{\mathcal{TID}}(\cdot)$ thanks to the AHVZK property of Σ. In particular for each identification query, $\mathcal{I}$ sends to $\mathcal{B}$ the set of indexes hon associated to the parties it want to interact with. Then $\mathcal{B}$ samples $\textsc{Ch} \xleftarrow{\$} \{0,1\}^{c(\lambda)}$, and executes $\mathcal{S}_{\mathsf{pk},\mathsf{cor}}(\textsc{Ch}, \mathsf{hon})$ simulating in this way the execution of $\mathcal{O}_{\mathcal{TID}}$, where the challenge used in the execution of $\mathcal{TID}$ is $\textsc{Ch}$ and $\mathcal{S}_{\mathsf{pk},\mathsf{cor}}$ is the simulator of Definition 9.

Exploit of the impersonation of $\mathcal{I}$. $\mathcal{I}$ will eventually perform a successful impersonation. At this point $\mathcal{B}$ rewinds $\mathcal{I}$ and changes the challenge corresponding to the first message used in the impersonation attempt. Since the challenge space is super-polynomial, with non-negligible probability this yields the two required accepting conversation (by the Forking Lemma [27], see Appendix C), thus $\mathcal{B}$ can use the extractor $\mathcal{E}$ from the special soundness Definition 10 to extract a witness w breaking the hardness of the relation $\mathcal{R}$. □

3.3 Distributed Fiat-Shamir Transform

We adapt the definition of *Fiat-Shamir transform* to the distributed case:

Definition 11 (Distributed Fiat-Shamir transform)**.** Let $\mathcal{TID}$ be a canonical threshold identification scheme with $\mathcal{TID} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TP}_{\mathrm{CMT}}, \mathsf{TP}_{\mathrm{RSP}}, V)$.

We define the threshold digital signature $\mathcal{TDS}$ built from the canonical $(t, n)-$identification scheme $\mathcal{TID}$ using the Fiat-Shamir transform as $\mathcal{TDS} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TSign}, \mathsf{Ver})$.

The signature has the same Setup and $\mathsf{Key\text{-}Gen}$ algorithm as the identification scheme, and the output length of the hash function equals the challenge length of the identification scheme. Let J be a set of signers with $|J| \geq t$. The signing and the verification algorithms are defined as follows:

$\mathsf{TSign}(\mathtt{m}, \{\mathsf{sk}_i\}_{i\in J})$:

1 : $\mathbf{R} \xleftarrow{\$} \mathsf{Coins}^t(\lambda)$
2 : $\mathrm{CMT} \leftarrow \mathsf{TP}_{\mathrm{CMT}}(\mathsf{pk}, \mathsf{pp}, J; \mathbf{R})$
3 : $\mathrm{CH} \leftarrow H(\mathrm{CMT}||\mathtt{m})$
4 : $\mathrm{RSP} \leftarrow \mathsf{TP}_{\mathrm{RSP}}(\{\mathsf{sk}_i\}_{i\in J}, \mathrm{CMT}, \mathrm{CH}; \mathbf{R})$
5 : **return** $\mathrm{CMT}||\mathrm{RSP}$

$\mathsf{Ver}(\mathsf{pk}, \mathtt{m}, \sigma)$:

1 : Parse σ as $\mathrm{CMT}|\mathrm{RSP}$
2 : $\mathrm{CH} \leftarrow H(\mathrm{CMT}||\mathtt{m})$
3 : **return** $V(\mathsf{pk}, \mathrm{CMT}, \mathrm{CH}, \mathrm{RSP})$

We now define two properties on $\mathsf{TP}_{\mathrm{CMT}}$ that are useful when discussing the security.

Requirements on $\mathsf{TP}_{\mathrm{CMT}}$ We want to characterise a class of algorithms $\mathsf{TP}_{\mathrm{CMT}}$ whose design guarantees that the output has high min-entropy if at least one of the parties taking part to the execution of the algorithm is not controlled by an adversary. Adapting [9, Definition 19.7] to the multi-party case, we refer to this class of $\mathsf{TP}_{\mathrm{CMT}}$ as *unpredictable*.

Definition 12 (Unpredictable $\mathsf{TP}_{\mathrm{CMT}}$)**.** Let $\mathcal{TID}$ be a (t, n)-threshold identification scheme with $\mathcal{TID} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TP}_{\mathrm{CMT}}, \mathsf{TP}_{\mathrm{RSP}}, V)$. We say that $\mathsf{TP}_{\mathrm{CMT}}$ is *unpredictable*, if the output CMT has super-logarithmic min-entropy when at least one party is honest.

As we will show, the unpredictability property of TP_{CMT} is not enough to guarantee the security of the threshold signature obtained by applying the distributed Fiat-Shamir transform to a threshold identification scheme. Indeed, to guarantee active unforgeability of the threshold signature scheme, a simulator of the experiment $\text{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}(\lambda)$ in Definition 7 needs to be able to extract from the adversary its secret values used in the computation of CMT, before the adversary can compute it. Moreover, we require the simulator to extract "online" the commitment, without rewinding the adversary, to guarantee the concurrent security of the threshold signature scheme against practical attacks [8]. More formally we have the following definition:

Definition 13 (Extractable TP_{CMT})**.** Let $\mathcal{TID}$ be a threshold identification scheme with $\mathcal{TID} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TP}_{\text{CMT}}, \mathsf{TP}_{\text{RSP}}, V)$.

We say that TP_{CMT} is online *extractable* if:

- it comprise two consecutive sub-protocols, $\mathsf{TP}^1_{\text{CMT}}$ and (deterministic) $\mathsf{TP}^2_{\text{CMT}}$, with the input of $\mathsf{TP}^2_{\text{CMT}}$ being the output of $\mathsf{TP}^1_{\text{CMT}}$ in addition to every other input of TP_{CMT}, which are executed one immediately after the other,
- for every non empty sets of parties A, B, such that $|A \cup B| = t$, there exists a simulator $\mathcal{S}_{\mathsf{ext}}$ that, on input all the messages from parties in A to parties in B during the execution of $\mathsf{TP}^1_{\text{CMT}}$ as well as every oracle query made by parties in A, is able to simulate the execution of $\mathsf{TP}^2_{\text{CMT}}$ on behalf of all the parties in A interacting with parties in B in an indistinguishable way.
- For every probabilistic polynomial time adversary $\mathcal{I}$, the probability of $\mathcal{I}$ guessing the output of $\mathsf{TP}^2_{\text{CMT}}$ before the execution of $\mathsf{TP}^2_{\text{CMT}}$ is negligible.

As we will see, the hypothesis about the input of the simulator could be lowered, allowing $\mathcal{S}_{\mathsf{ext}}$ to rewind the parties in A. However, doing so decrease the security of the signature that, while still unforgeable, would not be secure in the concurrent setting.

While the definition does not explicitly mention an extractor, the most common way to achieve this properties is to exploit the messages sent in $\mathsf{TP}^1_{\text{CMT}}$ to extract the private data of the adversary and use them to compute CMT ahead of time.

Observation 1. Given that the impersonator $\mathcal{I}$ has negligible probability of guessing CMT at the end of $\mathsf{TP}^1_{\text{CMT}}$, it follows immediately that every extractable TP_{CMT} is also an unpredictable TP_{CMT}.

Observation 2 (Commit-Release TP_{CMT})**.** A common way to design an extractable TP_{CMT} in the random oracle model is to use a "commit-release" structure, using an hash function for the commitment step. In particular, let $\mathcal{TID}$ be a threshold identification scheme with $\mathcal{TID} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TP}_{\text{CMT}}, \mathsf{TP}_{\text{RSP}}, V)$ where TP_{CMT} is defined by the following two sub-protocols:

- $\mathsf{TP}^{\text{Com}}_{\text{CMT}}(\mathsf{pk}, \mathsf{pp}, J; R)$ a non interactive protocol run locally by each party that outputs a commitment $\mathsf{com}_i \leftarrow H(\mathtt{ssid}||\text{CMT}_i)$ where CMT_i has high min-entropy, and $\mathtt{ssid}$ is a session identifier shared among all the parties involved in the execution.

– $\mathsf{TP}^{\mathrm{Rel}}_{\mathrm{CMT}}(\{\mathsf{Commit}(\mathtt{ssid}||\mathrm{CMT}_i)\}_{i\in[t]})$: each party reveals the value CMT_i, checks that the values CMT_j revealed by the parties P_j are consistent with their commitments com_j, and output a common CMT, obtained deterministically by combining the partial first messages CMT_i. The combining function must satisfies an analogous property of Definition 12, i.e. as long as at least one CMT_i has high min-entropy, then also final output CMT has high min-entropy.

It is easy to see that as long as H is a cryptographic hash function (thus is pre-image resistant) and CMT_i has high min-entropy both the hypothesis of Definition 13 are satisfied. Indeed, the extractor can learn all the CMT_i from the adversary from the random oracle queries and, since $\mathsf{TP}^{\mathrm{Rel}}_{\mathrm{CMT}}$ is deterministic, it can use them to compute CMT.

4 Main Result

In this section we state and prove our main result, namely the relation between the security of threshold identification protocol and the security of the threshold signature obtained by applying the distributed Fiat-Shamir Transform.

Theorem 2 (Active security)**.** *Let $\mathcal{TID} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TP}_{C_{MT}}, \mathsf{TP}_{R_{SP}}, V)$ be a canonical threshold identification scheme. Consider the associated signature scheme $\mathcal{TDS} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TSign}, \mathsf{Ver})$ as per Definition 11. Then, assuming the ROM, the following implications hold:*

1. *($\mathcal{TID} \implies \mathcal{TDS}$): if $\mathsf{TP}_{C_{MT}}$ is extractable as per Definition 13 and $\mathcal{TID}$ is secure against impersonation under active attacks, then $\mathcal{TDS}$ is secure against active chosen-message attacks.*
2. *($\mathcal{TDS} \implies \mathcal{TID}$): If $\mathcal{TDS}$ is secure against active chosen-message attacks, then $\mathcal{TID}$ is secure against impersonation under active attacks.*

As an immediate corollary, thanks to Theorem 1 we have that if $\mathsf{TP}_{\mathrm{CMT}}$ is extractable as per Definition 13 and the underlying sigma protocol is AHVZK and special sound, then the threshold signature is unforgeable.

We now prove separately the two implications of *Theorem* 2.

Lemma 1 ($\mathcal{TID} \implies \mathcal{TDS}$). *Under the assumptions of Theorem 2,if $\mathsf{TP}_{C_{MT}}$ satisfies the extractability property of Definition 13 and $\mathcal{TID}$ is secure against impersonation under active attacks, then $\mathcal{TDS}$ is unforgeable against active chosen-message attacks.*

Proof. We first provide a general idea of the proof, then we provide a detailed discussion.

Overview. Let $\mathcal{F}$ be a forger for the threshold signature. Our goal is to build an impersonator $\mathcal{I}$, that use $\mathcal{F}$ as a subroutine, to win the impersonation game. The impersonator $\mathcal{I}$ interacts with the challenger $\mathcal{C}_{\mathcal{TID}}$ of experiment $\text{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{A}}(\lambda)$ and has access to the transcript oracle $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, \mathsf{cor}, \mathsf{hon})$ that can query up to $q_s(\lambda)$ times, where cor and hon are chosen by $\mathcal{I}$. In order to exploit the advantage of $\mathcal{F}$, and use it as a subroutine, $\mathcal{I}$ simulates the challenger $\mathcal{C}_{\mathcal{TDS}}$ of the experiment $\text{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},F}(\lambda)$ executed by $\mathcal{F}$ and the sign and random oracles $\mathcal{O}^{H}_{\mathcal{TDS}}(\{sk_i\}_{i\in\mathsf{cor}})$[1] and $\mathcal{O}_H(\cdot)$ to which it can make respectively $q_s(\lambda)$ and $q_h(\lambda)$ queries, both polynomial in the security parameter λ, being $\mathcal{F}$ a polynomial-time adversary.

In Appendix A, Figure 4, we provide a graphical representation of the reduction we describe below.

Defining the forger. Let $\mathcal{F}$ be a forger that wins the $\text{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},F}(\lambda)$ with non-negligible advantage $\epsilon(\lambda)$. We require that $\mathcal{F}$ satisfies the following properties (as in [1]):

- all of its hash queries have the form $\textsc{Cmt}||\mathtt{m}$ with $\textsc{Cmt} \in \mathcal{X}, \mathtt{m} \in \{0,1\}^*$;
- before outputting a forgery $(\mathtt{m}, \textsc{Cmt}||\textsc{Rsp})$, $\mathcal{F}$ has performed an hash query for $(\textsc{Cmt}||\mathtt{m})$;
- if $\mathcal{F}$ outputs $(\mathtt{m}, \textsc{Cmt}||\textsc{Rsp})$, $\mathtt{m}$ was never a sign query.

It is easy to see that if there exists a forger $\mathcal{F}'$ who does not satisfy these requirements, it is possible to build a forger $\mathcal{F}$ satisfying the requirements using $\mathcal{F}'$ as a subroutine, as discussed in [1], Proof of Lemma 3.5.

Initialization. $\mathcal{I}$ initializes the hash query counter $hc = 0$ and the sign query counter $sc = 0$. $\mathcal{I}$ also initializes the hash table $\mathsf{HT} = \emptyset$, and the query table $\mathsf{QT} = \emptyset$, then generates a random forge pointer $fp \in [q_h(\lambda)]$.

$\mathcal{I}$ receives from $\mathcal{C}_{\mathcal{TID}}$ the public parameters pp of the identification protocol and the public key pk. $\mathcal{I}$ forwards this information to $\mathcal{F}$.

Training phase. $\mathcal{F}$ chooses the set cor (with $|\mathsf{cor}| \leq t-1$) of actors it wants to control. $\mathcal{I}$ chooses the same set cor and sends it to $\mathcal{C}_{\mathcal{TID}}$, receiving the secret keys of the players in cor, finally $\mathcal{I}$ forwards this information to $\mathcal{F}$.

Now $\mathcal{F}$ can perform $q_h(\lambda)$ hash queries and $q_s(\lambda)$ sign queries to $\mathcal{I}$. In the first case $\mathcal{I}$ uses the hash table HT to answer, while in the second $\mathcal{I}$ performs an identification query to its oracle $\mathcal{O}_{\mathcal{TID}}$ using the same input as part of the $\text{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}(\lambda)$ game. Specifically, the simulation works as follows:

- *$\mathcal{F}$ performs an hash query with input $x \in \{0,1\}^*$:* $\mathcal{I}$ returns $\mathsf{HT}[x]$ if it is defined. Otherwise, $\mathcal{I}$ increases the counter hc by 1 and sets $\mathsf{QT}[hc] = x$, then, if $hc \neq fp$, $\mathcal{I}$ picks uniformly at random $d \in \{0,1\}^{c(\lambda)}$, sends it to $\mathcal{F}$ and sets $\mathsf{HT}[x] = d$. If $hc = fp$ it parses x as $\textsc{Cmt}^*||\mathtt{m}^*$, sends to the

[1] We denote the signing oracle $\mathcal{O}^{H}_{\mathcal{TDS}}(\{sk_i\}_{i\in\mathsf{cor}})$ in place of $\mathcal{O}_{\mathcal{TDS}}(\{sk_i\}_{i\in\mathsf{cor}})$ to emphasize that it depends on the hash function H.

challenger $\mathcal{C}_{\mathcal{TID}}$ $\textsc{Cmt}^*$ as the first move of the impersonation attempt of the $\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},F}(\lambda)$ game and receives back from $\mathcal{C}_{\mathcal{TID}}$ a challenge $\textsc{Ch}^*$. In this case, I sets $\mathsf{HT}[x] = \textsc{Ch}^*$ and sends $\textsc{Ch}^*$ to $\mathcal{F}$. This procedure allows $\mathcal{I}$ to perfectly simulate the random oracle $\mathcal{O}_H$.

– *$\mathcal{F}$ performs a sign query for message* $\mathtt{m}$: $\mathcal{F}$ chooses hon, the set of the honest players who, together with the parties in cor, participates in the computation of a signature of $\mathtt{m}$. $\mathcal{I}$ increases the signature counter *sc* and sends to $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, \mathsf{cor}, \mathsf{hon})$ a request to perform the threshold identification protocol. The impersonator $\mathcal{I}$ acts as a man in the middle between $\mathcal{F}$ and $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, \mathsf{cor}, \mathsf{hon})$, and repeats the following operations for each step prescribed by the algorithm $\mathsf{TP}_{\textsc{Cmt}}$:
  1. $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, \mathsf{cor}, \mathsf{hon})$ produces the messages for the participants in hon, and anticipates $\mathcal{I}$ in the execution of the steps prescribed by $\mathsf{TP}^1_{\textsc{Cmt}}$;
  2. $\mathcal{I}$ forwards to $\mathcal{F}$ the messages received from $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, \mathsf{cor}, \mathsf{hon})$;
  3. $\mathcal{F}$ produces the messages executing $\mathsf{TP}^1_{\textsc{Cmt}}$ on behalf of the corrupted participants in cor.
  4. $\mathcal{I}$ forwards the messages received from $\mathcal{F}$ to $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, \mathsf{cor}, \mathsf{hon})$.

  These steps are repeated for the protocol $\mathsf{TP}^2_{\textsc{Cmt}}$, leading to the computation of the shared $\textsc{Cmt}$ by $\mathcal{F}$ at first, then by $\mathcal{I}$, once it receives the openings of the parties in cor from $\mathcal{F}$ and finally by $\mathcal{O}_{\mathcal{TID}}$, once it receives the messages forwarded by $\mathcal{I}$.
  The oracle $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, \mathsf{cor}, \mathsf{hon})$ produces a random challenge $\textsc{Ch}$ and $\mathcal{I}$ sets $\mathsf{HT}[\textsc{Cmt}||\mathtt{m}] = \textsc{Ch}$. This operation may overwrite the hash table HT but we show later that the probability of this happening is negligible if the simulator $\mathcal{I}$ adopts the countermeasures that we prescribe later in the proof.
  Together with $\textsc{Ch}$, $\mathcal{O}_{\mathcal{TID}}$ sends its contribution to the execution of $\mathsf{TP}_{\textsc{Rsp}}$ that $\mathcal{I}$ forwards to $\mathcal{F}$. As for the creation of $\textsc{Cmt}$, $\mathcal{I}$ acts as a man in the middle between $\mathcal{F}$ and $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, \mathsf{cor}, \mathsf{hon})$ in the execution of $\mathsf{TP}_{\textsc{Rsp}}$ until the identification process is completed as well as the signing process and the algorithm $\mathsf{TP}_{\textsc{Rsp}}$ outputs the response $\textsc{Rsp}$ and therefore $\mathcal{F}$ creates, together with $\mathcal{I}$ the signature $(\textsc{Cmt}||\textsc{Rsp})$ of $\mathtt{m}$.

This concludes the description of the simulation of the experiment of unforgeability under active attacks for $\mathcal{F}$ performed by $\mathcal{I}$. In order to state that $\mathcal{I}$ correctly simulates the experiment it remains to show that the simulation fails only with negligible probability.

Simulation failure. We now focus on the cases in which the simulation may fail and we find an upper bound to the probability that such failure happens. We have shown that the simulation of $\mathcal{I}$ fails only if $\mathcal{I}$ is forced to overwrite the hash table HT during a sign query performed by $\mathcal{F}$. The overwriting of HT during a sign query might refer to a previous hash query or to a previous sign query, therefore we must consider separately the following two cases.

1. During a sign query for a message $\mathtt{m}$, before computing $\textsc{Cmt}$, $\mathcal{F}$ has performed a hash query for $\textsc{Cmt}||\mathtt{m}$. We must consider again two possible scenarios:

   (a) During the execution of $\mathsf{TP}^2_{\text{CMT}}$ or at the end of it. Since $\mathcal{F}$ is malicious we can suppose that $\mathcal{F}$ always sends its messages after all the honest parties. In this setting, $\mathcal{F}$ learns the final output CMT before $\mathcal{I}$. Later we discuss how to deal with this case such that it never occurs and thus does not contribute in the evaluation of the failure probability of the simulation performed by $\mathcal{I}$.
   (b) Before the execution of $\mathsf{TP}^2_{\text{CMT}}$. By the third properties of Definition 13 and the unpredictability of TP_{CMT}, the output of TP_{CMT} has min-entropy $\beta(\lambda)$, super-logarithmic in λ, thus the probability of the adversary guessing CMT is less than $\frac{q_h(\lambda)}{2^{\beta(\lambda)}}$ which is negligible being $2^{\beta(\lambda)}$ super-polynomial in λ.
2. Before producing the first message CMT associated to a sign query for $\mathtt{m}$, $\mathcal{F}$ has performed another sign query for $\mathtt{m}$, and the output of TP_{CMT} results to be the same CMT. For $n \in [q_s(\lambda)]$ we define $\mathcal{X}_n \subset \mathcal{X}$ the set of first messages CMT generated in the previous $n-1$ sign queries, then the failure probability of the simulation during sign query n for a collision of the first message with the first message of a previous query is:

$$\mathbb{P}[\text{CMT} \in \mathcal{X}_n] = \frac{n-1}{2^{\beta(\lambda)}}$$

The probability that $\mathcal{I}$ is forced to overwrite the hash table HT during the n-th sign query (when the sign counter $sc = n$) is

$$\mathbb{P}[\mathcal{I} \text{ overwrites when } sc = n] = \frac{q_h(\lambda) + (n-1)}{2^{\beta(\lambda)}}.$$

Therefore the probability that $\mathcal{I}$ fails its simulation and overwrites the hash table is:

$$\begin{aligned}\mathbb{P}[\mathcal{I} \text{ fails}] &\leq \sum_{n=1}^{q_s(\lambda)} \frac{(n-1) + q_h(\lambda)}{2^{\beta(\lambda)}} = \\ &= \frac{q_h(\lambda) q_s(\lambda)}{2^{\beta(\lambda)}} + \sum_{n=1}^{q_s(\lambda)} \frac{(n-1)}{2^{\beta(\lambda)}} = \\ &= \frac{q_h(\lambda) q_s(\lambda) + q_s(\lambda)(q_s(\lambda)-1)/2}{2^{\beta(\lambda)}}\end{aligned}$$

Therefore it holds that

$$\mathbb{P}[\mathcal{I} \text{ fails}] \leq \frac{q_s(\lambda)(q_h(\lambda) + q_s(\lambda) - 1)}{2^{\beta(\lambda)}} \tag{4.1}$$

which is negligible in λ.

We now focus on the case described in Item 1a and explain how we deal with that.

At the end of $\mathsf{TP}^2_{\textsc{Cmt}}$, $\mathcal{F}$ knows the first message Cmt that will be used to create the signature on $\mathtt{m}$ and could ask for an hash query on it before sending Cmt to $\mathcal{I}$.

Without any countermeasure, if $\mathcal{F}$ would do so, $\mathcal{I}$ would returns a random digest d according to the simulation of the random oracle. However, when $\mathcal{F}$ would eventually reveal Cmt, $\mathcal{I}$ would not be able to set $\mathsf{HT}[\mathtt{m}||\textsc{Cmt}] = \textsc{Ch}$ where Ch is the challenge received by the oracle $\mathcal{O}_{\mathcal{TID}}$, causing a failure.

All of this is avoided thanks to the extractability properties of $\mathsf{TP}_{\textsc{Cmt}}$, see Definition 13. Indeed, at the end of $\mathsf{TP}^1_{\textsc{Cmt}}$, $\mathcal{I}$ can use the simulator of Definition 13 to anticipate the execution of $\mathsf{TP}^2_{\textsc{Cmt}}$ in the $\mathrm{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}$ game. This allows $\mathcal{I}$ to learn Cmt and Ch before the execution of $\mathsf{TP}^2_{\textsc{Cmt}}$ in the $\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},F}$, and thus, when $\mathcal{F}$ does an oracle query on input $\mathtt{m}||\textsc{Cmt}$, $\mathcal{I}$ can answer Ch and set $\mathsf{HT}[\mathtt{m}||\textsc{Cmt}] = \textsc{Ch}$. In this way $\mathcal{I}$ can always answer the random oracle queries made by $\mathcal{F}$ consistently and does not risk to overwrite the hash table. Notice that thanks to the last point of Definition 13 it is impossible for $\mathcal{F}$ to learn Cmt before $\mathcal{I}$, so this strategy never fails.

Exploit of $\mathcal{F}$'s forgery. Once $\mathcal{F}$ has concluded the training phase, $\mathcal{F}$ outputs a forgery $(\widehat{\textsc{Cmt}}, \widehat{\textsc{Rsp}})$ of a message $\widehat{m}$ not previously queried. Then $\mathcal{I}$ concludes its impersonation attempt by sending the message $\widehat{\textsc{Rsp}}$ as a response to the challenge $\textsc{Ch}^*$ received after the fp-th hash query, associated to the commitment $\textsc{Cmt}^*$.

Note that if $\widehat{\textsc{Cmt}} = \textsc{Cmt}^*$, $\widehat{\mathtt{m}} = \mathtt{m}^*$ and $(\widehat{\textsc{Cmt}}, \widehat{\textsc{Rsp}})$ is a valid forgery of $\widehat{\mathtt{m}} = \mathtt{m}^*$, which happens if fp was guessed by $\mathcal{I}$, then the impersonator will be successful in its impersonation attempt.

Evauation of $\mathcal{I}$'s advantage. We know that the forger $\mathcal{F}$ must perform an hash query $(\widehat{\textsc{Cmt}}, \widehat{m})$ among the $q_h(\lambda)$ hash queries it is allowed to perform during the training phase (according to the requirements listed at the beginning of the proof), therefore with probability

$$\mathbb{P}[\mathcal{I} \text{ guesses } fp \mid \mathcal{I} \text{ simulates}] = \frac{1}{q_h(\lambda)}$$

the impersonator guesses the right forge pointer fp. The probability is conditioned to the event that $\mathcal{I}$ (correctly) simulates the unforgeability experiment because otherwise the forge pointer might not be defined. We assumed that the forger $\mathcal{F}$ has non-negligible advantage in winning the real unforgeability experiment $\mathbb{P}[\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda) = 1] = \epsilon(\lambda)$. If $\mathcal{I}$ simulates the experiment $\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda)$, $\mathcal{F}$ wins the simulated experiment, while interacting with $\mathcal{I}$, with the same non-negligible probability

$$\mathbb{P}[\mathcal{F}^{\mathcal{I}} \text{ wins} \mid \mathcal{I} \text{ simulates}] = \mathbb{P}[\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda) = 1] = \epsilon(\lambda).$$

Finally we can find a lower bound to the probability of success of the impersonator $\mathcal{I}$ in playing the experiment $\text{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}$

$$\begin{aligned}
&\mathbb{P}[\text{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}} = 1] \geq \mathbb{P}[\mathcal{F}^{\mathcal{I}} \text{ wins} \wedge \mathcal{I} \text{ guesses } fp \wedge \mathcal{I} \text{ simulates}] = \\
&= \mathbb{P}[\mathcal{F}^{\mathcal{I}} \text{ wins} \wedge \mathcal{I} \text{ guesses } fp | \ \mathcal{I} \text{ simulates}] \cdot \mathbb{P}[\mathcal{I} \text{ simulates}] = \\
&= \mathbb{P}[\mathcal{F}^{\mathcal{I}} \text{ wins} \mid \mathcal{I} \text{ simulates}] \cdot \mathbb{P}[fp \text{ is guessed} \mid \mathcal{I} \text{ simulates}] \cdot \mathbb{P}[\mathcal{I} \text{ simulates}] \geq \\
&\geq \epsilon(\lambda) \frac{1}{q_h(\lambda)} \left(1 - \frac{q_s(\lambda)(q_h(\lambda) + q_s(\lambda) - 1)}{2^{\beta(\lambda)}}\right)
\end{aligned}$$

which is non-negligible in the security parameter λ.

Note that in the second equality we used the fact that fp is sampled uniformly at random by $\mathcal{I}$ before it starts interacting with $\mathcal{F}$ and the value of fp does not affect the simulation of the experiment with $\mathcal{F}$, and in the third equality we used the lower bound to the probability that $\mathcal{I}$ fails the simulation described in Equation 4.1.

Since we have designed an impersonator $\mathcal{I}$, using $\mathcal{F}$ as a subroutine, that has non-negligible advantage in winning the experiment $\text{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}(\lambda)$, where $\mathcal{TID}$ was assumed secure against impersonation under active attacks, this means that the algorithm $\mathcal{F}$, which has non-negligible advantage in winning the experiment $\text{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda)$, do not exist. Therefore the digital signature $\mathcal{TDS}$ is unforgeable under active attacks, and this concludes the proof. □

Observation 3 (Non concurrent security)**.** The assumption about the structure of $\mathsf{TP}_{\text{CMT}} = (\mathsf{TP}^1_{\text{CMT}}, \mathsf{TP}^2_{\text{CMT}})$ is essential to guarantee that the derived signature schemes are resistant against practical attacks on concurrent sessions as described in [8].

In principle one could avoid the "online" aspect of Definition 13. Indeed, the above simulation would be acceptable as long as $\mathcal{I}$ could rewind $\mathcal{F}$ after receiving CMT, in order to fix some inconsistencies in the hash table HT. While this would be fine for the standard notion of unforgeability this would cause major problems when proving the protocols secure under parallel composition.

Lemma 2. *[$\mathcal{TDS} \implies \mathcal{TID}$] Under the assumptions of Theorem 2, if $\mathcal{TDS}$ is unforgeable against active chosen-message attacks then $\mathcal{TID}$ is secure against impersonation under active attacks in the random oracle model.*

Proof Sketch. Let $\mathcal{I}$ be an impersonator which wins the experiment $\text{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}(\lambda)$ with non-negligible probability, then we build a forger $\mathcal{F}$ which uses $\mathcal{I}$ as a subroutine who wins the experiment $\text{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda)$ with non-neligible probability.

In this case, it is $\mathcal{F}$ who will simulate the identification oracle, by interacting with the real world oracles $\mathcal{O}^H_{\mathcal{TDS}}(\cdot)$ and $\mathcal{O}_H(\cdot)$ therefore the issues in simulating the random oracle as in Theorem 1 are not present anymore.

Initialization $\mathcal{F}$ interacts with $\mathcal{O}^H_{\mathcal{TDS}}(\cdot)$ and $\mathcal{O}_H(\cdot)$ who provides her with the public parameters pp and the public key of the n parties among which $t-1$ can be corrupted by $\mathcal{F}$. $\mathcal{F}$ simulates $\mathcal{O}_{\mathcal{TID}}(\cdot)$ and forwards this information to $\mathcal{I}$.

Training phase $\mathcal{I}$ selects the set of cor actors to corrupt and before each impersonation query it chooses the set of hon honest parties it wants to interact with. This information is sent to $\mathcal{F}$ who forwards it to $\mathcal{O}^H_{\mathcal{TDS}}(\cdot)$.

Whenever $\mathcal{I}$ makes an identification query, $\mathcal{F}$ sends to $\mathcal{O}^H_{\mathcal{TDS}}(\cdot)$ a sign query of a fresh new message $\mathtt{m}$ which gets increased for every different sign query.

The oracle sends the messages on behalf of the parties in hon and $\mathcal{F}$ forwards it to $\mathcal{I}$ correctly simulating the multiparty protocol TP_{CMT}. When it comes the time for $\mathcal{F}$ to send the challenge CH to $\mathcal{I}$, $\mathcal{F}$ queries $\mathcal{O}_H(\cdot)$ on $(\mathtt{m}||\text{CMT})$ and obtains CH which forwards to $\mathcal{I}$ as the challenge of the impersonation attempt. Since it is the first time that $\mathcal{F}$ queries the random oracle on $(\mathtt{m}||\text{CMT})$, since $\mathtt{m}$ is updated during every execution of the identification protocol, $\mathcal{F}$ correctly simulates the oracle $\mathcal{O}_{\mathcal{TID}}(\cdot)$ in sending a random challenge. Finally as with the protocol TP_{CMT}, $\mathcal{F}$ acts as a man in the middle in the execution of TP_{RSP} between $\mathcal{I}$ and $\mathcal{O}^H_{\mathcal{TDS}}(\cdot)$.

Simulation failure The simulation never fails because $\mathcal{F}$ always receives new random challenges from $\mathcal{O}_H$ since it provides $\mathcal{O}_H$ always with different inputs obtained by increasing $\mathtt{m}$ every time it performs a new sign query.

Exploit of $\mathcal{I}$'s impersonation When $\mathcal{I}$ produces its impersonation attempt, it sends CMT^* to $\mathcal{F}$ as if it were produced by executing TP_{CMT}. Then $\mathcal{F}$ starts preparing its forgery by sending to $\mathcal{O}_H(\cdot)$ a hash query with input $(\mathtt{m}^*||\text{CMT}^*)$ fresh new $\mathtt{m}^*$ that has never been used before and that will be the message that will be signed in the forgery. The oracle $\mathcal{O}_H$ returns to $\mathcal{F}$ the challenge CH^* that $\mathcal{F}$ sends to $\mathcal{I}$ correctly simulating the transcript oracle $\mathcal{O}_{\mathcal{TID}}(\cdot)$ in the generation of a random challenge.

Finally $\mathcal{I}$ concludes its impersonation by sending the response RSP^* that, if it is valid, allows $\mathcal{F}$ to produce a forgery of $\mathtt{m}$, namely $(\text{CMT}^*, \text{RSP}^*)$ which verifies since $H(\mathtt{m}^*||\text{CMT}^*) = \text{CH}^*$. □

The proofs of Lemma 1 and Lemma 2 prove Theorem 2.

Dealing with abort Our heuristic does not require an explicit behaviour to adopt when dealing with errors, such as a party refusing to sends data or failing some ZKPs. In general, a safe countermeasure would be to abort when any party deviate from the protocol. This can clearly lead to bias in the signature generation, but since we are limiting our analysis on the unforgeability property this is acceptable [15]. On the other hand, a strong selling point is that our heuristic does not intrinsically lower the security properties of the underlying identification protocols, and it is easy to see that if the starting protocols has identifiable abort then also the transformed protocol has this properties. In this sense, the obtained signature inherits all the properties regarding abort or robustness of the starting protocol, allowing to design more secure protocols.

5 Examples of Applications for Existing Signatures

In this section we show possible applications of our heuristic, using two existing threshold signatures as examples. In Section 5.1 we prove the security of GRASS [2], which is a group action based threshold signature. In Section 5.2 we analyse Sparkle [15], a threshold Schnorr signature: we provide an alternative proof of Theorem 1 of [15] ("Sparkle is statically secure under DL in the ROM") using our framework.

We recall the scheme of Sparkle in Appendix B, Figure 5, using the same notation used in their paper [15], and we also provide additional details about the associated threshold identification scheme.

5.1 GRASS

Let $\mathbb{G}$ be a multiplicative group, $\mathcal{X}$ be a set and $\star : \mathbb{G} \to \mathcal{X}$ be a regular group action hard to invert. The threshold sigma protocol that we present in Figure 2 is a threshold sigma protocol for relation $\mathcal{R} = \{(w, y = w \star x_0) | w \in \mathbb{G}, x_0 \in \mathcal{X}\} \subset \mathbb{G} \times \mathcal{X}$ and additive secret sharing SS. The witness w is split among all the parties, such that party i knows g_i, with $\prod_{i=1}^{n} g_i = w$. Moreover, each party has a public key $x_i = g_i \star x_{i-1}$. Since the challenge space is binary, the complete protocol is done by doing parallel repetition of the protocol presented in Figure 2, here we present only a single execution but all the arguments could be easily generalized to the complete protocol.

To prove their joint knowledge of w, the parties engage in a round robin protocol where P_i, on input $\tilde{x}_{i-1}$ published by previous party (or the common value x_0 in case of $i = 1$) picks a random $\tilde{g}_i \in \mathbb{G}$, computes $\tilde{x}_i = \tilde{g}_i \star \tilde{x}_{i-1}$ and outputs it. Moreover, they participate in a two step creation of random string r, by first sending $\mathsf{H}_{\mathsf{com}}(r_i)$ for a random r_i and then sending r_i. The common output of $\mathsf{TP}_{\textsc{Cmt}}$ is $\textsc{Cmt} = r || \tilde{x}_n$. On input a challenge bit $\textsc{Ch}$, the response works as follows: if $\textsc{Ch} = 0$ then each party reveals $\tilde{g}_i$ and $\textsc{Rsp} = \prod_{i=1}^{N} \tilde{g}_i$, if $\textsc{Ch} = 1$ then the parties engage in a round robin protocol where P_i, on input $\textsc{Rsp}_{i-1}$ published by previous party (or 1 in case of $i = 1$), publishes $\textsc{Rsp}_i = \tilde{g}_i \cdot \textsc{Rsp}_{i-1} \cdot g_i^{-1}$. The common response $\textsc{Rsp}$ is $\textsc{Rsp}_n$. To verify it, the verifier checks $\tilde{x}_n = \textsc{Rsp} \star x_0$ when $\textsc{Ch} = 0$ and $\tilde{x}_n = \textsc{Rsp} \star x_n$ otherwise.

Theorem 3. *If the group action $\star : \mathbb{G} \to \mathcal{X}$ is hard to invert, then the threshold signature scheme GRASS is unforgeable under active chosen message attacks.*

Proof. Our goal is to use Theorem 1, from which Theorem 2 follows immediately.

We start by proving that the threshold sigma protocol is special sound and then we prove the active honest-verifier zero-knowledge property.

Special soundness. Suppose to have two accepting transcripts $(\textsc{Cmt}, 0, \textsc{Rsp})$ and $(\textsc{Cmt}, 1, \textsc{Rsp}')$. Then it would be possible to compute the witness as $w = \textsc{Rsp} \cdot \textsc{Rsp}'^{-1}$.

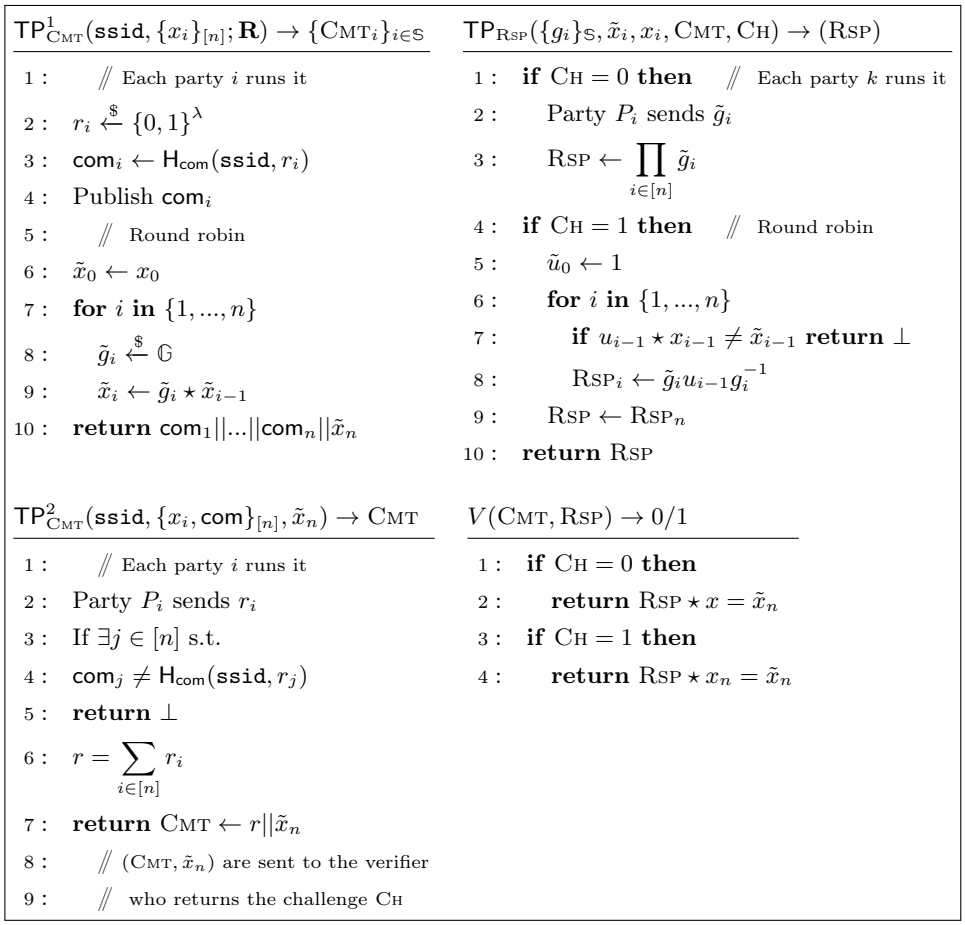

Fig. 2. Threshold sigma protocol for GRASS.

Active honest-verifier zero-knowledge. To prove that the protocol is AHVZK, we must show that it can be simulated by a simulator $\mathcal{S}$ taking in input (x_n, CH^*) and the $n-1$ shares of the private key controlled by the adversary. Let i be the honest party, controlled by the simulator $\mathcal{S}$. When $\text{CH}^* = 0$ the simulator follows the protocol normally, while when $\text{CH}^* = 1$ $\mathcal{S}$ picks a random $\tilde{g}_i$ and sends $\tilde{x}_i = \tilde{g}_i \star x_i$ in place of $\tilde{x}_i = \tilde{g}_i \star \tilde{x}_{i-1}$. It is immediate to see that in both cases the simulation is indistinguishable from a real execution and that, if the adversary act honestly, the resulting response verifies. Indeed, thanks to the check of line 7 of TP_{RSP}, the simulator is sure that all the previous parties P_j, $j = 1, ..., i-1$, in the round robin acted honestly, thus it can safely sends $\tilde{g}_i$, which verifies the check of line 7 of the next round.

It remains to prove that the above scheme has a secure TP_{CMT} as per Definition 13. At the end of $\mathsf{TP}^1_{\text{CMT}}$ the simulator knows all the $\tilde{x}_i$ and r_i of the

adversary, thanks to the random oracle queries. This allows the simulator to perfectly simulate $\mathsf{TP}^2_{\textsc{Cmt}}$, since it knows all the input. We can notice that as long as $\mathsf{H}_{\mathsf{com}}$ is binding, $\mathsf{TP}^2_{\textsc{Cmt}}$ is deterministic. It remains to prove that no adversary $\mathcal{F}$ can guess Cmt with non negligible probability before the execution of $\mathsf{TP}^2_{\textsc{Cmt}}$. This is thanks to the string r and the one way-property of $\mathsf{H}_{\mathsf{com}}$ (i.e. it is impossible to find x knowing only $\mathsf{H}_{\mathsf{com}}(x)$). Indeed, even if $\tilde{x}_n$ is fully known, the string r is uniformly distributed in $\{0,1\}^\lambda$ and the probability that $\mathcal{F}$ guesses it is 2^λ, due to the one-wayness of $\mathsf{H}_{\mathsf{com}}$. This ensure that the probability of $\mathcal{F}$ guessing Cmt is 2^λ, which is negligible.

It is immediate to see that the above arguments could be adapted when considering the parallel repetition of the protocol the protocol in Figure 2. By equipping it with the Setup and Key-Gen explained in [2] we obtain a threshold identification scheme $\mathcal{TID}$ which has a one-way Key-Gen, a super-polynomial challenge space and is special sound and zero-knowledge. Therefore by Theorem 1, $\mathcal{TID}$ is secure against active impersonation attacks. By applying Theorem 2 we proved that GRASS, the digital signature obtained by applying the distributed Fiat-Shamir transform, is unforgeable against active chosen message attacks. □

5.2 Sparkle

The threshold sigma protocol that we present in Figure 3 is a threshold sigma protocol for relation $\mathcal{R} = \{(w, y = g^w) | w \in \mathbb{Z}_p, \mathbb{G} =< g >\} \subset \mathbb{Z}_p \times \mathbb{G}$ and Shamir secret sharing SS, where $\mathbb{G}$ is a cyclic group with generator g of order p, a λ bits prime number. The challenge space is $C = \mathbb{Z}_p$ which is super-polynomial in the security parameter λ. This, equipped with the Setup and the one-way Key-Gen which is used in the threshold signature of Sparkle, will form the threshold identification scheme we use to prove Sparkle security.

Theorem 4. *If the discrete logarithm is hard in $\mathbb{G}$, then the threshold signature scheme Sparkle is unforgeable under active chosen message attacks.*

Proof. Our goal is to use Theorem 1, from which Theorem 2 follows immediately.

We start by proving that the threshold sigma protocol is special sound and then we prove the active honest-verifier zero-knowledge property.

Special soundness. The special soundness property is trivial and follows immediately from the special soundness of the standard Schnorr protocol [28]. Indeed, suppose to have two accepting transcripts $(R, \textsc{Ch}, z)$ and $(R, \textsc{Ch}', z')$ with $\textsc{Ch} \neq \textsc{Ch}'$. Then it would be possible to compute the discrete logarithm of $\mathsf{pk} = y$ by simply computing $w = (z - z')(\textsc{Ch} - \textsc{Ch}')^{-1}$.

Active honest-verifier zero-knowledge. To prove that the protocol is AHVZK, we must show that it can be simulated by a simulator $\mathcal{S}$ taking in input $(y = g^w, \textsc{Ch}^*)$ and the $t-1$ shares of the private key controlled by the adversary. Without loss of generality we can say that $\mathbb{S} = [t]$, the adversary controls $P_1, \ldots, P_{t-1}$ and $w_1, \ldots, w_{t-1}$ are their shares of the witness which

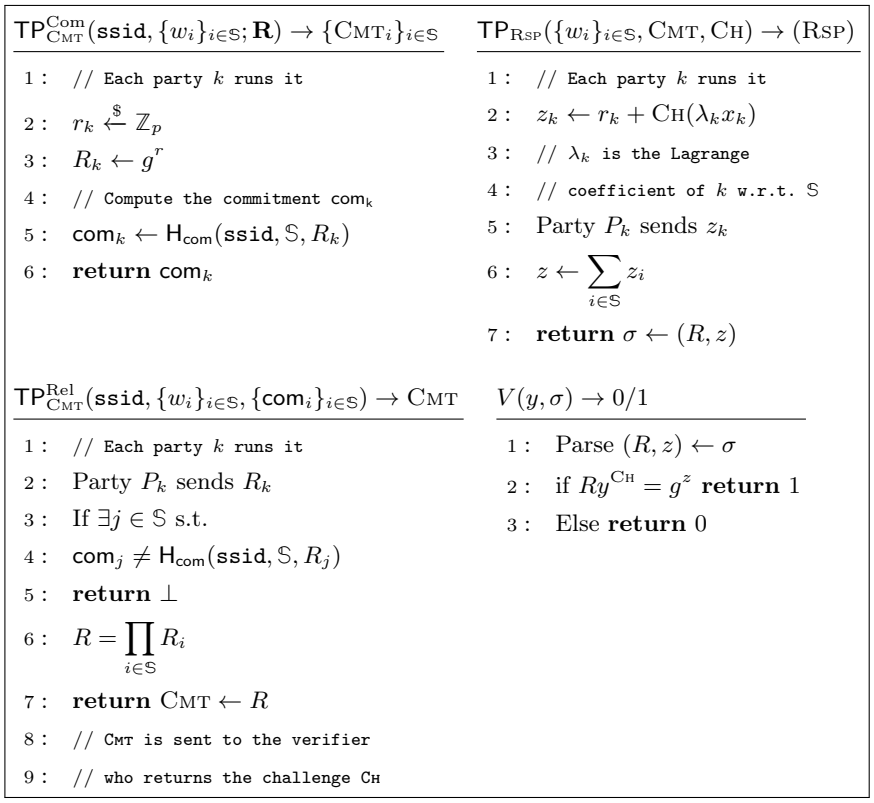

Fig. 3. Threshold sigma protocol for Sparkle.

are given also to the simulator $\mathcal{S}$ who must impersonate P_t without knowing w_t.
The simulation resembles the simulation of the centralized sigma protocol. The simulator $\mathcal{S}$ samples uniformly at random $z_t \in \mathbb{Z}_p$ and defines

$$R_t = g^{z_t} y^{-\mathrm{CH}^*} \prod_{j=1}^{t-1} g^{\lambda_j w_j \mathrm{CH}^*},$$

where CH^* is the challenge it received in input and λ_j is the Lagrange coefficient of j with respect to $\mathbb{S}$.
Note that, even if $\mathcal{S}$ does not know w_t, by definition of Shamir secret sharing $w = \sum_{i\in[t]} \lambda_i w_i$ and $y^{-\mathrm{CH}^*} = g^{w(-\mathrm{CH}^*)}$, therefore $y^{-\mathrm{CH}^*} \prod_{j=1}^{t-1} g^{\lambda_j w_j \mathrm{CH}^*} = g^{\lambda_t w_t (-\mathrm{CH}^*)}$, then it holds that $g^{z_t} = R_t g^{\lambda_t w_t \mathrm{CH}^*}$.
This means that the transcript $(R_t, \mathrm{CH}^*, z_t)$ is valid and, being z_t sampled uniformly at random, and R_t being univocally determined from (z_t, CH^*), $(R_t, \mathrm{CH}^*, z_t)$ is indistinguishable from an honest transcript (generated starting from R_t).

Finally $\mathcal{S}$ executes $\mathsf{TP}^{\text{Com}}_{\text{CMT}}$ computing $\mathsf{com}_t = \mathsf{H}_{\mathsf{com}}(\mathtt{m}, \mathcal{S}, R_t)$, then it executes $\mathsf{TP}^{\text{Rel}}_{\text{CMT}}$ by releasing R_t. The commitments are aggregated computing R, then the challenge CH^* will be used as the challenge of the transcript and $\mathcal{S}$ simulates the algorithm TP_{RSP} by broadcasting the responses $\text{RSP}_t = z_t$ it sampled randomly at the beginning of the simulation.
Note that the transcripts (R, CH^*, z), together with the transcript generated by the messages sent by $\mathcal{S}$, form an accepting transcript as long as the other parties in $\mathbb{S}$ act compute their responses correctly. Also, the transcripts of $\mathcal{S}$ are indistinguishable from a real execution since the messages that $\mathcal{S}$ must send are independent of the messages sent by the adversary who could be potentially malicious. Therefore the sigma protocol is active zero-knowledge according to Definition 9.

By equipping the threshold sigma protocol with the Setup and Key-Gen of Sparkle we obtain a threshold identification scheme $\mathcal{TID}$ which has a one-way Key-Gen, a super-polynomial challenge space and is special sound, active honest-verifier zero-knowledge. Therefore by Theorem 1, $\mathcal{TID}$ is secure against active impersonation attacks.

It remains to prove that $\mathcal{TID}$ has an extractable TP_{CMT} as per Definition 13. Indeed, at the end of the execution of $\mathsf{TP}^1_{\text{CMT}}$, the simulator knows R_i from the random oracle query made by the adversary and it can use them to simulate an execution of $\mathsf{TP}^2_{\text{CMT}}$. The simulation is perfect, since the simulator knows all the input of $\mathsf{TP}^2_{\text{CMT}}$. Notice that as long as $\mathsf{H}_{\mathsf{com}}$ is binding the protocol $\mathsf{TP}^2_{\text{CMT}}$ is deterministic and, as long as $\mathsf{H}_{\mathsf{com}}$ is one-way the final output R has high min entropy and the probability of an adversary being able to compute it before the execution of $\mathsf{TP}^2_{\text{CMT}}$ is negligible. Indeed, since $R = \prod_{i \in \mathbb{S}} R_i$ where the computations are executed in $\mathbb{G}$, if at least one party in $\mathbb{S}$ is honest, the value R will be uniformly distributed in $\mathbb{G}$. We can notice that TP_{CMT} is commit-release, as per Observation 1.

By applying Theorem 2 we prove that Sparkle, the digital signature obtained by applying the distributed Fiat-Shamir transform, is unforgeable against active chosen message attacks. □

6 Conclusions

Although threshold signature schemes have been known for a while and are more popular than ever, the concept of threshold identification scheme received very little attention. In particular, previous works focus their attention to protocols that do not allow communication between prover, either relying on some pre-computation or on the presence of a trusted third party (the combiner).

In our work we propose a new definition for threshold identification schemes, with the aim of capturing the multi-party nature of it. We model our definition to mimic the traditional structure of threshold signature schemes, in order to draw a link between the two worlds, thanks to a generalized version of the Fiat-Shamir Transform.

Following the footprint of M. Abdalla et al. in [1], we show the relation that links the security of a threshold identification protocol and the security of the threshold signature schemes derived by applying the distributed Fiat-Shamir Transform.

Finally, we move our attention to threshold sigma protocols and their link with threshold identification schemes. Similarly to the centralized case, we define properties of the sigma protocols that, if satisfied, guarantee that the associated identification schemes are secure. This provides a viable way to prove a threshold digital signature unforgeable as we show for Sparkle in Section 5.

Future works. Our approach could streamline the security analysis of many threshold signatures, however it covers only static corruptions, where the adversary decide which party to corrupt at the beginning of the protocol. While this is a relevant security notion, often used as in [4,24,14], many protocols are also proved secure in the adaptive case, where the adversary can, at any time, corrupt parties and learn their state [15]. It would be interesting to extend our analysis to the adaptive case. The structure of the proof of Theorem 2 suggests that if a threshold identification scheme is secure against adaptive adversaries (this can be done by adding an additional oracle $\mathcal{O}_{\texttt{Corrupt}}$ that can be adaptively called to learn honest parties input) also the derived threshold signature scheme is secure against adaptive attacks. In this case, the real challenge would be to define properties on the threshold sigma protocol, in the same vein of the zero knowledge properties, to achieve the adaptive security of the threshold identification scheme.

It would be also interesting to strengthen our security models and prove it in the UC framework, taking also in consideration the distribution of the signature and not only the unforgeability property.

Finally the results we prove in this paper should pave the way for the definition and design of threshold NIZKP, by applying the distributed Fiat-Shamir Transform to threshold sigma protocols.

7 Acknowledgments

The first author is member of GNSAGA of INdAM and 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 acknowledges support from Eustema S.p.A. through the PhD scholarship.

The authors acknowledge support from Ripple's University Blockchain Research Initiative.

We thank Elisa Trento for the interesting and deep discussions during her master thesis work.

We thank Lawrence Roy for the helpful comments about the UC model.

References

1. Abdalla, M., An, J.H., Bellare, M., Namprempre, C.: From identification to signatures via the fiat-shamir transform: Minimizing assumptions for security and forward-security. In: Knudsen, L.R. (ed.) Advances in Cryptology — EUROCRYPT 2002. pp. 418–433. Springer Berlin Heidelberg, Berlin, Heidelberg (2002)
2. Battagliola, M., Borin, G., Meneghetti, A., Persichetti, E.: Cutting the grass: Threshold group action signature schemes. Cryptology ePrint Archive, Paper 2023/859 (2023), https://eprint.iacr.org/2023/859, https://eprint.iacr.org/2023/859
3. Battagliola, M., Longo, R., Meneghetti, A., Sala, M.: Threshold ECDSA with an Offline Recovery Party. Mediterranean Journal of Mathematics **19**(1), 1–29 (2022)
4. Battagliola, M., Longo, R., Meneghetti, A., Sala, M.: Provably unforgeable threshold eddsa with an offline participant and trustless setup. Mediterranean Journal of Mathematics **20**(5), 253 (2023)
5. Baum, C., Jadoul, R., Orsini, E., Scholl, P., Smart, N.P.: Feta: Efficient threshold designated-verifier zero-knowledge proofs. In: Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security. p. 293–306. CCS '22, Association for Computing Machinery, New York, NY, USA (2022). https://doi.org/10.1145/3548606.3559354, https://doi.org/10.1145/3548606.3559354
6. Bellare, M., Neven, G.: Multi-signatures in the plain public-key model and a general forking lemma. In: Proceedings of the 13th ACM Conference on Computer and Communications Security. p. 390–399. CCS '06, Association for Computing Machinery, New York, NY, USA (2006). https://doi.org/10.1145/1180405.1180453, https://doi.org/10.1145/1180405.1180453
7. Ben-Or, M., Goldwasser, S., Kilian, J., Wigderson, A.: Multi-prover interactive proofs: How to remove intractability assumptions. In: Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing. p. 113–131. STOC '88, Association for Computing Machinery, New York, NY, USA (1988). https://doi.org/10.1145/62212.62223, https://doi.org/10.1145/62212.62223
8. Benhamouda, F., Lepoint, T., Loss, J., Orrù, M., Raykova, M.: On the (in) security of ros. Journal of Cryptology **35**(4), 25 (2022)
9. Boneh, D., Shoup, V.: A graduate course in applied cryptography. Draft 0.6 (2023)
10. Brandão, L.T.A.N., Davidson, M.: Notes on threshold eddsa/schnorr signatures, https://csrc.nist.gov/publications/detail/nistir/8214b/draft, accessed: 2023-05-01
11. Brandão, L.T.A.N., Davidson, M., Vassilev, A.: Nist roadmap toward criteria for threshold schemes for cryptographic primitives, https://nvlpubs.nist.gov/nistpubs/ir/2020/NIST.IR.8214A.pdf, accessed: 2020-08-27
12. Brandão, L.T.A.N., Peralta, R.: Nist first call for multi-party threshold schemes, https://nvlpubs.nist.gov/nistpubs/ir/2023/NIST.IR.8214C.ipd.pdf, accessed: 2020-08-27
13. Canetti, R.: Universally composable security: A new paradigm for cryptographic protocols. In: Proceedings 42nd IEEE Symposium on Foundations of Computer Science. pp. 136–145. IEEE (2001)
14. Chu, H., Gerhart, P., Ruffing, T., Schröder, D.: Practical schnorr threshold signatures without the algebraic group model. In: Handschuh, H., Lysyanskaya, A. (eds.) Advances in Cryptology – CRYPTO 2023. pp. 743–773. Springer Nature Switzerland, Cham (2023)

15. Crites, E., Komlo, C., Maller, M.: Fully adaptive schnorr threshold signatures. Cryptology ePrint Archive (2023)
16. Damgård, I.: On σ-protocols. Lecture Notes, University of Aarhus, Department for Computer Science **84** (2002)
17. Desmedt, Y., Di Crescenzo, G., Burmester, M.: Multiplicative non-abelian sharing schemes and their application to threshold cryptography. In: Pieprzyk, J., Safavi-Naini, R. (eds.) Advances in Cryptology — ASIACRYPT'94. pp. 19–32. Springer Berlin Heidelberg, Berlin, Heidelberg (1995)
18. Fiat, A., Shamir, A.: How to prove yourself: Practical solutions to identification and signature problems. In: Odlyzko, A.M. (ed.) Advances in Cryptology — CRYPTO' 86. pp. 186–194. Springer Berlin Heidelberg, Berlin, Heidelberg (1987)
19. Gennaro, R., Goldfeder, S.: Fast multiparty threshold ecdsa with fast trustless setup. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. p. 1179–1194. CCS '18, Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3243734.3243859, https://doi.org/10.1145/3243734.3243859
20. Gennaro, R., Goldfeder, S.: One round threshold ecdsa with identifiable abort. Cryptology ePrint Archive (2020)
21. Keller, M., Mikkelsen, G.L., Rupp, A.: Efficient threshold zero-knowledge with applications to user-centric protocols. In: Smith, A. (ed.) Information Theoretic Security. pp. 147–166. Springer Berlin Heidelberg, Berlin, Heidelberg (2012)
22. Komlo, C.: A note on various forking lemmas, https://www.chelseakomlo.com/assets/content/notes/Forking-Lemma-Variants.pdf
23. Libert, B., Joye, M., Yung, M.: Born and raised distributively: Fully distributed non-interactive adaptively-secure threshold signatures with short shares. Theoretical Computer Science **645**, 1–24 (2016). https://doi.org/https://doi.org/10.1016/j.tcs.2016.02.031, https://www.sciencedirect.com/science/article/pii/S0304397516001626
24. Lindell, Y.: Simple three-round multiparty schnorr signing with full simulatability. Cryptology ePrint Archive, Paper 2022/374 (2022), https://eprint.iacr.org/2022/374, https://eprint.iacr.org/2022/374
25. Pedersen, T.P.: Distributed provers with applications to undeniable signatures. In: Davies, D.W. (ed.) Advances in Cryptology — EUROCRYPT '91. pp. 221–242. Springer Berlin Heidelberg, Berlin, Heidelberg (1991)
26. Pointcheval, D., Stern, J.: Security proofs for signature schemes. In: Maurer, U. (ed.) Advances in Cryptology — EUROCRYPT '96. pp. 387–398. Springer Berlin Heidelberg, Berlin, Heidelberg (1996)
27. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of cryptology **13**, 361–396 (2000)
28. Schnorr, C.: Efficient signature generation by smart cards. Journal of Cryptology **4**, 161–174 (01 1991). https://doi.org/10.1007/BF00196725

A Reduction of Lemma 1

In this section we provide an overview of the reduction described in the proof of Lemma 1.

In Figure 4 we represent the impersonator $\mathcal{I}$ who executes the experiment $\text{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}$. $\mathcal{I}$ interacts with a challenger $\mathcal{C}_{\mathcal{TID}}$ who initialises the experiment and which will provide the challenge $\textsc{Ch}^*$ to $\mathcal{I}$ during the impersonation attempt, and with a transcript oracle $O_{\mathcal{TID}}$ which answers the threshold identification queries and takes part together with I to the creation of the transcripts generated during the training phase.

The impersonator $\mathcal{I}$ runs the forger $\mathcal{F}$ as a subroutine and simulates the experiment $\text{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}$, therefore it must simulate the challenger $\mathcal{C}_{\mathcal{TDS}}$, the random oracle $\mathcal{O}_H$ and the signature generation oracle $\mathcal{O}_{\mathcal{TDS}}$ which are represented with the bar $\bar{\mathcal{O}}$ to recall that $\mathcal{I}$ simulates the oracles.

The simulation comprises four parts, each of them denoted by a different enumerating system. Namely

Numbers (1)-(6): the initialization of the security game of $\mathcal{TID}$. $\mathcal{I}$ uses the same data in the initialization of $\mathcal{TDS}$ for $\mathcal{F}$. This allows $\mathcal{I}$ to correctly simulate $\mathcal{C}_{\mathcal{TDS}}$. Notice that the parties $\mathcal{I}$ corrupts are the same parties chosen by $\mathcal{F}$.

Lower case letters (a)-(o): the simulation of the sign queries made to $\mathcal{O}_{\mathcal{TDS}}$ by $\mathcal{F}$. In particular $\mathcal{F}$ sends to $\mathcal{I}$ a sign query for $\mathtt{m}$, asking for the cooperation of the parties in J_h. $\mathcal{I}$, to simulate the sign oracle, starts an interaction with $\mathcal{O}_{\mathcal{TID}}$ asking for the same J_h. $\mathcal{F}$ forwards the messages received by $\mathcal{O}_{\mathcal{TID}}$ to $\mathcal{F}$ (steps (c-d) and (g-h)) and vice versa (steps (e-f) and (i-j)). In step (k), when $\mathcal{I}$ receives the challenge $\textsc{Ch}$ from $\mathcal{O}_{\mathcal{TID}}$, it updates the hash table setting $\mathsf{HT}[\mathtt{m}||\textsc{Cmt}] = \textsc{Ch}$. Finally $\mathcal{I}$ carries out the whole signing protocol with the support of $\mathcal{O}_{\mathcal{TID}}$.

Greek letters $(\alpha) - (\beta)$: $\mathcal{F}$ sends an hash query for x to $\mathcal{I}$ (who simulates $\mathcal{O}_H$) and $\mathcal{I}$ answers with $\mathsf{HT}[x]$ if it is defined, otherwise it samples a random digest and updates the hash table. When $\mathcal{I}$ receives the fp-th hash query, it parses $x = \mathtt{m}^*||\textsc{Cmt}^*$ and starts the impersonation attempt sending $\textsc{Cmt}^*$ (step (A)) to $\mathcal{C}_{\mathcal{TID}}$, who answers with a challenge $\textsc{Ch}^*$ (step (B)). Finally $\mathcal{I}$ sets $\mathsf{HT}[x] = \textsc{Ch}^*$.

Upper case letters (A)-(D): $\mathcal{I}$ starts its impersonator attempt during the fp-th hash query of $\mathcal{F}$ (step (A) and (B)). After a polynomial number of hash queries and sign queries the forger $\mathcal{F}$ outputs its forgery $(\widehat{\textsc{Cmt}}, \widehat{\textsc{Ch}}, \widehat{\textsc{Rsp}})$ (step (C)). At this point $\mathcal{I}$ uses it in its impersonator attempt. In particular $\mathcal{I}$ sends $\widehat{\textsc{Rsp}}$ to $\mathcal{C}_{\mathcal{TID}}$ (step (D)) as the response.

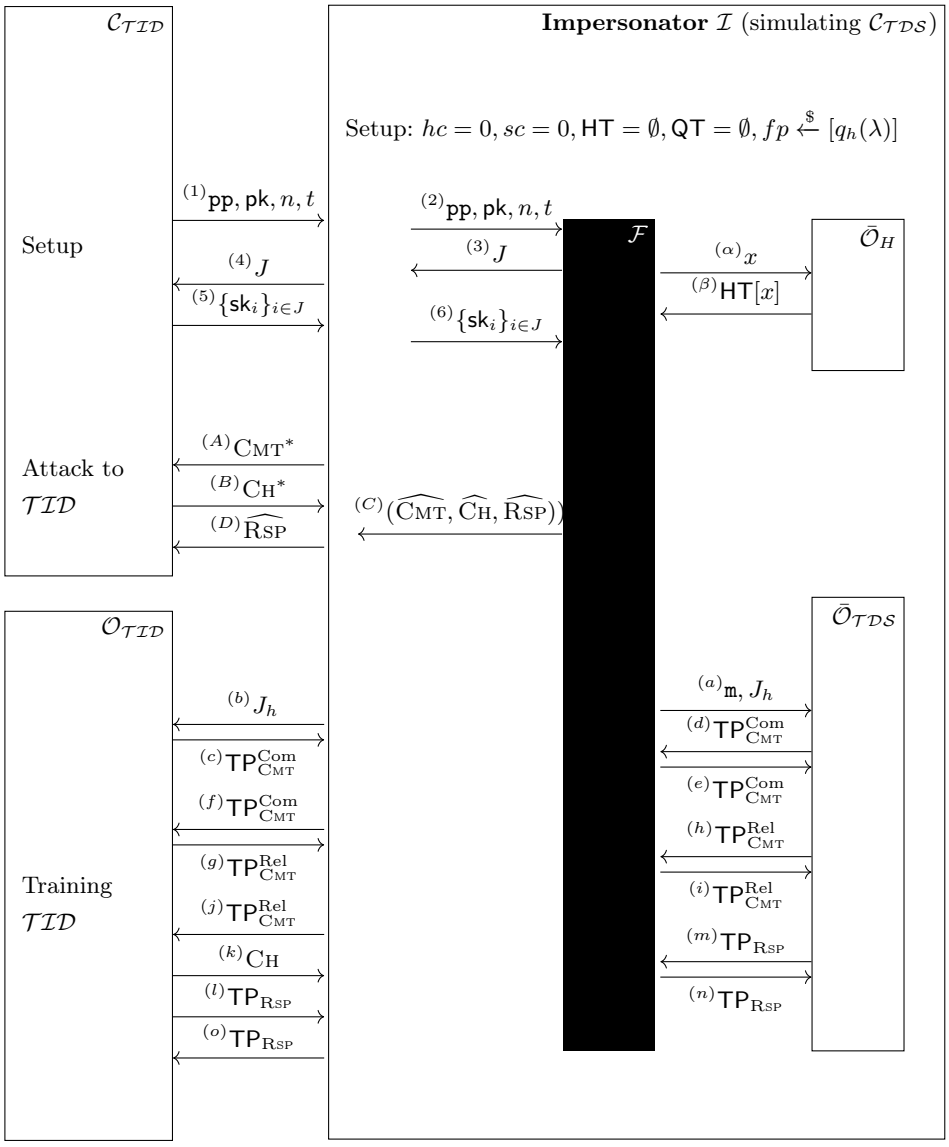

Fig. 4. High level description of the impersonator $\mathcal{I}$ using a forger $\mathcal{F}$ as a subroutine.

B Sparkle signature scheme

Below we provide the description of the Schnorr threshold signature Sparkle using the same notation used in [15].

$\underline{\mathsf{Setup}(\lambda) \to \mathsf{pp}}$

$(\mathbb{G}, p, g) \xleftarrow{\$} \mathtt{GrGen}(\lambda)$
$\mathsf{H}_{\mathsf{com}}, \mathsf{H}_{\mathsf{sig}} \xleftarrow{\$} \{\mathsf{H} : \{0,1\}^* \to \mathbb{Z}_p\}$
$\mathsf{pp} \leftarrow (p, \mathbb{G}, g, \mathsf{H}_{\mathsf{com}}, \mathsf{H}_{\mathsf{sig}})$

$\underline{\mathsf{Key\text{-}Gen}(n, t, \mathsf{pp}) \to (X, \{X_i\}_{i\in[n]}, \{x_i\}_{i\in[n]})}$

$x \xleftarrow{\$} \mathbb{Z}_p, X \leftarrow g^x$
$\{j, x_j\}_{j\in[n]} \leftarrow \mathtt{IssueShares}(x, n, t)$
for $j \in [n]$ do:
  $X_j \leftarrow g^{x_j}$
return $(X, \{X_j, x_j\}_{j\in[n]})$

$\underline{\mathsf{TSign}_1(\mathtt{m}, \mathbb{S}) \to (\mathsf{state}_k, c_k)}$

$r_k \xleftarrow{\$} \mathbb{Z}_p$
$R_k \leftarrow g^r$
$c_k \leftarrow \mathsf{H}_{\mathsf{com}}(\mathtt{m}, S, R_k)$
$\mathsf{state}_k \leftarrow (c_k, R_k, r_k, \mathtt{m}, \mathbb{S})$
return (state_k, c_k)

$\underline{\mathsf{TSign}_2(\mathsf{state}_k, \{c^i\}_{i\in\mathbb{S}}) \to (\mathsf{state}_k, R_k)}$

parse $(c_k, R_k, r_k, \mathtt{m}, \mathbb{S}) \leftarrow \mathsf{state}_k$
return $\perp$ if $c_k \notin \{c_i\}_{i\in\mathbb{S}}$
$\mathsf{state}_k \leftarrow (c_k, R_k, r_k, \mathtt{m}, \mathbb{S}, \{c_i\}_{i\in\mathbb{S}})$
return (state_k, R_k)

$\underline{\mathsf{TSign}_3(\mathsf{state}_k, x_k, \{R^i\}_{i\in\mathbb{S}}) \to (\mathsf{state}_k, R_k)}$

parse
$(c_k, R_k, r_k, \mathtt{m}, \mathbb{S}, \{c_i\}_{i\in\mathbb{S}}) \leftarrow \mathsf{state}_k$
return $\perp$ if $R_k \notin \{R_i\}_{i\in\mathbb{S}}$
for $i \in \mathbb{S}$ do:
  return $\perp$ if $c_i \neq \mathsf{H}_{\mathsf{com}}(\mathtt{m}, \mathbb{S}, R_i)$
$R \leftarrow \prod_{i\in\mathbb{S}} R_i$
$c \leftarrow \mathsf{H}_{\mathsf{sig}}(X, \mathtt{m}, R)$
$z_k \leftarrow r_k + c(\lambda_k x_k)$
return z_k

$\underline{\mathsf{Combine}(\{R_i\}_{i\in\mathbb{S}}, \{z_i\}_{i\in\mathbb{S}}) \to (\mathtt{m}, \sigma)}$

$R \leftarrow \prod_{i\in\mathbb{S}} R_i, z \leftarrow \sum_{i\in\mathbb{S}} z_k.$
return $\sigma \leftarrow (R, z)$

$\underline{\mathsf{Ver}(X, \mathtt{m}, \sigma) \to 0/1}$

parse $(R, z) \leftarrow \sigma$
$c \leftarrow \mathsf{H}_{\mathsf{sig}}(X, m, R)$
if $RX^c = g^z$ return 1
else return 0

Fig. 5. Sparkle Signature Scheme

In the $\mathcal{TID}$ of Figure 3 we avoided to explicitly write state. Moreover:

- TSign_1 is the same of $\mathsf{TP}^1_{\mathrm{CMT}}$ where the session id `ssid` is replaced by `m`.
- during TSign_2 each party checks the received data and outputs its partial commitment R_k. This is the same as the first line of $\mathsf{TP}^1_{\mathrm{CMT}}$, where the check are omitted for the sake of readability.
- in TSign_3 each party checks the consistency of each CMT_i, computes the joint commitment $\mathrm{CMT} = R$, computes the challenge and the partial signature z_k.

This is the same as the second part of $\mathsf{TP}^1_{\text{CMT}}$ as well as the first two line of TP_{RSP}.
- In Combine each party combines all the partial signatures to obtain the final signature. These are the last two lines of TP_{RSP}.

C Forking Lemma

First introduced by Pointcheval and Stern [26], the forking lemma is commonly used in proofs of security that require rewinding an adversary.

Let $\mathcal{A}$ be an adversary initialized with a random tape and having access to a random oracle (modeled by an hash function). While the behavior of the adversary is generally not defined, the adversary outputs some value that will either satisfy some pre-defined conditions (thus winning the security game), or not satisfy these conditions. If $\mathcal{A}$ completes its attack successfully, the forking lemma gives a lower bound for the probability that $\mathcal{A}$ wins again the security game in a second execution with the same random tape but with different outputs from the random oracle [22]. More formally we have the following lemma, by M. Bellare and G. Neven in [6]:

Lemma 3 (General Forking Lemma). *Let $q \in \mathbb{Z}$ with $q \geq 1$, H be a set with $|H| \geq 2$. Let IG be a randomized algorithm called input generator and let $\mathcal{A}$ be a randomized algorithm that, on input $x \xleftarrow{\$} \mathsf{IG}, h_1, ..., h_q \in H$, returns a pair (J, σ) with J being an integer $0 \leq J \leq q$ and σ a side output. The accepting probability p of $\mathcal{A}$, is defined as the probability that $J \geq 1$ in the experiment*

$$x \xleftarrow{\$} \mathsf{IG}; h_1, ..., h_q \xleftarrow{\$} H; (J, \sigma) \xleftarrow{\$} \mathcal{A}(x, h_1, ..., h_q)$$

The forking algorithm $\mathsf{F}_{\mathcal{A}}$ associated to $\mathcal{A}$ is the randomized algorithm that takes as input x and proceeds as follows:

$\mathsf{F}_{\mathcal{A}}(x)$:
$R \xleftarrow{\$} \{0,1\}^*$
$h_1, ..., h_q \xleftarrow{\$} H$
$(J, \sigma) \xleftarrow{\$} \mathcal{A}(x, h_1, ..., h_q; R)$
If $J = 0$ *Return*$(0, \epsilon, \epsilon)$
$h'_J, ..., h'_q \xleftarrow{\$} H$
$(J', \sigma') \xleftarrow{\$} \mathcal{A}(x, h_1, ..., h_{J-1}, h'_J, ..., h'_q; R)$
If $(J = J' \wedge h_J \neq h'_J)$ *Return*$(1, \sigma, \sigma')$
Else Return $(0, \epsilon, \epsilon)$

Then we have

$$\mathbb{P}[b = 1 | x \xleftarrow{\$} \mathsf{IG}; (b, \sigma, \sigma') \xleftarrow{\$} \mathsf{F}_{\mathcal{A}}(x)] \geq p\left(\frac{p}{q} - \frac{1}{|H|}\right).$$