# Distributed Fiat-Shamir Transform

Michele Battagliola$^{1[0000-0002-8269-2148]}$ and Andrea Flamini$^{1[0000-0002-3872-7251]}$

Department of Mathematics, University of Trento, Via Sommarive 14, Trento, Italy
`{michele.battagliola, 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. In this work, we translate the Fiat-Shamir Transform into a multi-party setting, building a framework that seeks to be an alternative, easier way to design threshold digital signatures. We do that by introducing the concept of *threshold identification scheme* and *threshold sigma protocol*, and showing necessary and sufficient conditions to prove the security of the threshold signature schemes derived from them.
Lastly, we show a practical application of our framework providing an alternative security proof for Sparkle, a recent threshold Schnorr signature. In particular, we consider the threshold identification scheme underlying Sparkle and prove the security of the signature derived from it.
We show that using our framework the effort required to prove the security of threshold signatures might be drastically lowered. In fact, instead of reducing explicitly its security to the security of a hard problem, it is enough to prove some properties of the underlying threshold sigma protocol and threshold identification scheme. Then, by applying the results that we prove in this paper it is guaranteed that the derived threshold signature is secure.

**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 [2,3,21,18], 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.

*Organization* In Section 2 we define the cryptographic preliminaries needed in our paper. Next, in Section 3 we define threshold identification schemes and a threshold variant of the Fiat-Shamir Transform. Section 4 and Section 5 contain the core of our work: first we show necessary security properties of the threshold identification scheme to obtain secure threshold signatures, next we consider the relations between identification protocols and sigma protocols to provide easier to prove sufficient conditions. Lastly, in Section 6 we show a possible application of our paradigm providing an alternative security proof for Sparkle. Finally, in Section 7 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 litterature: 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 [19], who maintained the setting of no communication between the provers, and by T. P. Pedersen in [27], 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 [23] 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 [4] 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 [4], 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 [7,19], we allow for communication between the prover and we do not rely on the presence of a combiner handling the communication with the verifier, like in [23]. Instead we focus our attention to protocols where provers communicate and jointly produce the proofs.

We then show how our definitions can lead to secure digital signature schemes. In particular, miming the approach proposed by Abdalla et al. in [1], we show of how it is possible to apply a distributed version of the Fiat-Shamir Transform [20] to obtain unforgeable threshold digital signatures. Moreover, we also show sufficient conditions that the starting distributed identification protocol must satisfy to guarantee that the obtained signature is secure according to the standard definitions of unforgeability under chosen message attack.

*Security Models* Introduced by R. Canetti in [13], Universal Composability (UC) is a widely used framework for the design and analysis of protocols due to the very strong security guarantees it provides. In particular, a protocol that is UC secure maintains its security properties when run together with other protocols and allows for both parallel and sequential composition. With regards of threshold digital signature, different UC security definitions are used, in particular we can distinguish a stronger definition, that essentially states that a threshold signature is a UC secure MPC protocol that outputs a signature [25]. This means that the distribution of output signatures must be the same as the distribution output by the centralized (non-threshold) signing algorithm. On the other hand, a weaker definition is often used, designing a threshold signature functionality that models both signing and verification. There is no requirement that the threshold signature algorithm should produce the same distribution as the centralized one. Instead, it is only required not to allow forgeries [15], in the same way as in the centralized definition [14].

Since in this paper we tackle for the first time the problem of adapting the Fiat-Shamir Transform to a distributed setting, which already requires defining several new cryptographic protocols, we favour a more straightforward approach both in terms of security definitions and security proofs. In particular our security analysis is game based, as the ones in [1,6], and provides security guarantees about a specific property, namely the unforgeability.

It is worth noticing that, under particular hypothesis, the proof in Section 4.1 do not require any rewinding, which suggests that it should be possible to prove our approach secure in the UC setting of [15]. Proving our approach secure in the stronger version would require more work and a completely different approach.

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

With $y \leftarrow A(x_1, x_2, \ldots)$ we refer to a deterministic algorithm $A$ taking in input the values $x_1, x_2, \ldots$ and returning the value $y$; if the input of some algorithm is clear from the context we might write $A(\cdot)$ instead of $A(x_1, x_2, \ldots)$ 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 $A(x_1, x_2, \ldots)$ is a probabilistic algorithm then we can use two notations for assigning to a variable $y$ the output of $A(x_1, x_2, \ldots)$: $y \overset{\$}{\leftarrow} A(x_1, x_2, \ldots)$ where the symbol $\overset{\$}{\leftarrow}$ emphasizes the probabilistic nature of the algorithm $A(x_1, x_2, \ldots)$ or $y \leftarrow A(x_1, x_2, \ldots; R)$, where $R \overset{\$}{\leftarrow} \mathsf{Coins}(\lambda)$ is drawn from the set of random coins $\mathsf{Coins}(\lambda)$, namely the set of bit strings of appropriate length which guarantees $\lambda$ bits of randomness. If the randomness $R$ is given in input to $A(x_1, x_2, \ldots)$, the output $y$ is uniquely determined.

Algorithms that start with the the letters $\mathsf{T}$ 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 $\mathsf{TSign}$ 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 $x_1, x_2 \ldots, x_n$ as $x_1||x_2|| \ldots ||x_n$. 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] 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 a *commitment* $\textsc{Cmt} \in \mathcal{X}$ to the verifier.
2. Then verifier returns a *challenge* $\textsc{Ch}$ consisting of a random string of fixed length $c(\lambda)$ which depends on the security parameter $\lambda$.
3. Lastly, the prover provides a *response* $\textsc{Rsp}$ and the verifier verifies it according to $y, \textsc{Cmt}, \textsc{Ch}$ and $\textsc{Rsp}$.

At the end of the interaction we want that an honest prover, who knows a witness $w$ for $y$, is able to convince the verifier with overwhelming probability (completness), a dishonest prover is not able to convince a verifier (soundness) and that the verifier does not learn anything more about the witness $w$ from the interaction with the prover besides that $(w, y) \in \mathcal{R}$ (zero-knowledge).

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 identification schemes. Informally speaking, we can imagine an identification scheme as a sigma protocol equipped with a secure key generation algorithm for the relation $\mathcal{R}$ that provides in a secure way the couple $(w, y) \in \mathcal{R}$ to the prover. In the context of identification protocols we say that $w$ is the secret key, denoted by $\mathsf{sk}$, while $y$ is the public key, denoted by $\mathsf{pk}$.

Formally we have the following definitions:

**Definition 1** (One-way key generation)**.** Let $\mathsf{Key\text{-}Gen}$ be a key generation algorithm for a relation $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{Y}$. Define the following experiment

```
$\mathrm{Exp}^{\text{One-way}}_{\mathsf{Key\text{-}Gen},\mathcal{A}}(\lambda)$ :
1 :  $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Key\text{-}Gen}(\lambda)$
2 :  $\mathsf{sk}' \xleftarrow{\$} \mathcal{A}(\mathsf{pk})$
3 :  $\textbf{return } (\mathsf{sk}', \mathsf{pk}) \in \mathcal{R}$
```

Define the advantage of $\mathcal{A}$ as $\mathrm{Adv}^{\text{One-way}}_{\mathsf{Key\text{-}Gen},\mathcal{A}}(\lambda) = \mathbb{P}(\mathrm{Exp}^{\text{One-way}}_{\mathsf{Key\text{-}Gen},\mathcal{A}}(\lambda) = 1)$. We say that $\mathsf{Key\text{-}Gen}$ is one-way if and only if $\mathrm{Adv}^{\text{One-way}}_{\mathsf{Key\text{-}Gen},\mathcal{A}}$ is negligible for every probabilistic polynomial time adversary $\mathcal{A}$.

From now on, when speaking about key generation algorithms, we always consider $\mathsf{Key\text{-}Gen}$ algorithms for which the one-way property holds. Moreover we omit to explicitly write the relation $\mathcal{R}$ when not necessary.

**Definition 2** (Canonical identification protocol)**.** A *canonical identification protocol* is an interactive protocol between a prover $P$ and a verifier $V$ and is defined by the tuple

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

- $\mathsf{Setup}(\lambda)$: on input a security parameter $\lambda$, it outputs public parameters $\mathsf{pp}$;
- $\mathsf{Key\text{-}Gen}(\mathsf{pp}; R)$: it is a probabilistic *key generation algorithm* that takes as input the public parameters $\mathsf{pp}$ and outputs a public key $\mathsf{pk}$ and the corresponding secret key $\mathsf{sk}$;
- $\mathsf{P}_{\mathrm{CMT}}(\mathsf{sk}; R)$: it is a probabilistic algorithm called *prover commitment* that takes as input a secret key $\mathsf{sk}$ and outputs a commitment $\mathrm{CMT} \in \mathcal{X}$;
- $\mathsf{P}_{\mathrm{RSP}}(\mathsf{sk}, \mathrm{CMT}, \mathrm{CH}; R)$: it is a probabilistic algorithm called *prover response* that takes as input a private key $\mathsf{sk}$, a commitment $\mathrm{CMT}$ and a challenge $\mathrm{CH}$ and outputs a response $\mathrm{RSP}$;
- $V(\mathsf{pk}, \mathrm{CMT}, \mathrm{CH}, \mathrm{RSP})$: it is a deterministic algorithm, called Verifier, which takes as input a public key, a commitment $\mathrm{CMT}$, a challenge $\mathrm{CH}$ and a response $\mathrm{RSP}$, and outputs `accept` or `reject`.

Public Data : the security parameter $\lambda$ and the public parameters $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(\lambda)$
Key Pair : $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Key\text{-}Gen}(\mathsf{pp}, \lambda)$.

| **PROVER** $P(\mathsf{pk}, \mathsf{sk}, \mathsf{pp})$ | | **VERIFIER** $V(\mathsf{pk}, \mathsf{pp})$ |
|---|---|---|
| $R \xleftarrow{\$} \mathtt{Coins}(\lambda)$ and $\textsc{Cmt} \leftarrow \mathsf{P}_{\textsc{Cmt}}(\mathsf{sk}; R)$ | $\xrightarrow{\textsc{Cmt}}$ | |
| | $\xleftarrow{\textsc{Ch}}$ | $\textsc{Ch} \xleftarrow{\$} \{0,1\}^{c(\lambda)}$. |
| $\textsc{Rsp} \leftarrow \mathsf{P}_{\textsc{Rsp}}(\mathsf{sk}, \textsc{Ch}, \textsc{Cmt}; R)$ | $\xrightarrow{\textsc{Rsp}}$ | |
| | | Return $V(\mathsf{pk}, \textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$. |

**Fig. 1.** Canonical identification scheme

A schematic interaction between the prover and the verifier is shown inFigure 1.

It is possible to characterise the way commitments are generated in a canonical identification scheme by evaluating the *min-entropy function* of $\mathsf{P}_{\textsc{Cmt}}$, which provides an upper bound to the probability that $\mathsf{P}_{\textsc{Cmt}}(\mathsf{sk}, R)$ generates a specific commitment in the space $\mathcal{X}$.

**Definition 3** (Min-entropy of $\mathsf{P}_{\textsc{Cmt}}$)**.** Being

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

the probability that $\mathsf{P}_{\textsc{Cmt}}$, executed by a user controlling the secret key $\mathsf{sk}$, 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

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

Note that, if the algorithm $\mathsf{P}_{\textsc{Cmt}}$ instructs the prover to select uniformly at random the commitment $\textsc{Cmt}$ in the set $\mathcal{X}$, as it happens for most canonical identification schemes[30,8,16], then $\alpha(\mathsf{sk}) = \frac{1}{|\mathcal{X}|}$ and the min-entropy is $\beta(\lambda) = \log_2 |\mathcal{X}|$.

An identification schemes which have the $\mathsf{P}_{\textsc{Cmt}}$ with high min-entropy is called *non-trivial canonical identification schemes.*

**Definition 4** (Non-triviality)**.** A canonical identification scheme is called *non-trivial* if the min-entropy of the commitments is super-logarithmic in the security parameter $\lambda$ [1].

To an identification scheme $\mathcal{ID}$ and a pair $(\mathsf{pk}, \mathsf{sk})$ it is associated a randomized transcript generation oracle $\mathtt{Tr}^{\mathcal{ID}}_{\mathsf{pk},\mathsf{sk},\lambda}$ which takes no inputs and returns a random transcript $(\textsc{Cmt}, \textsc{Ch}, \textsc{Rsp}) \xleftarrow{\$} \mathtt{Tr}^{\mathcal{ID}}_{\mathsf{pk},\mathsf{sk},\lambda}$ of an honest execution such that $V(\mathsf{pk}, \textsc{Cmt}, \textsc{Ch}, \textsc{Rsp}) = 1$.

An important notion of security for canonical identification schemes is the *security against impersonation under passive attacks or (eavesdropping attacks).*

In this notion we assume that the impersonator can see a polynomial number of transcripts of the real prover interacting with an honest verifier (it receives the transcripts from $\mathtt{Tr}^{\mathcal{ID}}_{\mathsf{pk},\mathsf{sk},\lambda}$), then it must produce its impersonation attempt.

**Definition 5** (Security against impersonation under passive attack)**.** Let $\mathcal{ID}$ be a canonical identification scheme and let $\mathcal{I}$ be an impersonator, $\mathsf{st}$ be its state and $\lambda$ be the security parameter.

Define the following experiment

```
$\mathrm{Exp}^{\text{imp-pa}}_{\mathcal{ID},\mathcal{I}}(\lambda)$ :
1 :  $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Key\text{-}Gen}(\lambda)$
2 :  $\mathsf{st}||\textsc{Cmt} \xleftarrow{\$} \mathcal{I}^{\mathtt{Tr}^{\mathcal{ID}}_{\mathsf{pk},\mathsf{sk},\lambda}}(\mathsf{pk})$
3 :  $\textsc{Ch} \xleftarrow{\$} \{0,1\}^{c(\lambda)}$
4 :  $\textsc{Rsp} \xleftarrow{\$} \mathcal{I}(\mathsf{st}, \textsc{Ch})$
5 :  return $V(\mathsf{pk}, \textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$
```

We define the advantage of $\mathcal{I}$ in winning $\mathrm{Exp}^{\text{imp-pa}}_{\mathcal{ID},\mathcal{I}}(\lambda)$ as $\mathrm{Adv}^{\text{imp-pa}}_{\mathcal{ID},\mathcal{I}}(\lambda) = \mathbb{P}(\mathrm{Exp}^{\text{imp-pa}}_{\mathcal{ID},\mathcal{I}}(\lambda) = 1)$. $\mathcal{ID}$ is secure against impersonations under passive attacks if $\mathrm{Adv}^{\text{imp-pa}}_{\mathcal{ID},\mathcal{I}}(\lambda)$ is negligible for every probabilistic polynomial time impersonator $\mathcal{I}$.

**Observation 1.** A standard way to prove a canonical identification scheme secure against impersonation under passive attacks consists into proving that:

1. no adversary $\mathcal{A}$ can win the experiment without interacting with the transcript oracle $\mathtt{Tr}^{\mathcal{ID}}_{\mathsf{pk},\mathsf{sk},\lambda}$ (see *direct attacks* in Definition 18.2 of [9]);
2. the transcript oracle $\mathtt{Tr}^{\mathcal{ID}}_{\mathsf{pk},\mathsf{sk},\lambda}$ can be simulated, i.e. the sigma protocol underlying the identification scheme is *honest-verifier zero-knowledge (HVZK).*

## 2.3 Fiat-Shamir Transform

Firstly introduced in [20], the Fiat-Shamir Transform is a widespread heuristic used to design digital signature schemes starting from canonical identification schemes, replacing the challenge step with a cryptographic hash function. This technique 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.

Now we can formally define the Fiat-Shamir Transform.

**Definition 6** (Fiat-Shamir Transform)**.** Let $\mathcal{ID}$ be a canonical identification scheme, let $c$ be the challenge length's function and let $H : \{0,1\}^* \to \{0,1\}^{c(k)}$ be a public hash function. The signature scheme $\mathcal{DS}$ uses the same setup and key generation algorithm as the identification scheme, while the signing and verification algorithms are the following:

```
Sign(sk, m)
1 : $R \xleftarrow{\$} \mathsf{Coins}(\lambda)$
2 : $\textsc{Cmt} \leftarrow \mathsf{P}_{\textsc{Cmt}}(\mathsf{sk}; R)$
3 : $\textsc{Ch} \leftarrow H(\textsc{Cmt}||\mathtt{m})$
4 : $\textsc{Rsp} \leftarrow \mathsf{P}_{\textsc{Rsp}}(\mathsf{sk}, \textsc{Cmt}||\textsc{Ch}; R)$
5 : return $\textsc{Cmt}||\textsc{Rsp}$
```

```
Ver(pk, m, σ)
1 : Parse $\sigma$ as $\textsc{Cmt}|\textsc{Rsp}$
2 : $\textsc{Ch} \leftarrow H(\textsc{Cmt}||\mathtt{m})$
3 : return $V(\mathsf{pk}, \textsc{Cmt}||\textsc{Ch}||\textsc{Rsp})$
```

A relevant notion of security for digital signature schemes is the notion of *unforgeability under chosen message attacks*. Informally speaking, a digital signature is unforgeable under chosen message attacks if it is impossible for an adversary to produce a forgery even after seeing the signature of a polynomial number of messages of its choice.

**Definition 7** (Unforgeability under chosen message attack)**.** Let $\mathcal{DS}$ be a digital signature scheme defined by the triple $\mathcal{DS} = (\mathsf{Key\text{-}Gen}, \mathsf{Sign}, V)$. Let $\mathcal{F}$ be a forger having access to a signing oracle $\mathcal{O}^H_{\mathcal{DS}}(\cdot)$ and to the random oracle $\mathcal{O}_H(\cdot)$[1]. Define the following experiment where $Q$ represents the set of messages queried by $\mathcal{F}$ to $\mathcal{O}^H_{\mathcal{DS}}(\cdot)$.

```
$\mathrm{Exp}^{\text{uf-cma}}_{\mathcal{DS},\mathcal{F}}(\lambda)$ :
1 : $H \xleftarrow{\$} [\{0,1\}^* \rightarrow \{0,1\}^c]$
2 : $(\mathsf{pk}, \mathsf{sk}) \xleftarrow{\$} \mathsf{Key\text{-}Gen}(\lambda)$
3 : $\mathtt{m}||\sigma \xleftarrow{\$} \mathcal{F}^{\mathcal{O}^H_{\mathcal{DS}}(\cdot), \mathcal{O}_H(\cdot)}(\mathsf{pk})$
4 : If $\mathtt{m} \in Q$ return 0
5 : Else return $V(\mathsf{pk}, \textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$
```

Define the advantage of $\mathcal{F}$ as $\mathrm{Adv}^{\text{uf-cma}}_{\mathcal{DS},\mathcal{F}}(\lambda) = \mathbb{P}(\mathrm{Exp}^{\text{uf-cma}}_{\mathcal{DS},\mathcal{F}}(\lambda) = 1)$. $\mathcal{DS}$ is unforgeable against chosen message attacks if $\mathrm{Adv}^{\text{uf-cma}}_{\mathcal{DS},\mathcal{F}}(\lambda)(\cdot)$ is negligible for every probabilistic polynomial time forger $\mathcal{F}$.

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].

[1] The access to the random oracle $\mathcal{O}_H(\cdot)$ is guaranteed only if it is required to create a signature according to $\mathcal{DS}$.

[2] In case the min-entropy of the commitments is less than super-logarithmic, it is always possible to consider a modified version of the protocol where $\textsc{Ch} = H(\textsc{Cmt}||R_{\textsc{Cmt}}||\mathtt{m})$, where $R_{\textsc{Cmt}}$ is a random string of appropriate length, such that $\textsc{Cmt}||R_{\textsc{Cmt}}$ have the desired min-entropy. For more details about this, see [1].

### 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 8** (Security of secret sharing)**.** A $(t, n)$-secret sharing scheme $\mathtt{SS}$ between a dealer $D$, holding a secret $s$, and parties $P_1, ..., P_n$, each of them holding a share $s_i$ of $s$, is (perfectly) secure if and only if $\mathbb{P}[\mathsf{secret} = s | \{s_i\}_J] = \mathbb{P}[\mathsf{secret} = s' | \{s_i\}_J]$ for all $J \subseteq \{1, ..., n\}$ such that $|J| < t$.

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 that any secret sharing scheme 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 $\mathsf{Setup}$ and $\mathsf{Key\text{-}Gen}$ are not considered in our discussion.

- $\mathsf{Setup}(\lambda)$, on input a security parameter $\lambda$, it outputs public parameters $\mathsf{pp}$.
- $\mathsf{Key\text{-}Gen}(\mathsf{pp}, n, t, \lambda)$, on input the number of participants $n$, the threshold $t$ and the security parameter $\lambda$, it outputs a public key $\mathsf{pk}$ and a secret sharing $\mathsf{sk}_i$ of the corresponding secret key $\mathsf{sk}$, having each participant $P_i$ holding $\mathsf{sk}_i$.
- $\mathsf{TSign}(\mathtt{m}, \{\mathsf{sk}_i\}_J)$ is a multi party protocol run by parties in $J$. On input an agreed upon message $\mathtt{m}$ and shards $\mathsf{sk}_i$ from various players, it outputs a valid signature $\sigma$ if $|J| \geq t$,
- $\mathsf{Ver}(\mathsf{pk}, \mathtt{m}, \sigma)$, on input a public key $\mathsf{pk}$, a message $\mathtt{m}$ and a signature $\sigma$, 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 $\mathsf{TSign}$ to sign it. The resulting signature is verifiable against the public key $\mathsf{pk}$ via the verification algorithm $\mathsf{Ver}$.

**Security notions for threshold signature schemes.** For the security of threshold signatures we need to distinguish two security notions: unforgeability against *passive chosen message attacks* and *active chosen message attacks*. The first deals with adversaries who corrupt some parties and gain read access to their state and the messages they exchange with the network. The second deals with adversaries that gain full control over the corrupted parties.

These two scenarios must be captured by two distinct security definitions, like Definition 7, where the difference is in the way the sign queries are carried out by $\mathcal{F}$ interacting with the sign oracle. In the passive case, when $\mathcal{F}$ sends a query to the sign oracle, it receives the view of each corrupted party, given by their state and every public message, but it is not allowed to control them. In the active case, instead, $\mathcal{F}$ has full control over the corrupted parties and thus can deviate from the protocol freely. In particular $\mathcal{F}$ is allowed to interact with the sign oracle $\mathcal{O}^{H}_{\mathcal{TDS}}(J, J_h, \cdot)$ and participate in the signature computation: at every step prescribed by the signing algorithm $\mathcal{F}$ sends messages of its choice on behalf of its corrupted parties (in $J$) to the oracle, that acts on behalf the honest parties (in $J_h$).

Formally we have the following definitions:

**Definition 9** (Unforgeability under passive chosen message attacks)**.** Let $\mathcal{TDS} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TSign}, \mathsf{Ver})$ be a $(t, n)$-threshold digital signature scheme with challenge length $c$ and security parameter $\lambda$. Let $\mathcal{F}$ be a forger having access to a signing oracle $\mathcal{O}^{H}_{\mathsf{View}-\mathcal{TDS}}(\cdot)$ and to the random oracle $\mathcal{O}_H(\cdot)$. Define the advantage of $\mathcal{F}$ in winning the experiment $\mathrm{Exp}^{\text{p-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda)$ in Figure 2 as:

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

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

**Definition 10** (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 with challenge length $c$ and security parameter $\lambda$. Let $\mathcal{F}$ be a forger having access to a signing oracle $\mathcal{O}^{H}_{\mathcal{TDS}}(\cdot)$ and to the random oracle $\mathcal{O}_H(\cdot)$. Define the experiment:

Define the advantage of $\mathcal{F}$ in winning $\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},F}(\lambda)$ described in Figure 2 as

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

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 $\mathcal{F}$.

## 3 Distributed Identification Schemes and Fiat-Shamir Transform

In Section 3.1 we aim to generalise the definition of identification scheme to *threshold identification scheme.* Then, in Section 3.2 we propose a generalisation of the Fiat-Shamir Transform to the distributed case and we show how it can be used to derive threshold digital signature schemes. Finally in Section 3.3 we define 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 into threshold signature schemes by applying the distributed Fiat-Shamir Transform.

```
$\mathrm{Exp}^{\text{p-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda)$ :
1 : $(\mathsf{pp}) \xleftarrow{\$} \mathsf{Setup}(\lambda)$
2 : $(\{sk_i\}, \mathsf{pk}) \xleftarrow{\$} \mathsf{Key\text{-}Gen}(\mathsf{pp}, n, t)$
3 : $(J, \{\mathsf{sk}_i\}_{i \in J}) \xleftarrow{\$} \mathcal{F}(\mathsf{pp}, \mathsf{pk}, n, t)$
4 : $//\ |J| < t$
5 : $(\mathtt{m}, \sigma) \leftarrow \mathcal{F}^{\mathcal{O}_{\mathsf{View}-\mathcal{TDS}}(\mathsf{pk}, J, J_h)^{\mathcal{O}_H(\cdot)}, \mathcal{O}_H(\cdot)}$
6 : If $\mathtt{m} \in Q$ return $0$
7 : Else return $\mathsf{Ver}(\mathsf{pk}, \mathtt{m}, \sigma)$
```

```
$\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},F}(\lambda)$ :
1 : $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(\lambda)$
2 : $(\{sk_i\}, \mathsf{pk}) \xleftarrow{\$} \mathsf{Key\text{-}Gen}(\mathsf{pp}, n, t)$
3 : $(J, \{\mathsf{sk}_i\}_{i \in J}) \xleftarrow{\$} \mathcal{F}(\mathsf{pp}, \mathsf{pk}, n, t)$
4 : $//\ |J| < t$
5 : $(\mathtt{m}, \sigma) \leftarrow \mathcal{F}^{\mathcal{O}_{\mathcal{TDS}}(\mathsf{pk}, J, J_h)^{\mathcal{O}_H(\cdot)}, \mathcal{O}_H(\cdot)}$
6 : If $\mathtt{m} \in Q$ return $0$
7 : Else return $\mathsf{Ver}(\mathsf{pk}, \mathtt{m}, \sigma)$
```

$\mathcal{O}_{\mathsf{View}-\mathcal{TDS}}(\mathsf{pk}, J, J_h)$

Provides $\mathcal{F}$ with the view of parties in $J$ who interact with the parties in $J_h$ in a honest execution.

$\mathcal{O}_{\mathcal{TDS}}(\mathsf{pk}, J, J_h)$

Controls the parties in $J_h$, and interacts with $\mathcal{F}$ controlling the parties in $J$.

**Fig. 2.** Experiments for the unforgeability of a threshold digital signature against active and passive attacks. $J_h \subset \{1, ..., n\} \setminus J$ denotes the set of honest parties that the oracle controls and that the adversary can choose adaptively before each query. $Q$ is the set of messages queried by $\mathcal{F}$ to the sign oracle.

## 3.1 Threshold identification schemes

We generalize Definition 2 and define protocols that allow multiple provers $P_1, ..., P_n$, holding a secret sharing of a secret $\mathsf{sk}$, to prove their joint knowledge of $\mathsf{sk}$. 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 commitment CMT, then, after receiving a challenge CH, the parties in $J$ jointly run $\mathsf{TP}_{\mathrm{RSP}}$ to produce a response RSP.

**Definition 11** (Canonical $(t, n)-$ identification protocol)**.** Let $P_1, \ldots, P_n$ be a set of players. 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 $\lambda$, it outputs public parameters $\mathsf{pp}$.
- $\mathsf{Key\text{-}Gen}(n, t, \mathsf{pp}; R)$: it is a probabilistic *key generation algorithm* that takes as input the public parameters $\mathsf{pp}$, the number of participants $n$ and the threshold $t$, and outputs a public key $\mathsf{pk}$ and a secret sharing $\mathtt{SS}(\mathsf{sk}, t, n; R) = \{\mathsf{sk}_i\}_{[n]}$ of the secret key $\mathsf{sk}$, with each participant $P_i$ holding $\mathsf{sk}_i$;
- $\mathsf{TP}_{\mathrm{CMT}}(\{\mathsf{sk}_i\}_J; \mathbf{R})$: it is a probabilistic multi-party protocol run by parties in $J$ called *threshold prover commitment.* On input the private keys $\mathsf{sk}_i$ it outputs a common commitment CMT;

– $\mathsf{TP}_{\mathrm{Rsp}}(\{\mathsf{sk}_i\}_J, \mathrm{Cmt}, \mathrm{Ch}; \mathbf{R})$: it is a probabilistic multi party protocol run by parties in $J$ called *threshold prover response*. It takes as input shards $\mathsf{sk}_i$ from the various player, a commitment $\mathrm{Cmt}$ and a challenge $\mathrm{Ch}$, and outputs a valid response $\mathrm{Rsp}$ if $|J| \geq t$;
– $V(\mathsf{pk}, \mathrm{Cmt}, \mathrm{Ch}, \mathrm{Rsp})$: it is a centralized protocol called Verifier which takes in input a public key, a commitment $\mathrm{Cmt}$, a challenge $\mathrm{Ch}$ and a response $\mathrm{Rsp}$, and outputs `accept` or `reject`.

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

In Figure 3 we represent the execution of a threshold identification scheme.

Public Data : public parameter $\mathsf{pp}$ and the security parameter $\lambda$

Private Key : Each player $i \in J$ holds $\mathsf{sk}_i$, such that $(\mathsf{pk}, \{\mathsf{sk}_i\}) \overset{\$}{\leftarrow} \mathsf{Key\text{-}Gen}(\mathsf{pp}, \lambda)$.

Public Key : $\mathsf{pk}$

| **PROVERS** | | **VERIFIER** |
|---|---|---|
| $\mathbf{R} = [R_i]_{i \in J} \leftarrow \mathtt{Coins}(\lambda)^t$ and | | |
| $\mathrm{Cmt} \leftarrow \mathsf{TP}_{\mathrm{Cmt}}(\{\mathsf{sk}_i\}_J; \mathbf{R})$ | $\xrightarrow{\mathrm{Cmt}}$ | |
| | $\xleftarrow{\mathrm{Ch}}$ | $\mathrm{Ch} \overset{\$}{\leftarrow} \{0,1\}^{c(\lambda)}$. |
| $\mathrm{Rsp} \leftarrow \mathsf{TP}_{\mathrm{Rsp}}(\{\mathsf{sk}_i\}_J, \mathrm{Ch}, \mathrm{Cmt}; \mathbf{R})$ | $\xrightarrow{\mathrm{Rsp}}$ | |
| | | Return $V(\mathsf{pk}, \mathrm{Cmt}, \mathrm{Ch}, \mathrm{Rsp})$. |

**Fig. 3.** Threshold identification scheme.

As before, since we are supposing the presence of a trusted dealer, both the $\mathsf{Setup}$ and $\mathsf{Key\text{-}Gen}$ 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 try to extend to the distributed case the concept of security against impersonation under passive attacks which applies to identification schemes (Definition 5). We extend it to the distributed case following the lead of the definition of unforgeability for threshold digital signatures. We make a distinction between passive and active adversaries, namely adversaries that during the training phase can only see honest transcripts of the identification process (passive adversary), and adversaries who can take part to the identification process (active adversary).

Let $\mathcal{TID}$ be a threshold identification protocol, with public key $\mathsf{pk}$ and secret keys $\{\mathsf{sk}_i\}_{i\in[n]}$, we associate to $\mathcal{TID}$ two oracles that will be used in the experiments that define the security notions below:

- a transcript generation oracle $\mathcal{O}_{\mathsf{View}-\mathcal{TID}}(\mathsf{pk}, J, J_h)$ that takes as input the public key, two non-empty sets of parties $J$ and $J_h$ such that $|J \cup J_h| = t$ and returns random transcript conversation of an honest execution of $\mathcal{TID}$, including all the public messages and the internal state of parties in $J$.
- a threshold identification oracle $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, J, J_h)$ that, on input a pubic key $\mathsf{pk}$ and two non-empty sets of parties $J, J_h$ such that $|J \cup J_h| = t$, interacts with $\mathcal{A}$ in an execution of $\mathcal{TID}$. In particular, $\mathcal{A}$ controls the parties in $J$, and $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, J, J_h)$ controls both the parties in $J_h$ and the verifier who randomly picks a challenge once the commitment has been created.

**Definition 12** (Security against impersonation under passive attacks)**.** Let $\mathcal{TID} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TP}_{\mathrm{CMT}}, \mathsf{TP}_{\mathrm{RSP}}, V)$ be a $(t, n)$-threshold identification scheme with challenge length $c$ and security parameter $\lambda$. Let $\mathcal{A}$ be an impersonator having access to a threshold transcript generation oracle $\mathcal{O}_{\mathsf{View}-\mathcal{TID}}(\cdot)$.

We define the advantage of $\mathcal{A}$ in winning the experiment $\mathrm{Exp}^{\text{p-imp}}_{\mathcal{TID},\mathcal{A}}(\lambda)$ described in Figure 4 as

$$\mathrm{Adv}^{\text{p-imp}}_{\mathcal{TID},\mathcal{A}}(\lambda) = \mathbb{P}(\mathrm{Exp}^{\text{p-imp}}_{\mathcal{TID},\mathcal{A}}(\lambda) = 1)$$

We say that $\mathcal{TID}$ is *secure against impersonation under passive attack* if $\mathrm{Adv}^{\text{p-imp}}_{\mathcal{TID},\mathcal{A}}(\lambda)(\cdot)$ is negligible for every probabilistic polynomial time impersonator $\mathcal{A}$.

**Definition 13** (Security against impersonation under active attacks)**.** Let $\mathcal{TID} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TP}_{\mathrm{CMT}}, \mathsf{TP}_{\mathrm{RSP}}, V)$ be a $(t, n)$-threshold identification scheme with challenge length $c$ and security parameter $\lambda$. Let $\mathcal{A}$ be an impersonator having access to a threshold identification oracle $\mathcal{O}_{\mathcal{TID}}(\cdot)$.

Define the advantage of $\mathcal{A}$ in winning the experiment $\mathrm{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{A}}(\lambda)$ described in Figure 4 as

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

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

In both cases $\mathcal{A}$ can corrupt at most $t - 1$ parties and obtain their private keys. The first difference is that in the active case, $\mathcal{A}$ interacts with the identification oracle $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, J, J_h)$ that plays the role of the honest parties (that the adversary can adaptively choose), while in the passive case it can only query the transcript generation oracle $\mathcal{O}_{\mathsf{View}-\mathcal{TID}}(\mathsf{pk}, J, J_h)$, that provides $\mathcal{A}$ with the transcript of honest execution of the identification protocol. In particular, those transcripts comprise all the internal states of the parties in $J$ and all the public messages.

```
$\mathrm{Exp}^{\text{p-imp}}_{\mathcal{TID},\mathcal{A}}(\lambda)$ :
1 : $(\mathsf{pp}) \xleftarrow{\$} \mathsf{Setup}(\lambda)$
2 : $(\{\mathsf{sk}_i\}, \mathsf{pk}) \xleftarrow{\$} \mathsf{Key\text{-}Gen}(\mathsf{pp}, n, t)$
3 : $(J, \{\mathsf{sk}_i\}_{i \in J}) \xleftarrow{\$} \mathcal{A}(\mathsf{pp}, \mathsf{pk}, n, t)$
4 : // $|\mathtt{J}| \leq \mathtt{t} - 1$
5 : $st || \textsc{Cmt} \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathsf{View}-\mathcal{TID}}(\mathsf{pk}, J, J_h)}$
6 : $\textsc{Ch} \xleftarrow{\$} \{0,1\}^{c(\lambda)}$
7 : $\textsc{Rsp} \xleftarrow{\$} \mathcal{A}(st, \textsc{Ch})$
8 : return $V(\mathsf{pk}, \textsc{Cmt}||\textsc{Ch}||\textsc{Rsp})$

$\mathcal{O}_{\mathsf{View}-\mathcal{TID}}(\mathsf{pk}, J, J_h)$
Provides $\mathcal{I}$ with the view of parties in
$J$ who interact with the parties in $J_h$.
```

```
$\mathrm{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{A}}(\lambda)$ :
1 : $\mathsf{pp} \xleftarrow{\$} \mathsf{Setup}(\lambda)$
2 : $(\{\mathsf{sk}_i\}, \mathsf{pk}) \xleftarrow{\$} \mathsf{Key\text{-}Gen}(\mathsf{pp}, n, t)$
3 : $(J, \{\mathsf{sk}_i\}_{i \in J}) \xleftarrow{\$} \mathcal{A}(\mathsf{pp}, \mathsf{pk}, n, t)$
4 : //$|\mathtt{J}| \leq \mathtt{t} - 1$
5 : $st || \textsc{Cmt} \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, J, J_h)}$
6 : $\textsc{Ch} \xleftarrow{\$} \{0,1\}^{c(\lambda)}$
7 : $st' \xleftarrow{\$} \mathcal{A}^{\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, J, J_h)}$
8 : $\textsc{Rsp} \xleftarrow{\$} \mathcal{A}(st', \textsc{Ch})$
9 : return $V(\mathsf{pk}, \textsc{Cmt}||\textsc{Ch}||\textsc{Rsp})$

$\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, J, J_h)$
Controls the parties in $J_h$, and
interacts with $\mathcal{I}$ controlling $J$.
```

**Fig. 4.** Experiments of active and passive impersonation attacks. $J_h \subset \{1, ..., n\} \setminus J$ denotes the set of honest parties that the oracle controls and that the adversary can choose adaptively before each query.

The second difference, is that while in the passive case $\mathcal{A}$ can be assumed to receive all the transcripts from the $\mathcal{O}_{\mathsf{View}-\mathcal{TID}}(\cdot)$ before it creates the commitment $\textsc{Cmt}$ of the impersonation attempt, in the active case $\mathcal{A}$ is allowed to interact with the identification oracle $\mathcal{O}_{\mathcal{TID}}(\cdot)$ also after it has sent the commitment of the impersonation attempt and has received the challenge $\textsc{Ch}$ from the verifier (Figure 4, $\mathrm{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{A}}(\lambda)$, line 7). This choice aims to expand as much as possible the capabilities of an attacker running the experiment.

### 3.2 Distributed Fiat-Shamir Transform

We adapt the definition of *Fiat-Shamir transform* presented in [1] to the distributed case:

**Definition 14** (Distributed Fiat-Shamir transform)**.** Let $\mathcal{TID}$ be a canonical threshold identification scheme with $\mathcal{TID} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TP}_{\textsc{Cmt}}, \mathsf{TP}_{\textsc{Rsp}}, V)$.

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

The signature has the same $\mathsf{Setup}$ 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 : $\textsc{Cmt} \leftarrow \mathsf{TP}_{\textsc{Cmt}}(\{sk_i\}_{i \in J}; \mathbf{R})$
3 : $\textsc{Ch} \leftarrow H(\textsc{Cmt}||\mathtt{m})$
4 : $\textsc{Rsp} \leftarrow \mathsf{TP}_{\textsc{Rsp}}(\{sk_i\}_{i \in J}, \textsc{Cmt}, \textsc{Ch}; \mathbf{R})$
5 : return $\textsc{Cmt}||\textsc{Rsp}$

$\mathsf{Ver}(\mathsf{pk}, \mathtt{m}, \sigma)$:
1 : Parse $\sigma$ as $\textsc{Cmt}|\textsc{Rsp}$
2 : $\textsc{Ch} \leftarrow H(\textsc{Cmt}||\mathtt{m})$
3 : return $V(\mathsf{pk}, \textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$
```

We now define two properties that can be satisfied by the protocol used as $\mathsf{TP}_{\textsc{Cmt}}$ in threshold identification schemes. In particular, these properties are used as additional assumptions that threshold identification schemes must satisfy in order to obtain threshold digital signature schemes unforgeable under passive and active chosen message attacks.

## 3.3 Requirements on $\mathsf{TP}_{\textsc{Cmt}}$

We want to characterise a class of algorithms $\mathsf{TP}_{\textsc{Cmt}}$ 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. We refer to this class of $\mathsf{TP}_{\textsc{Cmt}}$ as *unpredictable*.

**Definition 15** (Unpredictable $\mathsf{TP}_{\textsc{Cmt}}$)**.** Let $\mathcal{TID}$ be a $(t, n)$-threshold identification scheme with $\mathcal{TID} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TP}_{\textsc{Cmt}}, \mathsf{TP}_{\textsc{Rsp}}, V)$. We say that $\mathsf{TP}_{\textsc{Cmt}}$ is *unpredictable*, if and only if the output $\textsc{Cmt}$ has super-logarithmic min-entropy when at least one party is honest.

**Observation 2.** Note that the definition of threshold identification scheme with unpredictable $\mathsf{TP}_{\textsc{Cmt}}$ is the analogue of the definition of non-trivial identification scheme in the distributed case.

One way to design a $\mathsf{TP}_{\textsc{Cmt}}$ algorithm such that the output distribution can not be controlled by a subset of the parties is by requiring each party $P_i$ to produce and simultaneously share a partial commitment $\textsc{Cmt}_i$ starting from which the final commitment $\textsc{Cmt}$ can be deterministically computed. In multi party protocols simultaneity can be achieved by performing a two-steps protocol in which each party first creates a cryptographic commitment[3] to $\textsc{Cmt}_i$, and then, only after each party has published its cryptographic commitment, everyone

[3] From now on we must deal with the ambiguity of the term "commitment" which might refer both to the first message exchanged in a canonical identification scheme, and to the output of the commit algorithm in a commitment scheme. From the context it will be clear which commitment we are referring to and we will use for the latter the term "cryptographic commitment" or "commitment scheme".

opens it. At this point, the parties in $J$ aggregate all the $\textsc{Cmt}_i$ with an agreed upon function such that if at least one party $P_j$ in the group is honest and picked $\textsc{Cmt}_j$ uniformly at random in $\mathcal{X}$, then also $\textsc{Cmt}$ have uniform distribution in $\mathcal{X}$.

Formally we have the following definition:

**Definition 16** (Commit-Release $\mathsf{TP}_{\textsc{Cmt}}$)**.** Let $\mathcal{TID}$ be a threshold identification scheme with $\mathcal{TID} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TP}_{\textsc{Cmt}}, \mathsf{TP}_{\textsc{Rsp}}, V)$. We say that $\mathsf{TP}_{\textsc{Cmt}}$ *is commit-release* if and only if the $\mathsf{TP}_{\textsc{Cmt}}$ protocol is unpredictable and has the following structure:

- $\mathsf{TP}^{\mathrm{Com}}_{\textsc{Cmt}}(\mathsf{sk}_i; R)$: a non interactive protocol run locally by each party that outputs a one-way cryptographic commitment (Definition 17) $\mathsf{Commit}(\mathtt{ssid}||\textsc{Cmt}_i)$ where $\textsc{Cmt}_i$ is picked uniformly at random in $\mathcal{X}$, and $\mathtt{ssid}$ is a session identifier shared among all the parties involved in the execution.
- $\mathsf{TP}^{\mathrm{Rel}}_{\textsc{Cmt}}(\{\mathsf{Commit}(\mathtt{ssid}||\textsc{Cmt}_i)\}_{i\in[t]})$: an interactive deterministic protocol run by all the parties involved in the threshold identification execution that release $\textsc{Cmt}_i$ opening $\mathsf{Commit}(\mathtt{ssid}||\textsc{Cmt}_i)$, and output a common $\textsc{Cmt}$ obtained deterministically by combining the partial commitments $\textsc{Cmt}_i$.

Note that it is required for the commitment scheme to be binding, so that once created the cryptographic commitment to $\textsc{Cmt}_i$, $P_i$ can only reveal its partial commitment, but it is not required for the commitment to satisfy the hiding property [22]. In particular it is enough that the commitment scheme satisfies a weaker privacy property, namely the one-way property, that means that it is impossible to retrieve the committed data having access only to its cryptographic commitment. More formally we have the following definition:

**Definition 17** (One-way commitment scheme)**.** Let $(\mathsf{PGen}(\cdot), \mathsf{Commit}(\cdot), \mathsf{Open}(\cdot))$ be a commitment scheme where:

- $\mathsf{PGen}(1^\lambda)$ takes as input a security parameter $\lambda$ and returns public parameters $\mathsf{pp}$;
- $\mathsf{Commit}(\mathsf{pp}, x)$ takes as input the public parameters $\mathsf{pp}$, a message $x$ in $\mathcal{X}$ and returns the commitment $c$ and the *opening material* $r$;
- $\mathsf{Open}(\mathsf{pp}, x, c, r)$ takes as input the public parameters $\mathsf{pp}$, the message $x$, the commitment $c$ and the opening material $r$, and returns $\mathtt{accept}$ if $c$ is a commitment of $x$ or $\mathtt{reject}$ otherwise.

We say that $(\mathsf{PGen}, \mathsf{Commit}, \mathsf{Open})$ is a *one-way commitment scheme* if it satisfies the binding property [22] and the one-way property, which means that no adversary can win the following game with non-negligible advantage in the security parameter $\lambda$.

```
$\mathrm{Exp}^{\text{One-way}}_{\mathrm{Com},\mathcal{A}}(\lambda)$ :
1 : $\mathsf{pp} \xleftarrow{\$} \mathsf{PGen}(\lambda)$
2 : $x \xleftarrow{\$} X$
3 : $(c, r) \xleftarrow{\$} \mathsf{Commit}(\mathsf{pp}, x)$
4 : $x' \xleftarrow{\$} \mathcal{A}(\mathsf{pp}, c)$
5 : $\textbf{return } x' = x$
```

The advantage of an adversary $\mathcal{A}$ playing the game above is defined as $\mathrm{Adv}^{\text{One-way}}_{\mathrm{Com},\mathcal{A}}(\lambda) = \mathbb{P}[\mathrm{Exp}^{\text{One-way}}_{\mathrm{Com},\mathcal{A}}(\lambda) = 1]$.

**Observation 3.** A commitment scheme which satisfies the hiding property [22] also satisfies the one-way property as we prove in Appendix A

## 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. In Section 4.1 we analyse the security against active adversaries, then, in Section 4.2 we consider passive adversaries, which has a very similar proof, provided in Appendix C.

### 4.1 Active security

The main difference between the multi-party setting and the centralized one from [1] is that we also need to deal with active adversaries. Indeed, in the centralized we do not need to consider adversaries capable of influencing the messages distribution during the sign queries (indeed, the adversary only receives the signatures of the queried messages), while in the distributed case we also need to consider this eventuality.

**Theorem 1** (Active security)**.** *Let $\mathcal{TID} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TP}_{\mathrm{CMT}}, \mathsf{TP}_{\mathrm{RSP}}, V)$ 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 14. Then, assuming the ROM, the following implications hold:*

1. *($\mathcal{TID} \implies \mathcal{TDS}$): if $\mathsf{TP}_{\mathrm{CMT}}$ satisfies the commit-release property as per Definition 16 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.*

We prove separately the two implications. Notice that, as stated in Section 1.1, we provide a game based proof, thus security of parallel composition is not "automatically" granted. Instead, to obtain a signature secure also for parallel composition we should require the same property on the starting identification scheme.

**Lemma 1** **($\mathcal{TID} \implies \mathcal{TDS}$).** *Under the assumptions of Theorem 1,if $\mathsf{TP}_{\textsc{Cmt}}$ satisfies the commit-release property and $\mathcal{TID}$ is secure against impersonation under active attacks, then $\mathcal{TDS}$ is unforgeable against active chosen-message attacks.*

*Proof.* Let $\mathcal{F}$ be a forger that wins the $\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},F}(\lambda)$ 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.

Now we show how to define the impersonator $\mathcal{I}$ starting from $\mathcal{F}$. Firstly, we describe the high level picture, with the relation between the parties involved in the reduction, then we describe how $\mathcal{I}$ uses the adversary $\mathcal{F}$ by simulating the challenger of the experiment $\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda)$ both in the initialization and in the training phase. Then, we show that the simulation is successful with overwhelming probability, and we show how $\mathcal{I}$ can exploit $\mathcal{F}$'s forgery to perform its impersonation attempt. Finally, we show that if $\mathcal{F}$ has non-negligible advantage in winning $\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda)$, then also $\mathcal{I}$ has non-negligible advantage in winning $\mathrm{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}(\lambda)$ which leads to a contradiction because $\mathcal{TID}$ is secure against impersonation under active attacks. Therefore, such $\mathcal{F}$ do not exist and $\mathcal{TDS}$ is secure.

*High level picture.* The impersonator $\mathcal{I}$ interacts with the challenger $\mathcal{C}_{\mathcal{TID}}$ of experiment $\mathrm{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{A}}(\lambda)$ and has access to the transcript oracle $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, J, J_h)$ that can query up to $q_s(\lambda)$ times, where $J$ and $J_h$ are chosen by $\mathcal{I}$. In order to exploit the advantage of $\mathcal{F}$, and use it as a subroutine, $\mathcal{I}$ will simulate the challenger $\mathcal{C}_{\mathcal{TDS}}$ of the experiment $\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},F}(\lambda)$ executed by $\mathcal{F}$ and the sign and random oracles $\mathcal{O}^H_{\mathcal{TDS}}(\{sk_i\}_{i\in J})$ 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 $\lambda$, being $\mathcal{F}$ a polynomial-time adversary.

In Appendix B, Figure 7, we provide a graphical representation of the reduction we describe below.

*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 $\mathsf{pp}$ of the identification protocol and the public key $\mathsf{pk}$. $\mathcal{I}$ forwards this information to $\mathcal{F}$.

*Training phase.* $\mathcal{F}$ chooses the the set $J$ (with $|J| \leq t-1$) of actors it wants to control. $\mathcal{I}$ chooses the same set $J$ and sends it to $\mathcal{C}_{\mathcal{TID}}$, receiving the secret keys of the players in $J$, 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 $\mathsf{HT}$ 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 $\mathrm{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}(\lambda)$ 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 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 $J_h$, the set of the honest players who, together with the parties in $J$, participates in the computation of a signature of $\mathtt{m}$. $\mathcal{I}$ increases the signature counter $sc$ and sends to $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, J, J_h)$ a request to perform the threshold identification protocol. The impersonator $\mathcal{I}$ acts as a man in the middle between $\mathcal{F}$ and $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, J, J_h)$, and repeats the following operations for each step prescribed by the algorithm $\mathsf{TP}_{\textsc{Cmt}}$:
  1. $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, J, J_h)$ produces the messages for the participants in $J_h$, and anticipates $\mathcal{I}$ in the execution of the steps prescribed by $\mathsf{TP}^{\text{Com}}_{\textsc{Cmt}}$;
  2. $\mathcal{I}$ forwards to $\mathcal{F}$ the messages received from $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, J, J_h)$;
  3. $\mathcal{F}$ produces the messages executing $\mathsf{TP}^{\text{Com}}_{\textsc{Cmt}}$ on behalf of the corrupted participants in $J$.
  4. $\mathcal{I}$ forwards the messages received from $\mathcal{F}$ to $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, J, J_h)$.

  These steps are repeated for the protocol $\mathsf{TP}^{\text{Rel}}_{\textsc{Cmt}}$, 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 $J$ and finally by $\mathcal{O}_{\mathcal{TID}}$, once it receives the messages forwarded by $\mathcal{I}$.

  The oracle $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, J, J_h)$ 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 $\mathsf{HT}$ 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 $\mathcal{O}_{\mathcal{TID}}(\mathsf{pk}, J, J_h)$ 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 $\mathsf{HT}$ during a sign query performed by $\mathcal{F}$. The overwriting of $\mathsf{HT}$ 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. Before computing the commitment $\textsc{Cmt}$ associated to a sign query for $\mathtt{m}$, $\mathcal{F}$ has performed a hash query for $\textsc{Cmt}||\mathtt{m}$. We must consider again two possible scenarios:
   (a) After $\mathcal{I}$ has released its partial commitments $\textsc{Cmt}_i$,$i \in J_h$, opening its cryptographic commitments in the execution of $\mathsf{TP}^{Rel}_{\textsc{Cmt}}$ (therefore $\mathcal{F}$ already knows the value of $\textsc{Cmt}$). Later we discuss how to deal with this case which does not contribute in the evaluation of the failure probability of the simulation performed by $\mathcal{I}$.
   (b) Before that time. Since the output of $\mathsf{TP}_{\textsc{Cmt}}$ has min-entropy $\beta(\lambda)$, super-logarithmic in $\lambda$, this happens with probability less than $\frac{q_h(\lambda)}{2^{\beta(\lambda)}}$ which is negligible being $2^{\beta(\lambda)}$ super-polynomial in $\lambda$.
2. Before producing the commitment $\textsc{Cmt}$ associated to a sign query for $\mathtt{m}$, $\mathcal{F}$ has performed another sign query for $\mathtt{m}$, and the output of $\mathsf{TP}_{\textsc{Cmt}}$ results to be the same $\textsc{Cmt}$. For $n \in [q_s(\lambda)]$ we define $\mathcal{X}_n \subset \mathcal{X}$ the set of commitments $\textsc{Cmt}$ generated in the previous $n-1$ sign queries, then the failure probability of the simulation during sign query $n$ for a collision of the commitment with the commitment of a previous query is:

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

To summarize, the probability that $\mathcal{I}$ is forced to overwrite the hash table $\mathsf{HT}$ 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 $\lambda$.

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

Once $\mathcal{I}$ has revealed its partial commitment executing $\mathsf{TP}^{\mathrm{Rel}}_{\mathrm{CMT}}$ on behalf of the actors in $J_h$, $\mathcal{F}$ knows the commitment $\textsc{Cmt}$ that will be used to create the signature on $\mathtt{m}$.

If $\mathcal{F}$ performs an hash query on $\mathtt{m}||\textsc{Cmt}$ before $\mathcal{F}$ reveals its partial commitments, $\mathcal{I}$ returns a random digest $d$ according to the simulation of the random oracle. However, when $\mathcal{F}$ reveals its partial commitments, $\mathcal{I}$ realizes that it can not set $\mathsf{HT}[\mathtt{m}||\textsc{Cmt}] = \textsc{Ch}$ where $\textsc{Ch}$ is the challenge received by the oracle $\mathcal{O}_{\mathcal{TID}}$, because it was previously set to $d$.

Therefore, we instruct the simulator $\mathcal{I}$ to store the challenge $\textsc{Ch}$ received from the oracle $\mathcal{O}_{\mathcal{TID}}$, and to rewind the forger $\mathcal{F}$ to the moment in which it performs the random oracle query for $\mathtt{m}||\textsc{Cmt}$. This time $\mathcal{I}$ sets the digest to $\textsc{Ch}$, and since the algorithm $\mathsf{TP}^{\mathrm{Rel}}_{\mathrm{CMT}}$ is deterministic, when $\mathcal{F}$ will complete the algorithm $\mathsf{TP}_{\mathrm{CMT}}$ releasing its partial commitments, the final commitment will result to be $\textsc{Cmt}$ as expected. This means that after the rewinding of $\mathcal{F}$ the simulation does not fail and is correct since the value $\textsc{Ch}$ was picked uniformly at random by $\mathcal{O}_{\mathcal{TID}}$.

**Observation 4.** If the commitment scheme used in $\mathsf{TP}_{\mathrm{CMT}}$ uses a random oracle (e.g. $\mathsf{Commit}(\mathsf{pp}, x) = H_{\mathrm{Com}}(x)$) then the simulation of $\mathcal{I}$ is even simpler and there is not need to rewind the adversary $\mathcal{F}$ in the case described in Item 1a. In fact, $\mathcal{F}$, in order to create its cryptographic commitment to $\textsc{Cmt}_i$ must send a random oracle query for $\textsc{Cmt}_i$ to a random oracle identified by $\mathcal{O}_{H_{\mathrm{Com}}}$. Since also this oracle must be simulated by $\mathcal{I}$, $\mathcal{I}$ already knows the partial commitments of $\mathcal{F}$ and can compute in advance the value $\textsc{Cmt}$ output of $\mathsf{TP}_{\mathrm{CMT}}$. This requirement toward $\mathsf{Commit}$ is required for an eventual proof in the UC framework, as noted in Section 1.1.

*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 $(\widehat{\text{CMT}}, \widehat{m})$ 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 $\mathbb{P}[\text{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda) = 1] = \epsilon(\lambda)$. If $\mathcal{I}$ simulates the experiment $\text{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda)$, $\mathcal{F}$ wins the simulated experiment, while interacting with $\mathcal{I}$, with the same non-negligible probability

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

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

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

which is non-negligible in the security parameter $\lambda$.

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 5.** The hypothesis about the structure of $\mathsf{TP}_{\text{CMT}}$ is crucial to reproduce the simultaneity of the message exchange among the parties involved in the creation of CMT. The simultaneity is important to prevent the corrupted parties from choosing adaptively their $\text{CMT}_i$, which might lead to key recovery attacks. Let us consider the following a $\mathsf{TP}_{\text{CMT}}$ in which each party choose randomly $\text{CMT}_i$ and publishes it, then all the parties set $\text{CMT} = \sum_{i \in J} \text{CMT}_i$.

This protocol is clearly not secure, indeed $\mathcal{F}$ might force two consecutive signing sessions on different messages to have the same $\textsc{Cmt}$, while the challenge will be different with high probability. This can cause attacks, in particular it leads to key recovery attacks if the protocol has special soundness as per [9].

**Lemma 2.** *[$\mathcal{TDS} \implies \mathcal{TID}$] Under the assumptions of Theorem 1, 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 $\mathrm{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}(\lambda)$ with non-negligible probability, then we build a forger $\mathcal{F}$ which uses $\mathcal{I}$ as a subroutine who wins the experiment $\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}(\lambda)$ 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 $\mathtt{pp}$ 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 $J$ actors to corrupt and before each impersonation query it chooses the set of $J_h$ 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 $J_h$ and $\mathcal{F}$ forwards it to $\mathcal{I}$ correctly simulating the multiparty protocol $\mathsf{TP}_{\textsc{Cmt}}$. When it comes the time for $\mathcal{F}$ to send the challenge $\textsc{Ch}$ to $\mathcal{I}$, $\mathcal{F}$ queries $\mathcal{O}_H(\cdot)$ on $(\mathtt{m}||\textsc{Cmt})$ and obtains $\textsc{Ch}$ 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 $(\mathtt{m}||\textsc{Cmt})$, 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 $\mathsf{TP}_{\textsc{Cmt}}$, $\mathcal{F}$ acts as a man in the middle in the execution of $\mathsf{TP}_{\textsc{Rsp}}$ 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 to $\mathcal{F}$ a commitment $\textsc{Cmt}^*$ as if it were produced by executing $\mathsf{TP}_{\textsc{Cmt}}$. Then $\mathcal{F}$ starts preparing its forgery by sending to $\mathcal{O}_H(\cdot)$ a hash query with input $(\mathtt{m}^*||\textsc{Cmt}^*)$ 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 $\textsc{Ch}^*$ 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 $\textsc{Rsp}^*$ that, if it is valid, allows $\mathcal{F}$ to produce a forgery of $\mathtt{m}$, namely $(\textsc{Cmt}^*, \textsc{Rsp}^*)$ which verifies since $H(\mathtt{m}^*||\textsc{Cmt}^*) = \textsc{Ch}^*$. □

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

### 4.2 Passive security

In this section we present the security result which considers passive adversaries. The ideas behind the proofs in this case are similar to the ones proposed in the active case, therefore a sketch of the proof is deferred to Appendix C.

**Theorem 2** (Passive security result)**.** *Let $\mathcal{TID}$ be a canonical threshold identification scheme, with $\mathcal{TID} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TP}_{\textsc{Cmt}}, \mathsf{TP}_{\textsc{Rsp}}, V)$ and let $\mathcal{TDS} = (\mathsf{Setup}, \mathsf{Key\text{-}Gen}, \mathsf{TSign}, \mathsf{Ver})$ be the associated signature scheme as per Construction 14. Then, assuming the ROM, the following implications hold:*

1. *($\mathcal{TID} \implies \mathcal{TDS}$): if $\mathcal{TID}$ is secure against impersonation under passive attacks and $\mathsf{TP}_{\textsc{Cmt}}$ is unpredictable as for Definition 15, then $\mathcal{TDS}$ is secure against active chosen-message attacks.*
2. *($\mathcal{TDS} \implies \mathcal{TID}$): if $\mathcal{TDS}$ is secure against impersonation under passive attacks, then $\mathcal{TID}$ is secure against active chosen-message attacks.*

**Observation 6.** We highlight the fact that in the passive case an unpredictable $\mathsf{TP}_{\textsc{Cmt}}$ is enough to guarantee a correct simulation execution by $\mathcal{I}$. There is not need to use a commit-release $\mathsf{TP}_{\textsc{Cmt}}$, because the attacker receives honest transcripts from the transcript oracle $\mathcal{O}^H_{\mathsf{View}-\mathcal{TDS}}$ simulated by the impersonator $\mathcal{I}$ and do not take part to the creation of such transcripts. Therefore, $\mathcal{F}$ does not learn the value $\textsc{Cmt}$, output of $\mathsf{TP}_{\textsc{Cmt}}$, in advance with respect to $\mathcal{I}$.

## 5 Threshold Sigma Protocols and Zero-Knowledge Properties

In the same way we defined canonical identification schemes adding a key generation algorithm to sigma protocols, we can adapt Definition 11 to define threshold Sigma protocol by removing the $\mathsf{Key\text{-}Gen}$ and $\mathsf{Setup}$ algorithms. We define threshold sigma protocols more formally as follows.

**Definition 18** (Threshold sigma protocol)**.** Let $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{Y}$ be a relation and $\mathsf{SS}$ a secure $(t, n)$-secret sharing scheme for elements of $\mathcal{W}$. A *threshold sigma protocol* $\Sigma$ for $\mathcal{R}$ and $\mathsf{SS}$ is defined by the tuple

$$\Sigma_{\mathcal{R},\mathsf{SS}} = (\mathsf{TP}_{\textsc{Cmt}}(\{w_i\}_J; \mathbf{R}), \mathsf{TP}_{\textsc{Rsp}}(\{w_i\}_J, \textsc{Cmt}, \textsc{Ch}; \mathbf{R}), V(y))$$

where the algorithms $\mathsf{TP}_{\textsc{Cmt}}, \mathsf{TP}_{\textsc{Rsp}}, V$ are defined as for *canonical* threshold identification schemes (Definition 11), where instead of having in input a share of secret key $\mathsf{sk}_i$, each party in $J$ has in input a share of the witness $w_i$.

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

This definition naturally extends the definition of sigma protocol presented in [9], Section 19.4. When $\mathcal{R}$ and $\mathsf{SS}$ are clear from the context we will refer to a threshold sigma protocol as $\Sigma$ instead of $\Sigma_{\mathcal{R},\mathsf{SS}}$.

In this section we give sufficient conditions on threshold sigma protocols such that the identification protocols obtained by equipping them with a setup and key generation algorithm is secure under passive and active attacks. In particular our goal is to define properties analogous to the standard zero knowledge and special soundness in the threshold setting.

For what concerns the zero knowledge properties, we need two different definitions, the first for the passive case and the second for the active one.

**Definition 19** (Passive Zero Knowledge)**.** Let $\Sigma$ be a threshold sigma protocol for a relation $R \subseteq \mathcal{W} \times \mathcal{Y}$ and secret sharing $\mathsf{SS}$ and challenge space $C$. Let $(w, y) \in \mathcal{R}$ and $\{w_i\}_{i \in n}$ be a secret sharing of $w$. Let $\mathcal{S}$ be an efficient probabilistic algorithm, called simulator that takes as input $(y, \textsc{Ch}) \in \mathcal{Y} \times C$, and a set $J$ of parties with $|J| < t$ such that $\mathcal{S}$ knows only the secret shares $\{w_i\}_{i \in J}$. We say that $\Sigma$ is *passive zero knowledge* if for any set of parties $J_{\mathcal{S}}$ such that $|J|+|J_{\mathcal{S}}| \geq t$ and $J_{\mathcal{S}} \cap J = \emptyset$, $\mathcal{S}$ can generate $(\textsc{Cmt}, \textsc{Rsp})$ and a transcript $\Pi$ for all messages exchanged in the execution of $\mathsf{TP}_{\textsc{Cmt}}$ and $\mathsf{TP}_{\textsc{Rsp}}$ by parties in $J \cup J_{\mathcal{S}}$, as well as the internal state of the parties in $J$ such that:

- $(\textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$ form an accepting conversation for $y$;
- for all $(w, y) \in \mathcal{R}$, $(\textsc{Cmt}, \textsc{Rsp}, \Pi) \xleftarrow{\$} \mathcal{S}(y, \textsc{Ch}, \{w_i\}_{i \in J})$ has the same distribution as that of transcript of a conversation between the parties in $J \cup J_{\mathcal{S}}$ acting honestly.

Informally speaking, a threshold sigma protocol is Passive Zero Knowledge if there exist a simulator that, receiving in advance the challenge $\textsc{Ch}$, is able to produce accepting conversations for all the parties in $J \cup J_{\mathcal{S}}$. The parties in $J$ represent the parties controlled by the adversary $\mathcal{I}$ in the experiment $\mathrm{Exp}^{p-imp}_{\mathcal{TID},\mathcal{I}}$, while the parties in $J_{\mathcal{S}}$ represent the parties that $\mathcal{I}$ asks to interact with during the training phase of the same experiment. Now we define the active case.

**Definition 20** (Active Zero Knowledge)**.** Let $\Sigma$ be a threshold sigma protocol for a relation $R \subseteq \mathcal{W} \times \mathcal{Y}$ and challenge space $C$. Let $(w, y) \in \mathcal{R}$ and $\{w_i\}_{i \in n}$ be a secret sharing of $w$.

Let $\mathcal{S}$ be an efficient probabilistic algorithm, called simulator that takes as input $(y, \textsc{Ch}) \in \mathcal{Y} \times C$, and a $\{w_i\}_{i \in J}$ for a set $J$ of parties with $|J| < t$.

We say that $\Sigma$ is *active zero knowledge* if $\mathcal{S}$, controlling any set of parties $J_{\mathcal{S}}$ such that $|J| + |J_{\mathcal{S}}| \geq t$ and $J_{\mathcal{S}} \cap J = \emptyset$, can interact with an adversary $\mathcal{A}$ controlling the parties in $J$ executing the sigma protocol $\Sigma$ producing $(\textsc{Cmt}, \textsc{Rsp})$ and transcript $\Pi$ for all the messages sent by party in $J_{\mathcal{S}}$ to party in $J$ such that

– if $\mathcal{A}$ acts honestly, $(\textsc{Cmt}, \textsc{Ch}, \textsc{Rsp})$ is an accepting conversation for $y$.
– for all $(w, y) \in \mathcal{R}$, if the following two distributions

$$(\textsc{Cmt}, \textsc{Rsp}, \Pi) \xleftarrow{\$} \mathcal{S}^{\mathcal{A}}(y, \textsc{Ch}, \{w_i\}_{i \in J})$$

$$(\textsc{Cmt}_h, \textsc{Rsp}_h, \Pi_h) \xleftarrow{\$} \mathcal{A}^{\{P_i\}_{J_\mathcal{S}}}(\textsc{Ch}, \{w_i\}_{i \in J})$$

are indistinguishable, where $\mathcal{A}^{\{P_i\}_{J_\mathcal{S}}}$ denotes a real execution between the adversary $\mathcal{A}$ and honest parties in $J_\mathcal{S}$, with challenge $\textsc{Ch}$.

The key difference is that $\mathcal{S}$ is not allowed to compute the transcript by itself but instead it need to be able to simulate a real execution of the protocol interacting with an adversary.

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

**Definition 21** (Special Soundness)**.** Let $\Sigma$ be a threshold sigma protocol for a relation $R \subseteq \mathcal{W} \times \mathcal{Y}$. We say that $\Sigma$ 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 $y$.

**Theorem 3.** *Let $\Sigma = (\mathsf{TP}_{\textsc{Cmt}}, \mathsf{TP}_{\textsc{Rsp}}, V)$ be a $(t, n)-$ threshold Sigma protocol for a relation $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{Y}$, and secret sharing* $\mathsf{SS}$*, with super-polynomial challenge space $C$. Let*

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

*be the threshold identification scheme obtained by equipping $\Sigma$ with the* $\mathsf{Setup}$ *and a one-way key generation algorithm* $\mathsf{Key\text{-}Gen}$*. If $\Sigma$ provides passive (active) zero knowledge and special soundness then the $\mathcal{TID}$ is secure against passive (active) impersonation attacks.*

We show only the active case, the passive case can be done in the same way.

*Proof.* We want to show that if there exist an adversary $\mathcal{A}$ able to win the $\mathrm{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{A}}$ game, then it is possible to build an attacker $\mathcal{S}$ that is able to win the $\mathrm{Exp}^{\text{One-way}}_{\mathsf{Key\text{-}Gen},\mathcal{S}}$ game.

Firstly, $\mathcal{S}$ receives a challenge $y \in \mathcal{Y}$, having the goal of finding $w \in \mathcal{W}$ such that $(w, y) \in \mathcal{R}$. $\mathcal{S}$ sets $y$ as the public key $\mathsf{pk}$ for the $\mathrm{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{A}}$ and sends it to $\mathcal{A}$, who answers with the set $J$ of participants it desires to corrupt.

Then $\mathcal{S}$ sends to $\mathcal{A}$ random shares $w_i$ to simulate the secret sharing of $w' \in \mathcal{W}$ such that $(w', y) \in \mathcal{R}$. Since the secret sharing scheme is secure according to

Definition 8, this is indistinguishable from an execution of a real secret sharing, since $\mathcal{A}$ controls less than $t$ parties. Then $\mathcal{S}$ act as an oracle $\mathcal{O}_{\mathcal{TID}}(\cdot)$ for $\mathcal{A}$ and simulates the execution of $\mathcal{TID}$, that is possible thanks to the active zero knowledge property.

$\mathcal{A}$ will eventually perform a successful impersonation. At this point $\mathcal{S}$ rewinds $\mathcal{A}$ and changes the challenge sent. Since the challenge space $C$ is super-polynomial, with non-negligible probability this yields the two required accepting conversation (by the Forking Lemma [29] that we state in Appendix E), thus $\mathcal{S}$ can use the extractor $\mathcal{E}$ from the special soundness Definition 21 to extract a witness $w$ breaking the one-way assumption on the key generation. □

## 6 Security Proof of Schnorr Threshold Signature Sparkle

In this section we put together all the notions that we have introduced and we apply the theorems that we have proved in the previous sections. In particular, we show how it is possible to design a threshold signature and prove its security using our framework starting from a threshold sigma protocol. We design a threshold sigma protocol for a relation $\mathcal{R}$ and a secure secret sharing $\mathsf{SS}$, and we prove that the hypothesis of Theorem 3 are satisfied, therefore if we equip $\Sigma$ with a $\mathsf{Setup}$ and a one-way $\mathsf{Key\text{-}Gen}$ algorithm consistent with $\Sigma$, we obtain a threshold identification scheme $\mathcal{TID}$ secure against passive (active) attacks. Then we show that if the $\mathsf{TP}_{\mathrm{CMT}}$ is unpredictable (commit-release), then the hypothesis of Theorem 2 (Theorem 1) are satisfied. Therefore the threshold signature obtained by applying the distributed Fiat-Shamir Transform to $\mathcal{TID}$ is unforgeable against passive (active) attacks.

The example that we analyse in this section is the recent threshold signature of Schnorr (Sparkle) described by Crites et al. in [18]. We provide an alternative proof of Theorem 1 of [18] ("Sparkle is statically secure under DL in the ROM") using our framework. We recall the scheme of Sparkle in Appendix D, Figure 8, using the same notation used in their paper [18], and we also provide additional details about how to obtain it from the protocol that we present below.

The threshold sigma protocol that we present in Figure 5 is a threshold sigma protocol for relation $\mathcal{R} = \{(w, y = g^w) | w \in Z_p, \mathbb{G} =< g >\} \subset \mathbb{Z}_p \times \mathbb{G}$ and Shamir secret sharing $\mathsf{SS}$, where $\mathbb{G}$ is a cyclic group with generator $g$ of order $p$, a $\lambda$ bits prime number. The challenge space is $C = \mathbb{Z}_p$ which is super-polynomial in the security parameter $\lambda$. This, equipped with the $\mathsf{Setup}$ and the one-way $\mathsf{Key\text{-}Gen}$ 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 passive (active) chosen message attacks.*

*Proof.* Our goal is to use Theorem 3, from which Theorem 1 follows immediately.

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

```
$\mathsf{TP}^{\mathrm{Com}}_{\mathrm{CMT}}(\mathtt{ssid}, \{w_i\}_{i\in\mathbb{S}}; \mathbf{R}) \to \{\mathrm{CMT}_i\}_{i\in\mathbb{S}}$
1 : // Each party $k$ runs it
2 : $r_k \xleftarrow{\$} \mathbb{Z}_p$
3 : $R_k \leftarrow g^r$
4 : // Compute the commitment $\mathsf{com}_k$
5 : $\mathsf{com}_k \leftarrow \mathsf{H}_{\mathsf{com}}(\mathtt{ssid}, \mathbb{S}, R_k)$
6 : return $\mathsf{com}_k$
```

```
$\mathsf{TP}_{\mathrm{RSP}}(\{w_i\}_{i\in\mathbb{S}}, \mathrm{CMT}, \mathrm{CH}) \to (\mathrm{RSP})$
1 : // Each party $k$ runs it
2 : $z_k \leftarrow r_k + \mathrm{CH}(\lambda_k x_k)$
3 : // $\lambda_k$ is the Lagrange
4 : // coefficient of $k$ w.r.t. $\mathbb{S}$
5 : Party $P_k$ sends $z_k$
6 : $z \leftarrow \sum_{i\in\mathbb{S}} z_i$
7 : return $(R, z) \leftarrow \sigma$
```

```
$\mathsf{TP}^{\mathrm{Rel}}_{\mathrm{CMT}}(\mathtt{ssid}, \{w_i\}_{i\in\mathbb{S}}, \{\mathsf{com}_i\}_{i\in\mathbb{S}}) \to \mathrm{CMT}$
1 : // Each party $k$ runs it
2 : Party $P_k$ sends $R_k$
3 : If $\exists j \in \mathbb{S}$ s.t.
4 : $\mathsf{com}_j \neq \mathsf{H}_{\mathsf{com}}(\mathtt{ssid}, \mathbb{S}, R_j)$
5 : return $\perp$
6 : $R = \prod_{i\in\mathbb{S}} R_i$
7 : return $\mathrm{CMT} \leftarrow R$
8 : // CMT is sent to the verifier
9 : // who returns the challenge CH
```

```
$V(y, \sigma) \to 0/1$
1 : Parse $(R, z) \leftarrow \sigma$
2 : if $Ry^{\mathrm{CH}} = g^z$ return 1
3 : Else return 0
```

**Fig. 5.** Threshold sigma protocol for Sparkle.

**Special soundness.** The special soundness property is trivial and follows immediately from the special soundness of the standard Schnorr protocol [30]. Indeed, suppose to have two accepting transcripts $(R, \mathrm{CH}, z)$ and $(R, \mathrm{CH}', z')$ with $\mathrm{CH} \neq \mathrm{CH}'$. Then it would be possible to compute the discrete logarithm of $\mathsf{pk} = y$ by simply computing $w = (z - z')(\mathrm{CH} - \mathrm{CH}')^{-1}$.

**Active zero knowledge.** To prove that the protocol is active zero knowledge, we must show that it can be simulated by a simulator $\mathcal{S}$ taking in input $(y = g^w, \mathrm{CH}^*)$ 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 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 $\textsc{Ch}^*$ is the challenge it received in input and $\lambda_j$ 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 $y^{-\textsc{Ch}^*} = g^{w(-\textsc{Ch}^*)}$, therefore $y^{-\textsc{Ch}^*} \prod_{j=1}^{t-1} g^{\lambda_j w_j \textsc{Ch}^*} = g^{\lambda_t w_t(-\textsc{Ch}^*)}$, then it holds that $g^{z_t} = R_t g^{\lambda_t w_t \textsc{Ch}^*}$.
This means that the transcript $(R_t, \textsc{Ch}^*, z_t)$ is valid and, being $z_t$ sampled uniformly at random, and $R_t$ being univocally determined from $(z_t, \textsc{Ch}^*)$, $(R_t, \textsc{Ch}^*, z_t)$ is indistinguishable from an honest transcript (generated starting from $R_t$).
Finally $\mathcal{S}$ executes $\mathsf{TP}^{\text{Com}}_{\textsc{Cmt}}$ computing $\mathsf{com}_t = \mathsf{H}_{\mathsf{com}}(\mathtt{m}, \mathcal{S}, R_t)$, then it executes $\mathsf{TP}^{\text{Rel}}_{\textsc{Cmt}}$ by releasing $R_t$. The commitments are aggregated computing $R$, then the challenge $\textsc{Ch}^*$ will be used as the challenge of the transcript and $\mathcal{S}$ simulates the algorithm $\mathsf{TP}_{\textsc{Rsp}}$ by broadcasting the responses $\textsc{Rsp}_t = z_t$ it sampled randomly at the beginning of the simulation.
Note that the transcripts $(R, \textsc{Ch}^*, z)$, 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 20.

**Passive zero knowledge.** The proof is basically the same as the one for the active zero knowledge property. However the simulator $\mathcal{S}$ acts honestly on behalf of the parties $P_1, \ldots, P_{t-1}$ of which it knows the shares of witness.

By equipping the threshold sigma protocol with the $\mathsf{Setup}$ and $\mathsf{Key\text{-}Gen}$ of Sparkle we obtain a threshold identification scheme $\mathcal{TID}$ which has a one-way $\mathsf{Key\text{-}Gen}$, a super-polynomial challenge space and is special sound, active zero-knowledge and passive zero knowledge. Therefore by Theorem 3, $\mathcal{TID}$ is secure against active and passive impersonation attacks.

It remains to prove that $\mathcal{TID}$ has a commit-release $\mathsf{TP}_{\textsc{Cmt}}$. Indeed $\mathsf{TP}^{\text{Com}}_{\textsc{Cmt}}$ does not require any interaction between the parties and outputs $\mathsf{H}_{\mathsf{com}}(\mathtt{ssid}, S, R_k)$ that is a one-way commitment as long as $\mathsf{H}_{\mathsf{com}}$ is a secure cryptographic hash function. The function used to reconstruct the commitment $\textsc{Cmt} = R$ is $R = \prod_{i\in\mathbb{S}} R_i$ where the computations are executed in $\mathbb{G}$, therefore if at least one party in $\mathbb{S}$ is honest, the value $R$ will be uniformly distributed in $\mathbb{G}$. Moreover, being $\mathsf{H}_{\mathsf{com}}$ a secure cryptographic hash function, $\mathsf{TP}^{\text{Rel}}_{\textsc{Cmt}}$ is a deterministic protocol.

By applying Theorem 1 we prove that Sparkle, the digital signature obtained by applying the distributed Fiat-Shamir transform, is unforgeable against active chosen message attacks. The passive case follows trivially. □

## 7 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 6.

*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 [3,26,17], many protocols are also proved secure in the adaptive case, where the adversary can, at any time, corrupt parties and learn their state [18]. It would be interesting to extend our analysis to the adaptive case. The structure of the proof of Theorem 1 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 on 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.

## 8 Acknowledgments

The first author is member of GNSAGA of INdAM and acknowledges support from TIM S.p.A. through the Ph.D. scholarship.

The second author acknowledges support from Eustema S.p.A. through the PhD scholarship.

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., Longo, R., Meneghetti, A., Sala, M.: Threshold ECDSA with an Offline Recovery Party. Mediterranean Journal of Mathematics **19**(1), 1–29 (2022)
3. 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)
4. 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`
5. 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`
6. Bellare, M., Rogaway, P.: The security of triple encryption and a framework for code-based game-playing proofs. In: Vaudenay, S. (ed.) Advances in Cryptology - EUROCRYPT 2006. pp. 409–426. Springer Berlin Heidelberg, Berlin, Heidelberg (2006)
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. Biasse, J.F., Micheli, G., Persichetti, E., Santini, P.: Less is more: code-based signatures without syndromes. In: Progress in Cryptology-AFRICACRYPT 2020: 12th International Conference on Cryptology in Africa, Cairo, Egypt, July 20–22, 2020, Proceedings 12. pp. 45–65. Springer (2020)
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 (2001). https://doi.org/10.1109/SFCS.2001.959888
14. Canetti, R.: Universally composable signature, certification, and authentication. In: Proceedings. 17th IEEE Computer Security Foundations Workshop, 2004. pp. 219–233 (2004). https://doi.org/10.1109/CSFW.2004.1310743
15. Canetti, R., Gennaro, R., Goldfeder, S., Makriyannis, N., Peled, U.: Uc non-interactive, proactive, threshold ecdsa with identifiable aborts. In: Proceedings of the 2020 ACM SIGSAC Conference on Computer and Communications Security. p. 1769–1787. CCS '20, Association for Computing Machinery, New York, NY, USA (2020). https://doi.org/10.1145/3372297.3423367, https://doi.org/10.1145/3372297.3423367
16. Chou, T., Niederhagen, R., Persichetti, E., Randrianarisoa, T.H., Reijnders, K., Samardjiska, S., Trimoska, M.: Take your meds: Digital signatures from matrix code equivalence. In: International Conference on Cryptology in Africa. pp. 28–52. Springer (2023)
17. 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)
18. Crites, E., Komlo, C., Maller, M.: Fully adaptive schnorr threshold signatures. Cryptology ePrint Archive (2023)
19. 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)
20. 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)
21. 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
22. Katz, J., Lindell, Y.: Introduction to modern cryptography book (2020)
23. 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)
24. Komlo, C.: A note on various forking lemmas, https://www.chelseakomlo.com/assets/content/notes/Forking-Lemma-Variants.pdf
25. 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
26. 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
27. 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)

28. 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)
29. Pointcheval, D., Stern, J.: Security arguments for digital signatures and blind signatures. Journal of cryptology **13**, 361–396 (2000)
30. Schnorr, C.: Efficient signature generation by smart cards. Journal of Cryptology **4**, 161–174 (01 1991). https://doi.org/10.1007/BF00196725

## A Hiding and one-way commitments

A commitment scheme is hiding if no adversary can win the hiding experiment with non-negligible probability. The hiding experiment is presented below.

- The adversary $\mathcal{A}$ of the hiding game receives the public parameters $\mathtt{pp}$ generated by the challenger $\mathcal{C}$;
- $\mathcal{A}$ chooses two messages $x_0, x_1$ from the message space $X$ and sends them to $\mathcal{C}$;
- $\mathcal{C}$ randomly picks a bit $b$, computes a commitment $c$ to $x_b$ and sends it to the adversary;
- $\mathcal{A}$ returns a bit $b'$ and wins the game if $b' = b$.

We say that $\mathcal{A}$ has non-negligible advantage if $\mathbb{P}(b' = b | b' \overset{\$}{\leftarrow} \mathcal{A}(\mathtt{pp}, c)) - \frac{1}{2} > \nu(\lambda)$ for a function $\nu()$ non-negligible in the security parameter $\lambda$.

We now show that an adversary $\mathcal{A}'$ who wins the experiment $\mathrm{Exp}_{\mathrm{Com}}^{\text{One-way}}(\lambda)$ with non-negligible advantage (i.e. with non-negligible probability) $\nu(\lambda)$ can be used as a subroutine of an adversary $\mathcal{A}$ capable to win the hiding experiment with non-negligible advantage.

As we present in Figure 6, after receiving the public parameters $\mathtt{pp}$, and forwarding it to $\mathcal{A}'$, $\mathcal{A}$ sends to the challenger of the hiding game two random messages $x_0, x_1$ and receives a commitment to $x_b$ for a random bit $b$. $\mathcal{A}$ can simulate the challenger of the one-way experiment by sending the commitment to $\mathcal{A}'$ which returns $x'$ to $\mathcal{A}$. If $x' = x_0$ or $x' = x_1$, $\mathcal{A}$ returns 0 or 1 respectively, to the challenger of the hiding game. Otherwise $\mathcal{A}$ picks a random bit $b'$ and returns $b'$ to the challenger of the hiding experiment.

$\mathcal{A}$ correctly simulates the challenger of the one-way experiment since it sends the public parameters $\mathtt{pp}$ generated by the challenger of the hiding experiment, then it sends to $\mathcal{A}'$ the commitment to a random message (since both $x_0$ and $x_1$ are picked uniformly at random).

Now we show that $\mathcal{A}$ wins the hiding experiment with non-negligible advantage. When $x' \neq x_0$ and $x' \neq x_1$, $\mathcal{A}'$ has lost the one-way game, and this happens with probability at most $p_1 < 1 - \nu(\lambda)$ for a non-negligible function $\nu(\lambda)$, being $\nu(\lambda)$ the probability that $\mathcal{A}'$ guesses the right message and wins the one-way game. With probability greater then $\nu$ instead $\mathcal{A}'$ returns $x' = x_B$, $B \in \{0, 1\}$. In this case either $\mathcal{A}'$ wins the one-way experiment, and also $\mathcal{A}$ wins the hiding experiment, or it looses its experiment but guesses $x_{1-b}$, the other message that $\mathcal{A}$ randomly picked during the hiding experiment, which happens with negligible probability. In fact, being $x_{1-b}$ randomly picked independently from $x_b$, the probability that $\mathcal{A}'$ outputs $x' = x_{1-b}$ is $\frac{1}{N}$ where $N = |\mathcal{X}|$ has super-polynomial size, making the probability $\frac{1}{N}$ negligible.

To summarize,

$$\begin{aligned}
\mathbb{P}(\mathcal{A}\text{ wins}) &= \mathbb{P}((b = b' \wedge ((x' \neq x_0) \wedge (x' \neq x_1))) \vee x' = x_b) = \\
&= \mathbb{P}(b = b' \wedge ((x' \neq x_0) \wedge (x' \neq x_1))) + \mathbb{P}(x' = x_b) = \\
&= \mathbb{P}(b = b' | (x' \neq x_0) \wedge (x' \neq x_1))\mathbb{P}(((x' \neq x_0) \wedge (x' \neq x_1))) + \nu(\lambda) = \\
&= \frac{1}{2}\left(1 - \left(\nu(\lambda) + \frac{1}{N}\right)\right) + \nu(\lambda) = \frac{1}{2} + \frac{1}{2}\nu(\lambda) - \frac{1}{2N}
\end{aligned}$$

therefore the advantage of $\mathcal{A}$ in winning the hiding experiment is $\frac{1}{2}\nu(\lambda) - \frac{1}{N}$ which is non-negligible, being $N$ super-polynomial and $\nu(\lambda)$ non-negligible.

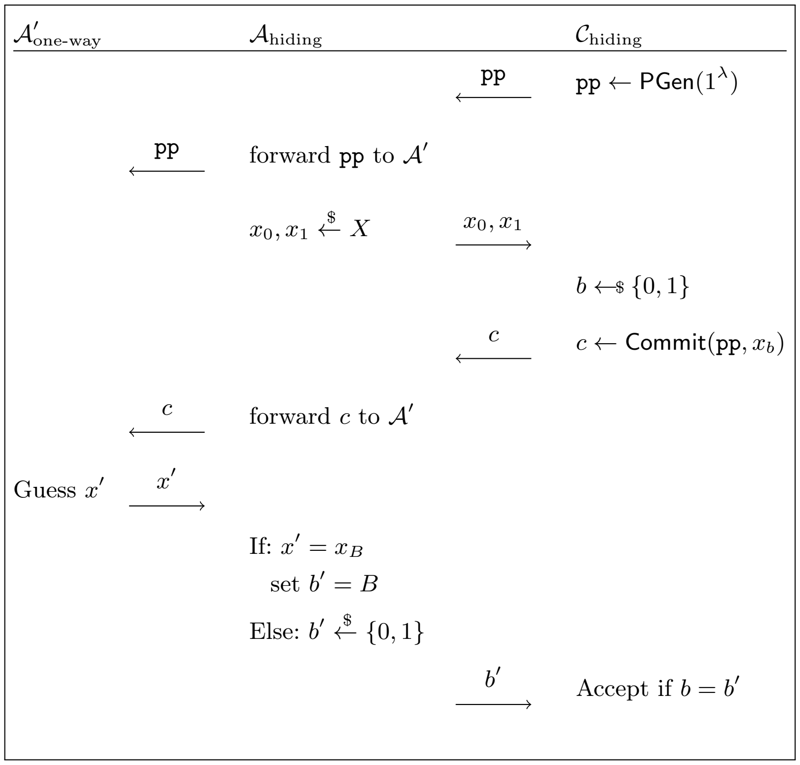
$\mathcal{A}'_{\text{one-way}}$
$\mathcal{A}_{\text{hiding}}$
$\mathcal{C}_{\text{hiding}}$
$\xleftarrow{\mathsf{pp}}$ $\mathsf{pp} \leftarrow \mathsf{PGen}(1^\lambda)$
$\xleftarrow{\mathsf{pp}}$ forward $\mathsf{pp}$ to $\mathcal{A}'$
$x_0, x_1 \overset{\$}{\leftarrow} X$ $\xrightarrow{x_0, x_1}$
$b \leftarrow\!\$\ \{0,1\}$
$\xleftarrow{c}$ $c \leftarrow \mathsf{Commit}(\mathsf{pp}, x_b)$
$\xleftarrow{c}$ forward $c$ to $\mathcal{A}'$
Guess $x'$ $\xrightarrow{x'}$
If: $x' = x_B$
set $b' = B$
Else: $b' \overset{\$}{\leftarrow} \{0,1\}$
$\xrightarrow{b'}$ Accept if $b = b'$

**Fig. 6.** Description of the adversary $\mathcal{A}$ of the hiding game which uses the adversary $\mathcal{A}'$ of the one-way experiment as a subroutine.

## B Reduction of Lemma 1

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

In Figure 7 we represent the impersonator $\mathcal{I}$ who executes the experiment $\mathrm{Exp}^{\text{a-imp}}_{\mathcal{TID},\mathcal{I}}$. $\mathcal{I}$ 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 $\mathrm{Exp}^{\text{a-uf-cma}}_{\mathcal{TDS},\mathcal{F}}$, 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.

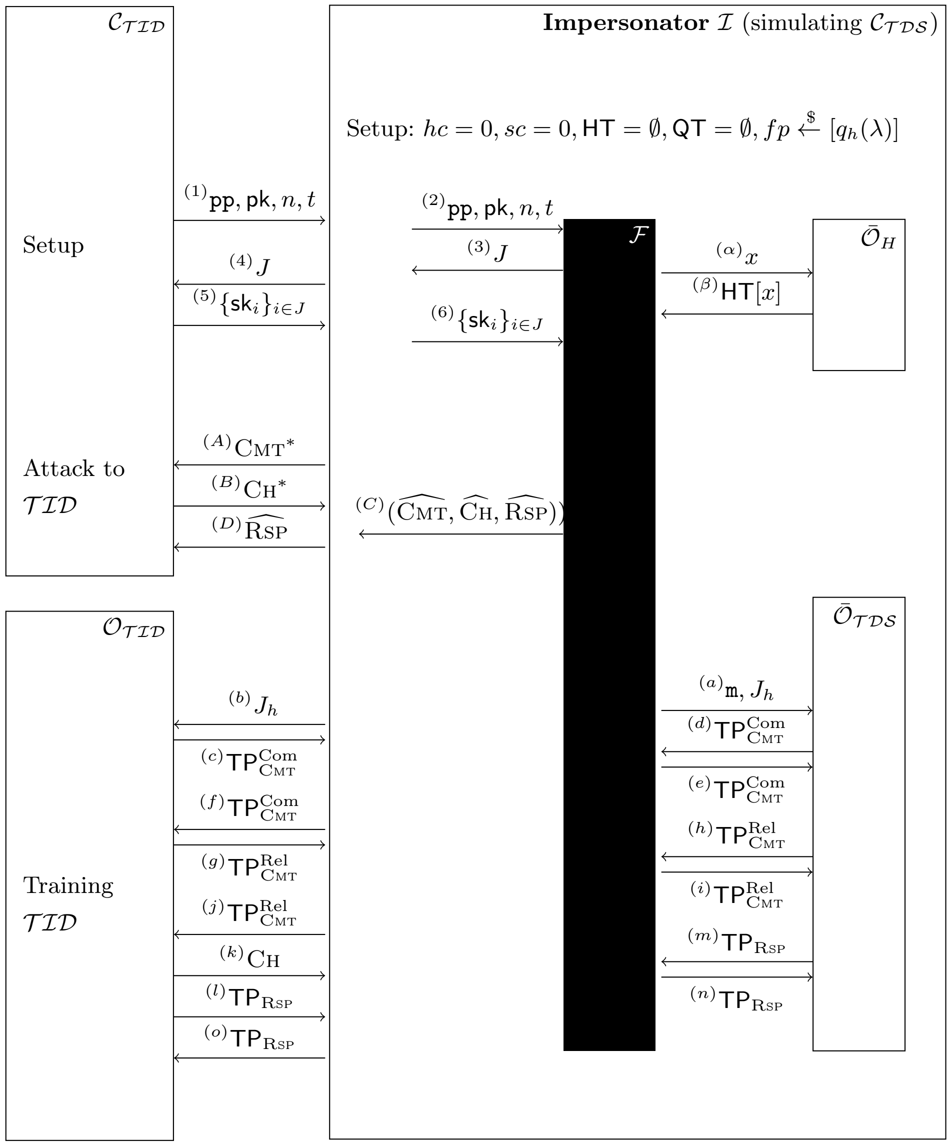
$\mathcal{C}_{\mathcal{TID}}$
Impersonator $\mathcal{I}$ (simulating $\mathcal{C}_{\mathcal{TDS}}$)
Setup: $hc = 0, sc = 0, \mathsf{HT} = \emptyset, \mathsf{QT} = \emptyset, fp \xleftarrow{\$} [q_h(\lambda)]$
Setup
$^{(1)}\mathsf{pp}, \mathsf{pk}, n, t$
$^{(2)}\mathsf{pp}, \mathsf{pk}, n, t$
$\mathcal{F}$
$\bar{\mathcal{O}}_H$
$^{(3)}J$
$^{(4)}J$
$^{(5)}\{\mathsf{sk}_i\}_{i \in J}$
$^{(6)}\{\mathsf{sk}_i\}_{i \in J}$
$^{(\alpha)}x$
$^{(\beta)}\mathsf{HT}[x]$
Attack to $\mathcal{TID}$
$^{(A)}\mathrm{CMT}^*$
$^{(B)}\mathrm{CH}^*$
$^{(C)}(\widehat{\mathrm{CMT}}, \widehat{\mathrm{CH}}, \widehat{\mathrm{RSP}})$
$^{(D)}\widehat{\mathrm{RSP}}$
$\mathcal{O}_{\mathcal{TID}}$
$\bar{\mathcal{O}}_{\mathcal{TDS}}$
Training $\mathcal{TID}$
$^{(a)}\mathtt{m}, J_h$
$^{(b)}J_h$
$^{(c)}\mathsf{TP}^{\mathrm{Com}}_{\mathrm{CMT}}$
$^{(d)}\mathsf{TP}^{\mathrm{Com}}_{\mathrm{CMT}}$
$^{(e)}\mathsf{TP}^{\mathrm{Com}}_{\mathrm{CMT}}$
$^{(f)}\mathsf{TP}^{\mathrm{Com}}_{\mathrm{CMT}}$
$^{(g)}\mathsf{TP}^{\mathrm{Rel}}_{\mathrm{CMT}}$
$^{(h)}\mathsf{TP}^{\mathrm{Rel}}_{\mathrm{CMT}}$
$^{(i)}\mathsf{TP}^{\mathrm{Rel}}_{\mathrm{CMT}}$
$^{(j)}\mathsf{TP}^{\mathrm{Rel}}_{\mathrm{CMT}}$
$^{(k)}\mathrm{CH}$
$^{(l)}\mathsf{TP}_{\mathrm{RSP}}$
$^{(m)}\mathsf{TP}_{\mathrm{RSP}}$
$^{(n)}\mathsf{TP}_{\mathrm{RSP}}$
$^{(o)}\mathsf{TP}_{\mathrm{RSP}}$

**Fig. 7.** High level description of the impersonator $\mathcal{I}$ using a forger $\mathcal{F}$ as a subroutine.

## C Proof of Theorem 2

In this section we will sketch the proof of Theorem 2. The proof is very similar to the one of Theorem 1. We split the proof in two parts, one for each side of the implication.

**Lemma 3** **($(\mathcal{TID} \implies \mathcal{TDS})$).** *Under the assumptions of Theorem 2, if $\mathcal{TID}$ is secure against impersonation under passive attacks and $\mathsf{TP}_{\textsc{Cmt}}$ is unpredictable as for Definition 15, then $\mathcal{TDS}$ is secure against passive chosen-message attacks.*

*Proof Sketch.* We assume that there exists a forger $\mathcal{F}$ with non-negligible advantage in winning the $\mathrm{Exp}^{\text{p-uf-cma}}_{\mathcal{TDS},\mathcal{F}}$. Without loss of generality we require that $\mathcal{F}$ satisfies the following properties, as it was required in Lemma 1:

- all of its hash queries have the form $\textsc{Cmt}||\mathtt{m}$ with $\textsc{Cmt}, \mathtt{m} \in \{0,1\}^*$;
- 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.

Now we show how to define the impersonator $\mathcal{I}$ starting from the forger $\mathcal{F}$. $\mathcal{I}$ will act as a "man-in-the-middle" between $\mathcal{F}$ and the challenger $\mathcal{C}_{\mathcal{TID}}$. In particular it forwards the initial message containing the public parameters and the public keys of the parties received by $\mathcal{C}_{\mathcal{TID}}$ to $\mathcal{F}$. Then, when $\mathcal{F}$ decides the set $J$ to corrupt during the experiment, and during each sign query the set $J_h$ of honest parties who contribute, $\mathcal{I}$ makes the same choices. Now $\mathcal{I}$ needs to simulate the sign query and the hash query. To do so, $\mathcal{F}$ initialize an empty hash table $\mathsf{HT}$ and:

- when $\mathcal{F}$ performs an hash query with input $x \in \{0,1\}^*$, $\mathcal{I}$ returns $\mathsf{HT}[x]$ if it is defined, otherwise it returns a random value and saves it in $\mathsf{HT}[x]$ for all but one query. In that specific query for $x = \textsc{Cmt}^*||m^*$, the $fp$-th query, where $fp$ is randomly selected in $[q_h(\lambda)]$ at the beginning of the experiment, $\mathcal{F}$ forwards $\textsc{Cmt}^*$ to $\mathcal{C}_{\mathcal{TID}}$ as part of the impersonation attempt of $\mathrm{Exp}^{\text{p-imp}}_{\mathcal{TID},\mathcal{A}}(\lambda)$ and get a challenge $\textsc{Ch}^*$. It then returns $\textsc{Ch}^*$ to $\mathcal{F}$ and updates $\mathsf{HT}$ setting $\mathsf{HT}[\textsc{Cmt}^*||m^*] = \textsc{Ch}^*$.
- When $\mathcal{F}$ performs a sign query for $\mathtt{m}$ and $J_h$, $\mathcal{I}$ queries the oracle $\mathcal{O}_{\mathsf{View}-\mathcal{TID}}$ who provides it with a transcript of an identification scheme execution performed by the parties in $J \cup J_h$. Being $\textsc{Cmt}$ and $\textsc{Ch}$ the commitment and the challenge included in the transcript, $\mathcal{I}$ updates the hash table $\mathsf{HT}$ setting $\mathsf{HT}[\textsc{Cmt}||\mathtt{m}] = \textsc{Ch}$ and forwards it to $\mathcal{F}$.

The simulation may fail if $\mathcal{I}$ must overwrite the hash table $\mathsf{HT}$ but this happens with negligible probability being $\mathsf{TP}_{\textsc{Cmt}}$ unpredictable, therefore the commitments are generated with super-logarithmic min-entropy. After at most $q_h$ hash queries and $q_s$ sign queries, $\mathcal{F}$ will eventually output a forgery $(\widehat{\textsc{Cmt}}, \widehat{\textsc{Rsp}})$ of $\widehat{\mathtt{m}}$. If $\mathcal{F}$ successfully produce a forgery and $\mathcal{I}$ correctly guessed the hash query corresponding to it, i.e. $\textsc{Cmt}^* = \widehat{\textsc{Cmt}}$ and $\mathtt{m}^* = \widehat{\mathtt{m}}$, $\mathcal{I}$ also wins the impersonation game.

□

**Observation 7.** Note that, since the transcript oracle generates honest transcripts, where all the parties involved behave honestly, we would not necessarily need an unpredictable $\mathsf{TP}_{\text{CMT}}$, which guarantees a sufficiently random output if at least one party is honest, but it would be enough a $\mathsf{TP}_{\text{CMT}}$ that returns a random output when all the parties involved are honest (e.g. the toy $\mathsf{TP}_{\text{CMT}}$ described in Observation 5).

**Lemma 4 (($\mathcal{TDS} \implies \mathcal{TID}$)).** *Under the assumptions of Theorem 2, if $\mathcal{TDS}$ is secure 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 $\text{Exp}^{\text{p-imp}}_{\mathcal{TID},\mathcal{A}}(\lambda)$ with non-negligible probability, then we build a forger $\mathcal{F}$ which uses $\mathcal{I}$ as a subroutine who wins the experiment $\text{Exp}^{\text{p-uf-cma}}_{\mathcal{TDS},F}(\lambda)$ with non-neligible probability.

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

*Training phase.* The impersonator $\mathcal{I}$ selects the set $J$ of parties it wants to corrupt and sends it to $\mathcal{F}$, who makes the same choice and sends it to $\mathcal{O}_{\mathsf{View}-\mathcal{TDS}}(\cdot)$. The oracle sends to $\mathcal{F}$ the secret keys of the parties in $J$, and $\mathcal{F}$ forwards it to $\mathcal{I}$ who can start the training phase in which it asks $\mathcal{F}$ for transcripts of the identification scheme executed with the parties in $J_h \subset [n] \setminus J$ such that $|J \cup J_h| = t$. $\mathcal{F}$ simulates the oracle $\mathcal{O}_{\mathsf{View}-\mathcal{TID}}(\mathsf{pk}, \mathsf{pp})$ by querying, for each identification transcript query, a digital signature query to $\mathcal{O}_{\mathsf{View}-\mathcal{TDS}}(\cdot)$ for message $\mathtt{m} \in \{0,1\}^\lambda$. The oracle $\mathcal{O}_{\mathsf{View}-\mathcal{TDS}}(\mathsf{pk}, \mathsf{pp})$ answers with a signature of $\mathtt{m}$ together with the public messages exchanged between the parties in $J$ and $J_h$ and the state of the parties in $J$, the ones corrupted by $\mathcal{F}$. $\mathcal{F}$ forwards the messages received from $\mathcal{O}_{\mathsf{View}-\mathcal{TDS}}(\mathsf{pk}, \mathsf{pp})$ which are indistinguishable from a real execution of the threshold identification scheme since, according to Definition 14 the creation of $\text{CMT}$ is exactly the same as in the associated canonical identification scheme (see Definition 11), the challenge is the output of a random oracle on input $(\text{CMT}||\mathtt{m})$ which is a random element, and the response is again computed as in the canonical identification scheme. $\mathcal{F}$ every time it must provide $\mathcal{I}$ with a new identification transcript must query the sign oracle with a new sign query, every time for a different message. One way to do this is the following: $\mathcal{F}$ can treat the message $\mathtt{m}$ used in the sign query by $\mathcal{F}$ as an element in $\mathbb{Z}_{2^\lambda}$ and for new identification transcript queries performed by $\mathcal{I}$, $\mathcal{F}$ always updates $\mathtt{m}$ setting $\mathtt{m} \leftarrow \mathtt{m} + 1$. Therefore, for each transcript query from $\mathcal{I}$, $\mathcal{F}$ will provide it with a transcript with challenge which is the output of the random oracle $\mathcal{O}_H(\cdot)$ always on distinct inputs.

*Exploit of $\mathcal{I}$'s impersonation.* When $\mathcal{I}$ starts its impersonation attempt, it sends a commitment $\text{CMT}^*$. $\mathcal{F}$ computes a fresh new $\mathtt{m}^*$ and sends a random oracle query to $\mathcal{O}_H(\cdot)$ with input $\textsc{Cmt}^*||\mathtt{m}^*$ receiving $\textsc{Ch}^*$. $\mathcal{F}$ sends $\textsc{Ch}^*$ to $\mathcal{I}$, correctly simulating the oracle $\mathcal{O}_{\mathsf{View}-\mathcal{TID}}(\cdot)$ being $\textsc{Ch}^*$ the output of a random oracle of an input that has never been queried before. $\mathcal{I}$ generates a valid response $\textsc{Rsp}^*$ and $\mathcal{F}$ use it to generate its forgery $(\textsc{Cmt}^*||\textsc{Rsp}^*)$ to the message $\mathtt{m}$ which has never been queried in a sign query. Whenever the impersonator succeeds in its impersonation attempt, also $\mathcal{F}$ succeeds and creates a valid forgery.

$\square$

The proofs of Lemma 3 and Lemma 4 prove Theorem 2.

## D Sparkle signature scheme

Below we provide the description of the Schnorr threshold signature Sparkle using the same notation used in [18].

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

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

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

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

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

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

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

    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 $(\mathsf{state}_k, R_k)$

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

    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 $z_k$

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

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

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

    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. 8.** Sparkle Signature Scheme

In the $\mathcal{TID}$ of Figure 5 we avoided to explicitly write $\mathsf{state}$. Moreover:

- $\mathsf{TSign}_1$ is the same of $\mathsf{TP}^{\mathrm{Com}}_{\mathrm{CMT}}$ where the session id `ssid` is replaced by `m`.
- during $\mathsf{TSign}_2$ each party checks the received data and outputs its partial commitment $R_k$. This is the same as the first line of $\mathsf{TP}^{\mathrm{Rel}}_{\mathrm{CMT}}$, where the check are omitted for the sake of readability.
- in $\mathsf{TSign}_3$ each party checks the consistency of each $\mathrm{CMT}_i$, 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 $\mathsf{TP}^{\text{Rel}}_{\text{CMT}}$ as well as the first two line of $\mathsf{TP}_{\text{RSP}}$.

– In Combine each party combines all the partial signatures to obtain the final signature. These are the last two lines of $\mathsf{TP}_{\text{RSP}}$.

## E Forking Lemma

First introduced by Pointcheval and Stern [28], 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 [24]. More formally we have the following lemma, by M. Bellare and G. Neven in [5]:

**Lemma 5 (General Forking Lemma).** *Let $q \in \mathbb{Z}$ with $q \geq 1$, $H$ be a set with $|H| \geq 2$. Let $\mathsf{IG}$ 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 $(J, \sigma)$ with $J$ being an integer $0 \leq J \leq q$ and $\sigma$ 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*

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