NON-INTERACTIVE
SCHNORR THRESHOLD IDENTIFICATION

FOTEINOS MERGOUPIS-ANAGNOU

Abstract. Threshold zero-knowledge protocols have not been widely adopted, presumably due to the relevant network overhead, required infrastructure maintenance and limited interoperability chances. We propose an agile threshold identification scheme based on the Schnorr protocol, which is the native identification scheme in discrete-logarithm based settings. Given a Shamir (n, t)-shared secret x, the proposed protocol allows any $t^* \geq t$ (but no less) shareholders to collectively prove that their secret keys combine to x in the sense of interpolation without revealing any information beyond that. The protocol is non-interactive *and* non-reliant on the public shares. On the one side, the provers do not need to engage in any multi-party computation assisted by a trusted combiner; instead, they act independently and asynchronously, sending their proof packets to the verifier directly. On the other side, the verification operation contains no reference to the public shares, meaning that the received packets are verified against the group public key alone and that the verifier does not need to trust and register the individual members' identities. The protocol can be cheaply adopted in dynamic and unregulated environments, where no certrification processes or infrastructure support are available, and be seamlessly integrated with existing verifiers by means of pure software extension. No trusted setup is required beyond the minimum assumption that the key has been shared according to Shamir's secret sharing (or equivalent distributed key generation (DKG) protocol) and that the group public key has been advertised correctly; in particular, the protocol is intended to be secure against impersonation attacks in the plain public-key model. We provide evidence that this has good chances to hold true by giving a formal security proof in the random oracle model under the one-more discrete-logarithm hardness (OMDL) assumption.

1. Introduction

Distributing the prover of a zero-knowledge protocol is usually achieved my means of a trusted combiner who acts as group proxy. The shareholders engage in some distributed computation in order to produce a uniform commitment, which they subsequently use in order to generate ordinary proofs for their local shares. After aggregating the respective proof shares, the proxy combines them into an ordinary proof and forwards it to the verifier, who proceeds to verification like in the non-distributed case. That is, the combined proof resides in the space of proofs that could have been generated for the combined secret directly. The idea is similar to that of the Schnorr threshold signature scheme [12] and has been elaborated in the proof-of-knoweldge context by [5], who apply it to distribute protocols as general as proving the preimage of a homomorphism [7].

The integration of such protocols is seamless on the verifier's side because the latter does not need to trust and register the shareholders' public keys. However, this convenience comes at significant cost on the prover's side. First and foremost, the distributed protocol is highly interactive even though its non-distributed counterpart may consist of a single communication step. The multi-party computation for the uniform commitment, usually a distributed key generation (DKG) scheme, introduces significant network overhead upon every protocol round while also relying on subtle security assumptions regarding the involved parties. Note that delegating combination to the verifier's side makes no difference (except maybe for fewer trust assumptions) since the protocol would remain equally interactive. Techniques for reducing interactivity, e.g., pseudorandom secret sharing, do not scale beyond a limited number of involved provers and, most importantly, require preprocessing along with the relevant infrastructure maintenance. While the above may not be an issue for operations with long-term impact in monitored and regulated environments (e.g., distributed signatures stored in a public ledger), they become prohibiting if the provers are compelled to act unattendedly and asynchronously in the wild, e.g., for the purpose of multi-factor authentication or when performing some joint IoT or network operation in order to rapidly release a sensitive action. We are primarily interested for use cases of this kind.

Another relevant paradigm is that of threshold decryption, which employs decoupled zero-knowledge proofs in order to detect maliciously fabricated partial decryptors. Given a ciphertext, the shareholders attach a Chaum-Pedersen proof to their respective decryptor that bounds it securely to their respective share. After aggregating sufficiently many of these packets, the decrypting party reconstructs the combined decryptor only if the attached proofs verify against the respective public shares, i.e., decryption is not released unless all proofs are individually found to be valid. Since every proof includes a Schnorr proof for the respective shareholder's key, one could eliminate the encryption specific details in order to derive a pure identification protocol.

A protocol of this kind is non-interactive and requires no infrastructure or coordination on the provers' side, allowing the shareholders to act independently; however, it does require from the decrypting party (i.e., the verifier) to maintain the correct public shares in order to be able to individually verify the attached proofs. This evidently entails the need for a trusted multi-key setup that certifies both the ownership of the public shares and their compatibility (i.e., that they comprise a sharing of the group public key), otherwise the protocol remains susceptible to all sorts of rogue-key attacks. Note that rogue-key attacks may not be a major issue for threshold decryption because the attached proofs are verified against both the respective public shares and the given ciphertext; if the ciphertext has been generated with respect to the correct group public key, a shareholder's rogue key will most probably be detected. This is not so for the derived identification protocol, where the verification operation involves nothing beyond the registered public shares. In this case, security becomes entirely dependent on the initial multi-key setup, which has a generally intractable attack surface and can only be analysed per use case. Even if an encompassing setup could be designed and analysed, this does not guarantee its adoptability; any such setup is not seamless to integrate with and, if adopted, it comes at the cost of limited interoperability and significant infrastructure extensions and maintenance. It is completely useless in a flexible and unregulated identification setting, where no trust infrastructre can be assumed to be in place at all.

1.1. **Contribution.** The above discussion implies that non-reliance on the public shares and non-interactiveness on the provers' side appear to be diametrically opposed properties; eliminating infrastructure on one side requires its existence and maintenance on the other. We propose an infrastucture-free Schnorr-based threshold identification scheme which achieves non-interactiveness *and* non-reliance; as such, it is suitable for flexible identification settings and can be cheaply adopted in the following sense: (a) existing verifiers need only extend their software in order to support the distributed case, without adding new infrastructure for trusting and registering public shares (b) existing keys can be shared at the verifiers' complete negligence. No trusted setup is required beyond the minimum assumption that a key has been shared amongst a group of shareholders according to some Shamir-based scheme (no matter which) and that the combined public key has been advertised correctly; in particular, the protocol should be secure against impersonation attacks in the plain public-key model ([3]). We provide evidence that this has good chances to be true by demonstrating a formal security proof in the random oracle model under the one-more discrete-logarithm hardness (OMDL) assumption.

1.2. **Organization.** In Section 2, we get a closer look at distributed Schnorr identification and argue in detail why certain solutions fail to meet its pragmatic needs, specifically with respect to unattended, unregulated and decentralized settings. In Section 3, we introduce the proposed threshold identification scheme and make some high-level security considerations, including a key-recovery attack against its interactive variance. In Section 4, we specify the threat model and derive the relevant attack games. We do not formulate auxiliary notions for zero-knowledge protocols (e.g., soundness, honest-verifier zero-knowledge, witness indistinguishability etc.) in the threshold setting, focusing directly on security against impersonation attacks; however, we do derive a restricted notion of extractability (in the sense of special soundness) in order to specify a reasonable attack model under the OMDL hardness assumption. In Section 5 we present the formal security proof in the random oracle model, employing the above mentioned key-recovery attack. The formalism and main facts regarding Shamir's sharing are presented in Appendix A and a quick refresher on the Schnorr protocol can be found below.

1.3. **Schnorr protocol.** Fix a cyclic group $\mathbb{G} = \langle g \rangle$ of order q. A secret key is a scalar $x \in \mathbb{Z}_q$, with the group element $y = g^x$ being its public counterpart; equivalently, $x = \log y$, where the logarithm is taken with respect to the generator g of the underlying group structure. *Discrete-logarithm hardness* (DL) for the system under consideration is the assumption that computing the logarithm of a uniformly random element is infeasible for every existing computer. More accurately, given $y \leftarrow_\$ \mathbb{G}$, every probabilistic polynomial-time algorithm should correctly compute $x \leftarrow \log y$ with negligible chance of success, where negligibility is determined by the current level of disposable computational power. Industrial public-key cryptography relies increasingly more on the plausibility of this assumption over carefully chosen groups, a trend that will only be challenged with the advent of practical quantum computing.

The *Schnorr identification protocol* is a non-interactive zero-knowledge (NIZK) scheme, allowing the key holder of $y = g^x$ to prove knowledge of x without revealing any information about the latter. Its original interactive variance [10] is a sigma protocol. The prover samples $r \leftarrow_\$ \mathbb{Z}_q$, stores it for the current session and sends the commitment $u = g^r$ to the verifier. The verifier samples a challenge $c \leftarrow_\$ \mathbb{Z}_q$, caches it for the current session and sends it to the prover. Finally, the prover responds with $s = r + cx$ and the verifier accepts only if the relation

$$g^s = y^c u$$

is satisfied; indeed, $g^s = g^{r+cx} = (g^x)^c g^r = y^c u$. In the presence of a secure hash function $H : \mathbb{G} \to \mathbb{Z}_q$, the protocol can be made non-interactive (1-step) by means of the Fiat-Shamir transform. Specifically, the prover precomputes $c \leftarrow H(u)$ and sends (u, s) to the verifier, who retrieves c from u and proceeds to verification. Non-interactiveness is most often preferrable due to greater robustness, reduced throughput and non-reliance on real-time communication. The intuition behind is that a secure hash function should inject sufficiently high entropy across every bit of its output, so that assigning $c \leftarrow H(u)$ becomes indistinguishable from sampling c uniformly at random. The current mainstream formalizes this fact within the random oracle model (ROM).

The Schnorr protocol is an identification scheme because it verifies the holder of a key against its public counterpart in a sound and complete fashion. Key semantics are not essential; the protocol identifies any procedure that "sees" the secret logarithm of some given group element. It is namely a zero-knowledge (ZK) proof-of-knowledge primitive for the discrete logarithm and as such applicable in any context where this knowledge arises as crucial (e.g., in plain ElGamal encryption, where a proof of randomness may be attached in order to derive a non-malleable ciphertext); most notably, it is the building block for proving knowledge of witnesses to arbitrary linear relations, common instances of which are the Chaum-Pedersen protocol for Diffie-Hellman tuples and the Okamoto protocol for representations. From this perspective, security against impersonation attacks amounts to the infeasibility of fabricating a proof that verifies against a group element without knowing its logarithm; in the zero-knowledge setting, this is called witness extractability under discrete-logarithm hardness. The protocol becomes meaningful only in the font of DL hardness, which in turn guarantees that the protocol is secure.[1]

2. The problem of distributed Schnorr identification

Suppose that some secret x with public counterpart y is distributed as $x_1, \ldots, x_n$ among n shareholders by means of Shamir's (n, t)-sharing or equivalent distributed key generation (DKG) protocol. That is, it shouldn't matter if x is shared by a dealer or is implicitly defined in a decentralized fashion without being locally reconstructible; it only suffices that the output of the distribution scheme is indistinguishable from that of Shamir's secret sharing. We are interested for a protocol that allows any $t^* \geq t$ (but no less) parties to collectively prove that they belong to the group represented by y in a zero-knowledge fashion. More accurately, involved shareholders should be able to prove that their shares combine to x by means of Shamir's reconstruction formula (cf. Remark A.2) without revealing any information beyond that; evidently, this reduces to ordinary Schnorr identification for the edge case $t = 1$. Security against impersonation attacks should roughly mean that

[1]The most far-reaching result of this kind is that Schnorr identification is secure against active and concurrent impersonation attacks under "one-more" discrecte logarithm (OMDL) hardness; cf. Theorem 2, [4].

any efficient adversary controlling up to $t-1$ shareholders remains unable to fabricate a proof that verifies with non-negligible chances. We obviously ask for the protocol to be as less interactive as possible; in the ideal case, involved provers should not need to engage in any communication rounds, performing single and independent requests to the verifier.

There is a seemingly trivial solution to this problem involving the public keys $y_i = g^{x_i}$, $1 \leq i \leq n$, of the shareholders. Let $Q \subset \{1, \ldots, n\}$ be a coalition of shareholders that engage in an identification session with a verifier. The i-th party generates an ordinary Schnorr proof for its respective share, i.e.,

$$(u_i, s_i), \quad s_i \leftarrow r_i + c_i x_i, \quad c_i \leftarrow H(u_i), \quad u_i \leftarrow g^{r_i}, \quad r_i \leftarrow_{\$} \mathbb{Z}_q$$

and sends it to the verifier. After aggregating the proof packets (u_i, s_i), $i \in Q$, the verifier computes $c_i \leftarrow H(u_i)$, $i \in Q$, and accepts only if the relation

$$y = \prod_{i \in Q} y_i^{\lambda_i} \;\wedge\; g^{s_i} = y_i^{c_i} u_i, \, \forall i \in Q \tag{2.1}$$

is satisfied, where $\{\lambda_i\}_{i \in Q}$ are the Lagrange interpolation coefficients for Q. We refer to this protocol as the *trivial approach*. The verifier identifies the involved parties only if their public shares combine to the group public key and they individually prove knwoledge of their respective secret. The first condition is satisfied only if $|Q| \geq t$ (as desired) while the second is a method for hardening distributed protocols against malleability, i.e., requiring from the involved parties to present proofs for contextual statements that verify against their respective public keys. Note also that (2.1) is an overkill. If the public shares $y_1, \ldots, y_n$ must be advertised and trusted anyway (i.e., the verifier registers them as the fixed sharing of y), the identifying condition reduces to

$$|Q| \geq t \;\wedge\; g^{s_i} = y_i^{c_i} u_i, \, \forall i \in Q \, . \tag{2.2}$$

The protocol is secure if $y_1, \ldots, y_n$ is the correct (n, t)-sharing of y; however, in the absence of side input on the verifier's side, its security relies entirely on the registration process. Indeed, as illustrated by (2.2), the bulk of the work has been absorbed in the registration of the public shares; the verifier accepts only because the public shares have already been trusted and the security analysis of the protocol reduces trivially to the threat model for its initial setup. The problem is that a trusted multi-key setup is itself a highly non-trivial assumption, since the possibilities regarding the involved roles and operations compose a rather broad attack surface. For example, a proxy that is responsible for advertising the public shares may tamper with a certain subset of them, causing denial-of-service whenever the respective shareholders engage in a session. Or, worse, the proxy colludes with a malicious registry delivering the group public key and mounts the following rogue-key attack. The proxy retains $y_1, \ldots, y_{t-1}$, replaces y_t with some y_t^* for which it knows the discrete logarithm and instructs the registry to deliver

$$y^* = (y_t^*)^{\lambda_t} \prod_{i=1}^{t-1} y_i^{\lambda_i}$$

in place of y, where $\{\lambda_i\}_i$ are the Lagrange interpolation coefficients for $\{1, \ldots, t\}$. By reverse-engineering Shamir reconstruction, the proxy tunes y_i^*, $t < i \leq n$, so that $y_1, \ldots, y_{t-1}, y_t^*, \ldots, y_n^*$ becomes a (n, t)-sharing of y^* and broadcasts it to the verifiers; in the sequel, it can jump in and impersonate the t-th shareholder whenever the shareholders $1, \ldots, t-1$ engage in a session. While ad hoc threat models can obviously be formulated for specific use cases (e.g., for a restricted pool of shareholders that undergo a well specified certification process), this discussion implies that an ecompassing threat model is impossible to pin down. In the terminology of Bellare and Neven [3], the protocol is not secure in the *plain public-key* model.

One can alleviate the situation by working in the *knowledge of secret key* (KOSK) model, i.e., design a zero-knowledge based registration process, capable of detecting rogue-key attempts by means of purely cryptographic techniques. In this case, a security analysis becomes more tractable because a threat model for the initial setup can be formulated in terms of well defined malicious acts. However, an operation of this kind is not obvious to scale and its repetition can be cumbersome if the shareholders are not static. More importantly, KOSK narrows the applicability of the protocol severely, since any verifiable registration process is an infrastructure plugin that real-world vendors may be unable or unwilling to interoperate with.

These are standard arguments speaking in favour of secure multi-party protocols that are agnostic with respect to their setup internals. For threshold identification, this requirement is further hardened as *non-reliance on the public shares*, meaning that the verification operation should only involve the group public key. In fact, this is a sine qua non if the desired protocol is to be applicable in flexible decentralized settings.

2.1. **Distributed provers in decentralized networks.** We contended above that non-reliance on the public shares is a necessary condition for pragmatic security in the plain public-key model, where no concrete certification process for the public shares can be assumed to be available or known. Existence of such processes indicates a relatively limited number of shareholders that either belong to an organization or participate in a supposedly decentralized network, where group constitution is subject to regulation and monitoring. In such cases, the role of verifier is usually reserved for a bunch of authorized servers or nodes, responsible for managing some protected resource or performing specific acts in the context of a broader protocol. The trivial approach with an ad hoc setup, supported by the relevant infrastructure and attack model, can be a decent solution, at least insofar as the group of shareholders remains relatively static and general interoperability is not a hard requirement. We are interested for use cases beyond that.

We address distributed identification in a fairly unconditioned context, with as few assumptions as possible beyond a generic network with reliable packet delivery. The only assumption regarding Shamir-based group constitution is that the combined public key be correctly advertised. No infrastructure is assumed to be available after that phase, e.g., proxies, registries for individual member identities, storages for pre-processed quantities, aggregators interacting with involved provers, or even traceable communication channels between them. A group might be created on the fly and its public key be the only remnant of that ceremony; after dissolution of the setup channels, its members may have no way to coordinate or identify with each other, be unwilling to do so or be compelled to act independently.

Absence of infrastructure implies that the provers should be able to act without relying on real-time communication, meaning that the protocol has to be non-interactive on their side. This introduces a high degree of asynchronicity which imposes constraints on their interaction with the verifier. In particular, a verifier's response to the j-th shareholder should not be dependent on any request from the i-th shareholder for $i \neq j$. Equivalently, the proving session should decouple into independent interactions between the verifier and the involved provers; after aggregating the decoupled results of these interactions, the verifier proceeds to the final verification step. Ideally, the decoupled interactions should themselves be non-interactive, i.e., consist of a single communication step in which the involved parties send their respective proofs to the verifier.

On the verifier's side, absence of infrastructure implies non-reliance on the public shares. Consider the extreme case of a *truly* decentralized network, where qualified verifiers can be user devices or application servers alike in a dynamic fashion. That is, verification can occur on a multitude of devices that are not aware of the individual members' identities for every group they come accross in the network. It simply cannot be expected from a mobile or other IoT device to maintain the public shares for every group it might want to verify, let alone engage in a certification process in order to register them. Note also that fetching the public shares from remote registries is not an option; even if such registries exist, trusting them is a highly non-trivial assumption (comparable to KOSK) and depending on them would significantly affect latency and robustness.

Further efficiency considerations are here in order, since the decentralized verifiers do not necessarily possess the capabilities of a fine-tuned application server. It is well known that the wireless transmission of a single bit requires more power than a 64-bit (or 32-bit) instruction by orders of magnitude, which implies that throughput is the dominating factor with respect to overall energy consumption. From the individual node's perspective, lower throughput implies longer batterly life if it runs on a wireless device. Low throughput is even more important in order to avoid network congestion, since the proof packets have to be massively trasmitted through relay nodes in order to be verified at arbitrary locations. The problem remains to design a non-interactive threshold protocol with low throughput that is non-reliant on public shares, infrastructure-free on both sides, secure in the plain public-key model, and as such applicable in truly decentralized settings.

2.2. **Non-working solutions.** We discuss two plausible approaches to the problem of threshold Schnorr identification and show how they fail to meet the requirements of Section 2.1. Both are within the plain public-key model.

2.2.1. *Uniform commitment approach.* Recall that Schnorr signature is Schnorr identification with a message baked into the hash function. This suggests an approach similar to the threshold signature scheme introduced by [12]. Let $x_1, \ldots, x_n$ be the fixed (n, t)-sharing of x and y be its public counterpart. A coalition of shareholders $Q \subset \{1, \ldots, n\}$, $|Q| \geq t$, execute some DKG protocol to generate

$$r = \sum_{i \in Q} \lambda_i r_i,$$

where $\{\lambda_i\}_{i \in Q}$ are the interpolation coefficients for Q, holding the respective r_i's in private. r need not be locally reconstructible but involved parties should be able to verifiably infer the distributed commitment $u = g^r$ at the end of the process. Next, the involved parties generate Schnorr proofs for their keys using the respective commitment share and uniform challenge $H(u)$, i.e.,

$$(u_i, s_i), \quad s_i \leftarrow r_i + cx_i, \quad u_i \leftarrow g^{r_i}, \quad c \leftarrow H(u),$$

and send them to the verifier. After aggregating (u_i, s_i), $i \in Q$, the verifier computes

$$s \leftarrow \sum_{i \in Q} \lambda_i s_i, \quad c \leftarrow H(u), \quad u \leftarrow \prod_{i \in Q} u_i^{\lambda_i}$$

and accepts only if the relation $g^s = y^c u$ is satisfied. Indeed, since $|Q| \geq t$, twofold application of Shamir's reconstruction formula yields

$$g^s = \prod_{i \in Q} (g^{s_i})^{\lambda_i} = \Big(\prod_{i \in Q} (g^{x_i})^c \Big)^{\lambda_i} \prod_{i \in Q} (g^{r_i})^{\lambda_i} = \Big(\prod_{i \in Q} y_i^{\lambda_i} \Big)^c \prod_{i \in Q} u_i^{\lambda_i} = y^c u.$$

Verification requires $|Q| + 2$ exponentations and the inbound throughput grows linearly with respect to $|Q|$; for example, taking $\mathbb{G}$ to be the elliptic curve P256 and assuming that 128-bit challenges is sufficient to work with, the total throughput is $512\,|Q|$ bits per protocol round. If accomodated, we can offload the computation of s and u to a semi-trusted combiner, who acts as group proxy by forwarding (u, s) to the verifier; in this case, the verifier coincides with that of the non-distributed case and its performance is not dependent on the number of involved shareholders. This is usually the case for Schnorr threshold signatures.

This protocol is equivalent to ordinary Schnorr identification if we treat the involved provers as a single entity; i.e., (u, s) lies in the domain of proofs that could have been generated for x directly. This commutation is crucial for a threshold signature scheme, where the final object may need to remain available in the long run and consequently use minimum space, but not for an identification protocol, where no persistent data are generated. In fact, it countervenes the requirement of reduced interactiveness, as it is achieved through the distributed computation of u. This introduces significant network overhead upon every protocol round and presupposes that some infrastructure is available to the shareholders. Consequently, despite the efficiency of its verifier, the uniform commitment approach does not meet the requirements of Section 2.1 on the provers' side. The state-of-the-art for managing the network overhead of Schnorr threshold signatures is the FROST protocol introduced by [6] and its variants. This reduces the number of communication rounds to one or two by means of a semi-trusted aggregator, depending on whether a preprocessing stage takes place during the initial setup; however, it comes at the cost of more infrastructure (and more trust assumptions) which makes it even more prohibiting for a flexible identification setting.

2.2.2. *Completely decoupled approach.* This is a variance of the trivial approach, where the public shares need not be advertised. Instead, they are attached to the proofs. Given a (n, t)-sharing $x_1, \ldots, x_n$ of some secret x with public counterpart y, a coalition of shareholders $Q \subset \{1, \ldots, n\}$ generate

$$(y_i, u_i, s_i), \quad y_i \leftarrow g^{x_i}, \quad s_i \leftarrow r_i + c_i x_i, \quad c_i \leftarrow H(u_i), \quad u_i \leftarrow g^{r_i}, \quad r_i \leftarrow_{\$} \mathbb{Z}_q$$

respectively and send them to the verifier. After aggregating the packets (y_i, u_i, s_i), $i \in Q$, the verifier computes $c_i \leftarrow H(u_i)$ and accepts only if condition (2.1) is satisfied. Security reduces to the proof-of-knowledge property of the Schnorr protocol. Every party "should know" the logarithm of the attached group element because the respective proof verifies; since the attachments combine to $y = g^x$, the respective secret logarithms comprise a sharing of x and the provers should belong to the group represented by y.

The protocol is non-reliant on public shares and infrastructure-free on both sides. However, this is achieved at the cost of $3\,|Q|$ exponentiations per verification and at least 33% increased throughout as compared to the trivial approach. While this regression might be insignificant for a central identifying server with even several thousands of users, it can badly affect efficiency under the energy intensive and highly congested network conditions of true decentralization (cf. Section 2.1).

Moreover, the security argument is not as strong as it might seem. Recall that the Schnorr proof-of-knowledge property is not defined in terms of an attack game and it basically means extractability under DL hardness. Consequently, the security of the threshold protocol cannot reduce to it strictly; by saying that it "reduces to" it, what we actually mean is that some relevant extraction procedure should itself be reproducible in the context of a rewindable black-box threshold attack. This does not seem to be possible. Consider an adversary $\mathcal{A}$ who knows $\{x_i\}_{i \in J}$ for some $J \subset \{1, \ldots, n\}$ with $|J| = t - 1$ and wants to impersonate $Q = J \cup \{j\}$ with $j \notin J$ against an interactive verifier. The extractability argument regarding security means that any such $\mathcal{A}$ should necessarily spit x_j by rewinding in case of a successful impersonation attack. One possible attack pattern could be using the shares that are known to $\mathcal{A}$ directly. That is, for every $i \in J$, $\mathcal{A}$ sends

$$(y_i, u_i), \quad y_i \leftarrow g^{x_i}, \quad u_i \leftarrow g^{r_i}, \quad r_i \leftarrow_{\$} \mathbb{Z}_q$$

to the verifier, who samples $c_i \leftarrow_{\$} \mathbb{Z}_q$ for u_i and sends it to $\mathcal{A}$; in turn, $\mathcal{A}$ responds with $s_i \leftarrow r_i + c_i x_i$. Next, $\mathcal{A}$ fabricates u_j accoridng to some black-box strategy and sends (y_j, u_j), $y_j \equiv g^{x_j}$, to the verifier, who samples $c_j \leftarrow_{\$} \mathbb{Z}_q$ for u_j and sends it to the adversary. Finally, $\mathcal{A}$ fabricates s_j according to some black-box strategy and sends it to the verifier. Evidently, extraction of x_j is possible by rewinding $\mathcal{A}$ exactly before sending (y_j, u_j) to the verifier, provided that it succeeds in generating a valid transcript. This is ordinary extraction embedded into a decoupled threshold attack pattern. However, a black-box threshold attack is far from confined to this pattern. Since y admits many possible sharings, $\mathcal{A}$ could possibly fabricate attachments that do not coincide with the fixed public shares and still combine to the group public key; in this case, rewinding $\mathcal{A}$ at any location would fail to yield x_j. We would need the decoupled proofs to be further junct into a single operation in order for the extraction of x_j to be possible by rewinding.

3. One-round Schnorr threshold identification

We fix a cyclic group $\mathbb{G} = \langle g \rangle$ of order q where the DL problem is hard and a secure hash function $H : \mathbb{G} \to \mathbb{Z}_q$. Shamir's secret sharing is denoted by D (cf. Appendix A) and the values of its parameters are inferred always from context.

3.1. **The ORST protocol.** Given integers (n, t) such that $1 \leq t \leq n < q$, we define the following threshold identification scheme.

Protocol 3.1. ($\mathsf{ORST}_{\mathbb{G},n,t}$ – *One-Round Schnorr* (n,t)*-Threshold Identification*)

- *Setup.* Some uniformly random secret is distributed among a group of n shareholders by means of Shamir's (n,t)-sharing, i.e.,

$$(x_1, \dots, x_n;\, y) \leftarrow D(x),\;\; x \leftarrow_\$ \mathbb{Z}_q$$

  The threshold parameter and the public shares may remain private to the group, who needs only advertise the combined public key y.

- *Proving phase.* A coalition of shareholders $Q \subset \{1, \dots, n\}$ generate

$$(u_i, s_i),\quad s_i \leftarrow r_i + c_i x_i,\quad c_i \leftarrow H(u_i),\quad u_i \leftarrow g^{r_i},\quad r_i \leftarrow_\$ \mathbb{Z}_q$$

  and send it to the verifier holding r_i in private, $i \in Q$.

- *Verification.* After aggregating (u_i, s_i), $i \in Q$, the verifier accepts only if

$$\exp\Big(g, \sum_{i \in Q} \mu_i s_i\Big) = y^{\bar{c}} \prod_{i \in Q} u_i^{\mu_i}, \tag{3.2}$$

  where

$$\bar{c} \leftarrow \prod_{i \in Q} c_i,\quad \mu_i \leftarrow \lambda_i \prod_{j \in Q \setminus \{i\}} c_j,\quad c_i \leftarrow H(u_i) \tag{3.3}$$

  and $\{\lambda_i\}_{i \in Q}$ are the Lagrange interpolation coefficients for Q.

The sharing scheme of the setup phase may be replaced with any DKG whose output distribution is indistinguishable from that of Shamir's secret sharing. That is, it shouldn't matter if x has been shared by a dealer or it cannot be reconstructed at a single location; it only suffices that the correct public key $y = g^x$ be advertised at the end of the process. Note that DKG's usually assume a bound $0 \leq l < t$ on the number of corrupt parties in order for the sharing to be secure, but this does not affect the security analysis of ORST (cf. Section 4.1.1) because it belongs to the threat model of the specific use case.

During the proving phase, the involved shareholders generate ordinary Schnorr proofs for their respective shares and send them to the verifier asynchronously. No real-time communication is required between them. Order coordination or some session timeout may be enforced by the verifier for the needs of a specific use case, but this should not affect the threat model essentially because the local computations remain decoupled. The protocol is non-interactive in a strict sense ("one-round") and meets the requirements of Section 2.1 on the provers' side.

Verification efficiency is comparable to that of the uniform commitment approach (Section 2.2.1), if at the cost of more (non-exponential) numerical operations. It requires $|Q| + 2$ exponentiations while its throughput grows linearly with respect to the number of involved shareholders; for example, if $\mathbb{G}$ is the elliptic curve P256 and 128-bit challenges is sufficient to work with, the inbound throughput is $512\,|Q|$ bits per protocol round. Since the operation per se involves the group public key alone, no maintentance of the public shares is required and the protocol meets the requirements of Section 2.1 on the verifier's side. Existing verifiers need only extend their software in order to support the distributed case, without adding new infrastructure for trusting and registering any public shares.

Remark 3.4. (*Validity condition*) By equating exponents, (3.2) translates to

$$\sum_{i \in Q} \mu_i \, (s_i - r_i) = \bar{c} \cdot x \,, \tag{3.5}$$

where $r_i \equiv \log u_i$. Observe that this contains no reference to secret shares.

Remark 3.6. (*Correctness*) Let $y_i = g^{x_i}$, $1 \le i \le n$ be the public shares. Under this notation, the left-hand side of condition (3.2) transforms as

$$\prod_{i \in Q} (g^{s_i})^{\mu_i} = \prod_{i \in Q} (g^{x_i})^{c_i \mu_i} (g^{r_i})^{\mu_i} = \prod_{i \in Q} (g^{x_i})^{\lambda_i \bar{c}} \prod_{i \in Q} (g_i^{r_i})^{\mu_i} = \Big(\prod_{i \in Q} y_i^{\lambda_i} \Big)^{\bar{c}} \prod_{i \in Q} u_i^{\mu_i},$$

which yields the right-hand side if (and only if) $|Q| \ge t$. In other words, the proper protocol execution verifies only if at least t shareholders are involved.

Remark 3.7. (*Interactive variance*) We may regard H as approximating a random oracle insofar as predicting any of its output bits remains infeasible; in the limit, the provers submit their commitments to an oracle that is accessible to the verifier. Or, equivalently, they send them directly to the verifier, who lazily samples challenges by maintaining a lookup table Map per session. For every $i \in Q$, the i-th shareholder samples $r_i \leftarrow_\$ \mathbb{Z}_q$ and sends $u_i = g^{r_i}$ to the verifier. If $u_i \notin \mathrm{Dom}(\mathsf{Map})$, the verifier samples $c_i \leftarrow_\$ \mathbb{Z}_q$ and caches $\mathsf{Map}[u_i] \leftarrow c_i$; in either case, it sends $c_i \leftarrow \mathsf{Map}[u_i]$ to the shareholder, who responds with $s_i \leftarrow r_i + c_i x_i$. After aggregating s_i, $i \in Q$, the verifier checks (3.2) using the previously cached challenges $c_i = \mathsf{Map}[u_i]$, $i \in Q$.

We proceed to the formal definitions regarding ORST.

Definition 3.8. The *weights of* $\{u_i\}_{i \in Q} \subset \mathbb{G}$ are the scalars $\{\mu_i\}_{i \in Q}$ defined as

$$\mu_i = \lambda_i \prod_{j \in Q \setminus \{i\}} c_j \tag{3.9}$$

where $c_i \equiv H(u_i)$ and $\{\lambda_i\}_{i \in Q}$ are the interpolation coefficients for Q. Under the same notation, the *normalizer of* $\{u_i\}_{i \in Q}$ is the scalar

$$\bar{c} = \prod_{i \in Q} c_i \tag{3.10}$$

That is, the verifier juncts the decoupled proofs through the weights and the normalizer of the points $\{u_i\}_{i \in Q}$, which the individual provers respectively commit to. We have already made use of the following relation (cf. Remark 3.6).

Remark 3.11. (*Relation between weights and the normalizer*) Let $\{\mu_i\}_{i \in Q}$, $\bar{c}$ be the weights and the normalizer of a collection $\{u_i\}_{i \in Q}$ respectively. Then

$$c_i \mu_i = \lambda_i \bar{c}, \quad \forall i \in Q \tag{3.12}$$

Definition 3.13. By *proof* is meant a collection $\sigma = \{(u_i, s_i)\}_{i \in Q}$ of the form

$$(u_i, s_i) \in \mathbb{G} \times \mathbb{Z}_q, \forall i \in Q \ \wedge \ Q \subset \{1, \ldots, n\}.$$

Given σ as above, we refer to $\{u_i\}_{i\in Q}$ as its *commitments* and say that σ is *based on* them. We also refer to $\{s_i\}_{i\in Q}$ as the *responses*. The *weights* and *normalizer of* σ are the weights and normalizer of its commitments respectively. The *proof space* is the set of proofs and will be denoted by Σ. We also denote by

$$\Sigma\big[\{u_i\}_{i\in Q}\big]$$

the subspace of proofs that are based on commitments $\{u_i\}_{i\in Q}$. Evidently, proofs in this subspace share the same weights and normalizer.

Note that the above proof notion does not impose constraints on the relation between the commitments and the responses; spcecifically, it does not confine to properly generated transcripts, modelling the verifier's view in the wild and including anything that malicious provers can possibly fabricate. The proper execution of the protocol is modelled as follows.

Definition 3.14. The proof $\{(u_i, s_i)\}_{i\in Q} \in \Sigma$ is called *canonical with respect to* a sharing $(x_1, \ldots, x_n;\, y) \in D_x$ if

$$s_i = r_i + c_i x_i, \quad c_i \equiv H(u_i), \quad r_i \equiv \log u_i, \quad \forall i \in Q, \tag{3.15}$$

i.e., its components are ordinary Schnorr proof for the respective shares. Evidently, the subspace $\Sigma\big[\{u_i\}_{i\in Q}\big]$ contains a unique canonical representative with respect to every fixed sharing.

Definition 3.16. The *one-round Schnorr* (n, t)*-threshold verifier* is the deterministic algorithm $\mathcal{V} : \mathbb{G} \times \Sigma \to \{\mathsf{true}, \mathsf{false}\}$ invoked as

$$\big(y, \{(u_i, s_i)\}_{i\in Q}\big) \quad \mapsto \quad \exp\Big(g, \sum_{i\in Q} \mu_i s_i\Big) = y^{\bar{c}} \prod_{i\in Q} u_i^{\mu_i},$$

where $\{\mu_i\}_{i\in Q}$, $\bar{c}$ are the weights and the normalizer of $\{(u_i, s_i)\}_{i\in Q}$ respectively.

Definition 3.17. A proof $\sigma \in \Sigma$ is called *valid against* $y \in \mathbb{G}$ if $\mathcal{V}(y, \sigma) = \mathsf{true}$.

ORST correctness (cf. Remark 3.6) means that a proof canonical with respect to $(x_1, \ldots, x_n;\, y)$ is valid against y if and only if $|Q| \geq t$. The subspace of proofs that are based on $\{u_i\}_{i\in Q}$ and valid against y is denoted by

$$\Sigma_y\big[\{u_i\}_{i\in Q}\big]$$

If $|Q| \geq t$, then this space contains a unique canonical representative with respect to every fixed sharing of $x = \log y$.

Remark 3.18. In the non-distributed case $t = 1$, the subspace $\Sigma_y\big[\{u_i\}_{i\in Q}\big]$ is a singleton due to the "unique responses" property of the ordinary Schnorr protocol. Specifically, for every commitment u there exists a unique response s such that the proof (u, s) is valid against $y = g^x$, namely $s = r + H(u)x$ with $r \equiv \log u$. In general, however, this space inflates dramatically at a rate dependent on $|Q| \geq t$ (cf. Remark 3.33). This implies an abundance of valid proofs that are non-canonical, a fact that every relevant security notion should be able to capture.

Proposition 3.19. (*Validity conditions*) *Let $x \in \mathbb{Z}_q$, $y = g^x$ and $\sigma = \{(u_i, s_i)\}_{i \in Q}$. Then, with overwhelming probability, σ is valid against y if and only if*

$$\frac{1}{\bar{c}} \sum_{i \in Q} \mu_i (s_i - r_i) = x, \quad r_i \equiv \log u_i \,, \tag{3.20}$$

where $\{\mu_i\}_{i \in Q}$, $\bar{c}$ are its weights and normalizer respectively or, equivalently,

$$x = \sum_{i \in Q} \lambda_i \, \frac{s_i - r_i}{c_i}, \tag{3.21}$$

where $c_i \equiv H(u_i)$ and $\{\lambda_i\}_{i \in Q}$ are the interpolation coefficients for Q.

Proof. Since H is a secure hash function, we have $\bar{c} \neq 0$ with overwhelming probability in the bitlength of the group order. In this case, the verification condition (3.5) becomes (3.20). This further reformulates to (3.21) by means of (3.12). □

This statement holds true irrespective of size (including the case $|Q| < t$), establishing a relation between x and any proof that happens to be valid against y without reference to sharings. Both (3.20) and (3.21) generalize the relation of a secret to a valid non-distributed proof; it should be stressed however that (3.21) does not imply $x_i = (r_i - s_i)/c_i$ and can hold true without σ being canonical. Indeed (cf. Remark 3.26), valid proofs do not confine to those properly generated in ORST and a relevant security notion should ensure that such proofs are infeasible to compute without knowledge of at least t shares (cf. Section 4). The exact meaning of (3.21) is clarified in Proposition 3.25. Evidently, a proof should not verify against different public keys. In fact, this key almost always exists and it is unique.

Proposition 3.22. (*Almost every proof is valid against some unique public key*) *Given commitments $\{u_i\}_{i \in Q} \subset \mathbb{G}$, then, with overwhelming probability, every $\sigma \in \Sigma$ based on them is valid against some unique $y \in \mathbb{G}$.*

Proof. Let $\{\mu_i\}_{i \in Q}$, $\bar{c}$ be the weights and the normalizer of $\{u_i\}_{i \in Q}$, which are common to all proofs based on them. Since H is a secure hash function, $\bar{c} \neq 0$ with overwhelming probability in the bitlength of the group order. Consequently, given any proof of the form $\sigma = \{(u_i, s_i)\}_{i \in Q}$, we can define

$$x = \frac{1}{\bar{c}} \sum_{i \in Q} \mu_i \, (s_i - r_i) \,,$$

where $r_i \equiv \log u_i$. By Proposition 3.19, σ is valid against $y = g^x$ solely. □

Corollary 3.23. *Given $\{u_i\}_{i \in Q} \subset \mathbb{G}$, then, with overwhelming probability,*

$$\Sigma\big[\{u_i\}_{i \in Q}\big] = \bigcup_{y \in \mathbb{G}} \Sigma_y \big[\{u_i\}_{i \in Q}\big] \tag{3.24}$$

where the right-hand side is a partition, i.e.,

$$\Sigma_y \big[\{u_i\}_{i \in Q}\big] \cap \Sigma_{y^*} \big[\{u_i\}_{i \in Q}\big] = \varnothing \;\; \textit{for} \;\; y \neq y^*.$$

Proof. Direct consequence of Proposition 3.22. □

Proposition 3.25. (*Almost every proof of size $|Q| = t$ is canonical with respect to some unique sharing*) *Given commitments $\{u_i\}_{i\in Q} \subset \mathbb{G}$ with $|Q| = t$, then, with overwhelming probability, every $\sigma \in \Sigma$ based on them is canonical with respect to some unique sharing $(x_1, \ldots, x_n; y) \in D_x$ of some unique $x \in \mathbb{Z}_q$.*

Proof. Let $\{\mu_i\}_{i\in Q}$ be the weights of $\{u_i\}_{i\in Q}$ and denote $c_i \equiv H(u_i)$. Note that these parameters are common to all proofs based on the given commitments. Since H is a secure hash function, $c_i \neq 0$, $\forall i \in Q$, with overwhelming probability in the bitlength of the group order. Consequently, given $\sigma = \{(u_i, s_i)\}_{i\in Q}$, we can define

$$x_i = \frac{s_i - r_i}{c_i}, \quad i \in Q$$

where $r_i \equiv \log u_i$. By virtue of interpolation, since $|Q| = t$, the scalars $\{x_i\}_{i\in Q}$ uniquely determine a (n, t)-sharing $(x_1, \ldots, x_n; y)$ of some $x \in \mathbb{Z}_q$, namely

$$x = \sum_{i\in Q} \lambda_i x_i \,.$$

Since $s_i = r_i + c_i\, x_i, \forall i \in Q$, the proof is canonical with respect to this sharing and thus valid against $y = g^x$. The claim follows because a proof can be valid against a unique public key (cf. Prop. 3.22). ☐

Remark 3.26. (*Validity and degrees of freedom*) Fix commitments $\{u_i\}_{i\in Q} \subset \mathbb{G}$. Choose $J \subset Q$ with $|J| = |Q| - 1$, pick arbitrary $\{s_i\}_{i\in J} \subset \mathbb{Z}_q$ and set

$$s_j \;=\; r_j + \frac{1}{\mu_j}\Big(\bar{c}\cdot x \;-\; \sum_{i\in J} \mu_i\,(s_i - r_i)\Big)$$

where $\{\, j\,\} = Q \setminus J$ and $r_i \equiv \log u_i$. By Proposition 3.19 (cf. (3.20)), this process exhausts the proofs based on $\{u_i\}_{i\in Q}$ and valid against $y \equiv g^x$ bijectively. That is, the subspace $\Sigma_y\big[\{u_i\}_{i\in Q}\big]$ has $|Q| - 1$ degrees of freedom in the sense that every included proof is uniquely determined by the choice of $\{s_i\}_{i\in J}$ and $J \subset Q$ (for $|Q| = 1$ this is the "unique responses" property of ordinary Schnorr identification). Observe that this process has no obvious utility for an impersonating adversary, since the latter would still need to eliminate x in order to compute s_j.

Proposition 3.27. (*Validity against sharing*) *Let $x \in \mathbb{Z}_q$ and $\sigma = \{(u_i, s_i)\}_{i\in Q}$. Given $(x_1, \ldots, x_n; y) \in D_x$, the proof σ is valid against y if and only if*

$$\sum_{i\in Q} \mu_i\,(s_i - r_i - c_i\, x_i) = \bar{c}\cdot \Big(x - \sum_{i\in Q} \lambda_i x_i\Big), \tag{3.28}$$

Proof. Follows from the verification condition (3.5) by substracting

$$\sum_{i\in Q} \mu_i c_i x_i$$

on both sides and applying (3.12) on the right. ☐

Remark 3.29. (*Valid proofs with $|Q| < t$*) We know that if a proof of size $< t$ is valid, then it cannot be canonical (cf. Remark 3.6). Such proofs exist in abundance (cf.

Remark 3.26) and Prop. 3.27 indicates a way to generate them programmatically. Fix $\{u_i\}_{i\in Q} \subset \mathbb{G}$ and set $s_i = r_i + c_i x_i$, where $r_i \equiv \log u_i$, $c_i \equiv H(u_i)$. Pick $S \subset Q$ with $S \neq \varnothing$ and define $s_i^* = s_i$ if $i \notin S$, otherwise

$$s_i^* = s_i + \delta_i, \quad \delta_i \equiv \frac{1}{\mu_i} \cdot \frac{\bar{c}}{|S|} \cdot \Big(x - \sum_{j\in Q} \lambda_j x_j \Big),$$

where $\bar{c}$ is the normalizer of $\{u_i\}_{i\in Q}$. Consider the proof $\sigma = \{(u_i, s_i)\}_Q$. If $|Q| \geq t$, then $\delta_i = 0$, $\forall i \in S$, so that σ is canonical and thus automatically valid. If $|Q| < t$, then σ still remains valid because it satisfies (3.28) by design. Note however that this construction has no obvious utility for a malicious coalition $Q \subset \{1, \dots, n\}$ with $|Q| < t$ since the colluders (or, equivalently, an impersonating adversary who knows $\{x_i\}_{i\in Q}$) would still need to know x in order to adjust the δ_i's appropriately.

Corollary 3.30. (*Validity against sharing, $|Q| \geq t$*) *Let $x \in \mathbb{Z}_q$, $\sigma = \{(u_i, s_i)\}_{i\in Q}$ with $|Q| = t$. Given $(x_1, \dots, x_n; y) \in D_x$, σ is valid against y if and only if*

$$\sum_{i\in Q} \mu_i s_i = \sum_{i\in Q} \mu_i (r_i + c_i x_i), \tag{3.31}$$

where $r_i \equiv \log u_i$, $c_i \equiv H(u_i)$ and $\{\mu_i\}_{i\in Q}$ are the weights of σ.

Proof. The right-hand side of (3.28) vanishes exactly if $|Q| \geq t$. ☐

Remark 3.32. ($t-1$ *degrees of freedom*) Fix $\{u_i\}_{i\in Q}$ with $|Q| = t$. Choose $J \subset Q$ with $|J| = t-1$, pick arbitrary $\{\delta_i\}_{i\in J} \subset \mathbb{Z}_q$ and set

$$s_j \;=\; r_j + c_j x_j - \frac{1}{\mu_j} \sum_{i\in J} \mu_i (s_i - r_i - c_i x_i),$$

where $\{j\} = Q \setminus J$ and $r_i \equiv \log u_i$, $c_i \equiv H(u_i)$. By Corollary 3.30, this process exhausts the proofs that are based on $\{u_i\}_{i\in Q}$ and valid against y bijectively. This is a special case of Remark 3.26 with reference to a fixed sharing. Observe that it has no obvious utility for an adversary who knows $\{x_i\}_{i\in J}$ but not x_j.

Remark 3.33. (*Valid proofs with $|Q| \geq t$*) This can be seen from a more general perspective. Consider a canonical proof $\{(u_i, s_i)\}_{i\in Q}$ with $|Q| \geq t$ and let $\{\mu_i\}_{i\in Q}$ be its weights. Pick $j \in Q$ and arbitrary scalars δ_i, $i \in Q \setminus \{j\}$, such that at least one of them is non-zero. If we define

$$\delta_j = -\frac{1}{\mu_j} \sum_{i\neq j} \mu_i \delta_i,$$

then the proof $\{(u_i,\, s_i + \delta_i)\}_{i\in Q}$ satisfies condition (3.31) by design. Evidently, this process exhausts the valid proofs that are based on $\{u_i\}_{i\in Q}$ (cf. Corollary 3.30).

3.2. **Multi-round executions.** While ORST has been designed so that the shareholders can act independently and asynchronously, any $|Q| \geq t$ provers have the option to collaboratively generate a valid non-canonical proof without revealing their secret shares to each other (as indicated by Remark 3.33), provided that traceable communication channels exist between them and the network overhead due to their interaction is not prohibiting. One possible reason is that the provers may want to blind their individual identities for privacy and anonymity.

Observe that a man-in-the-middle who intercepts a canonical proof $\{(u_i, s_i)\}_{i \in Q}$ can trivially infer the involved provers' public shares in the form

$$y_i = \sqrt[c_i]{\frac{u_i}{g^{s_i}}} \quad c_i \equiv H(u_i) \tag{3.34}$$

However, instead of sending the packets (u_i, s_i), $i \in Q$, to the verifier, the provers can first engage in the multi-party computation of scalars $\{\delta_i\}_{i \in Q}$ such that

$$\sum_{i \in Q} \mu_i \delta_i = 0 \tag{3.35}$$

and send the respective packets (u_i, s_i^*), $i \in Q$, where $s_i^* = s_i + \delta_i$. Indeed, if $|Q| \geq t$ then the proof $\{(u_i, s_i^*)\}_{i \in Q}$ remains valid because

$$\sum_{i \in Q} \mu_i s_i^* = \sum_{i \in Q} \mu_i s_i + \sum_{i \in Q} \mu_i \delta_i = \sum_{i \in Q} \mu_i s_i$$

(cf. Corollary 3.30). In this case, if $\delta_i \neq 0$, then

$$y_i \neq \theta_i, \quad \theta_i \equiv \sqrt[c_i]{\frac{u_i}{g^{s_i^*}}}$$

and anonymity follows because y_i cannot be feasibly inferred from θ_i, provided that the δ_i's remain private to the group of involved provers. We refer to $\{\delta_i\}_{i \in Q}$ as the *blinding factors*.

Note that the provers can follow almost arbitrary multi-round strategies in order to compute blinding factors (cf. Remark 3.33). If they want to keep them in private and do not trust each other, they can operate as follows. They first execute a DKG protocol in order to generate a verifiable (n, t)-sharing $(\eta_1, \dots, \eta_n;\, 1)$ of the zero scalar, so that η_i is known only to the i-th shareholder. By the homomorphic property of Shamir's secret sharing, $(x_1 + \eta_1, \dots, x_n + \eta_n;\, y)$ remains a (n, t)-sharing of the group secret x. Consequently, the provers can proceed to the proving phase of ORST with secret shares $x_i + \eta_i$, $i \in Q$ and generate a valid proof that is non-canonical with respect to $(x_1, \dots, x_n;\, y)$. Although not explicitly computed, the blinding factors are $\delta_i = c_i \eta_i$. By Proposition 3.22, any multi-round execution is essentially equivalent to an execution of this kind.

3.3. **Security considerations.** Since ORST decouples into concurrent executions of the ordinary Schnorr protocol, any security precaution for the latter applies also in the threshold case. For example, security against side-channel attacks cannot be attained if the random number generator leaks linear relations between the discrete logarithms of distinct commitments. We here focus on high-level attacks that arise particularly in the distributed setting.

3.3.1. *Generalized replay attack.* ORST is as much susceptible to replay attacks as the ordinary Schnorr protocol. That is, a man-in-the-middle can passively capture and replay the packets sent by a coalition of shareholders in order to impersonate them anytime. Things seem worse because the attacker can transform the intercepted proof almost arbitrarily, provided that the commitments are kept fixed. Let $\{(u_i, s_i)\}_{i \in Q}$ be a valid proof with $|Q| \geq t$ and weights $\{\mu_i\}_{i \in Q}$. For fixed $j \in Q$, pick $\{\delta_i\}_{i \in J} \subset \mathbb{Z}_q$ for $J = Q \setminus \{j\}$ and define

$$\delta_j = -\frac{1}{\mu_j} \sum_{i \in J} \mu_i \delta_i \, .$$

Then $\{(u_i, s_i^*)\}_{i \in Q}$ with $s_i^* = s_i + \delta_i$ remains valid bacause

$$\sum_{i \in Q} \mu_i s_i^* = \sum_{i \in Q} \mu_i s_i + \Big(\mu_j \delta_j + \sum_{i \in J} \mu_i \delta_i \Big) = \sum_{i \in Q} \mu_i s_i$$

(cf. Cor. 3.30). This essentially exploits the structure described in Remark 3.33. That is, in case of interception, the attacker has a least $t-1$ degrees of freedom in fabricating a valid proof based on the same commitments. This evidently generalizes the replay attack of the non-distributed case $t = 1$, where the intercepted proof can only be replayed as is. It should be clear that generalized replay attacks are mitigated by the same method as in the non-distributed case, i.e., by baking a nonce into the hash function, capable of maintaining state between the verifier and the involved provers.

3.3.2. *Non-accountability.* Since the verifier does not generally maintain any trusted shares, ORST is inherently non-accountable. In particular, if an acclaimed shareholder sends a maliciously fabricated proof packet (u_j, s_j) such that (3.2) resolves to false, the index j is generally untraceable for, say, security investigation purposes. If a use case presupposes that the public shares $y_i = g^{x_i}$, $1 \leq i \leq n$, are trustedly advertised along with y and registered on the verifier's side, the latter can accurately blame cheating j's by checking which of the conditions $y_i = g^{s_i} u_i$, $i \in Q$, are not satisfied.

Accurate blaming is sometimes possible in the general case. Suppose that the verifier has already identified some coalition of shareholders $Q \subset \{1, \ldots, n\}$ and an acclaimed shareholder sends a delayed proof packet (u_j, s_j), $j \notin Q$, such that (3.2) resolves to false for $Q \leftarrow Q \cup \{j\}$; the verifier can then deny access to j separately without this affecting the session for the rest shareholders. More generally, suppose that the verifier has aggregated proof packets for an index set $Q^* \subset \{1, \ldots, n\}$ and follows an incremental verification strategy. That is, instead of verifying all of them at once, the verifier tries combinations of increasing order until it hits a verifying collection of size t. If feasible, the verifier can then exhaust all combinations $Q \subset Q^*$ of size $|Q| = t$ in order to possibly detect cheating indexes, while maintaining a list with the honest ones.

3.3.3. *Key-recovery attack – interactive aspect.* This affects only the interactive version of the protocol (cf. Remark 3.7) under rather special circumstances, but sheds light on its inherent security structure. Similar to the non-distributed case, we will employ this attack in order to derive a formal security proof by means of fork and extraction (cf. Section 5.5). Evidently, if a shareholder uses the same commitment in two canonical executions of the interactive ORST (due to, say, broken random generator), then its secret share leaks trivially like in the non-distributed case. This is ordinary key-recovery embedded into the canonical threshold setting. We will show that key-recovery is possible even for non-canonical multi-round executions (cf. Section 3.2) given a malicious verifier who controls $t-1$ shareholders.

Let $Q = J \cup \{j\}$ be a coalition of shareholders such that $|J| = t-1$ with $j \notin J$ and the cluster J is controlled by an adversary $\mathcal{A}^*$. We further assume that $\mathcal{A}^*$ controls the identifying server. The shareholders engage in a multi-round session and generate a transcript $\{(u_i, c_i, s_i)\}_{i \in Q}$ that verifies, so that

$$\sum_{i \in Q} \mu_i (s_i - r_i - c_i x_i) = 0$$

holds true, where $r_i \equiv \log u_i$ and $\{\mu_i\}_{i \in Q}$ are the weights induced by the challenges $\{c_i\}_{i \in Q}$. (cf. Corollary 3.31). They next engage in a second session, where the j-th shareholder reuses the commitment u_j. After receiving u_j on the server's side, $\mathcal{A}^*$ responds with a challenge $c_j^* \neq c_j$ and tunes the rest shareholders to reuse the commitments u_i, $i \in J$ from the previous session. To each of them, the server responds with the same challenge $c_i^* = c_i$ from the previous session. Subsequently, the provers compute honestly their respective responses s_i, $i \in Q$, and send them to the server; since the transcript $\{(u_i, c_i^*, s_i^*)\}_{i \in Q}$ verifies, we again have

$$\sum_{i \in Q} \mu_i^* (s_i - r_i - c_i^* x_i) = 0\,,$$

where $\{\mu_i^*\}_{i \in Q}$ are the weights induced by the challenges $\{c_i^*\}_{i \in Q}$. Note that, since $c_i = c_i^*$ for all $i \neq j$, by definition of weights (cf. Definition 3.8) we get $\mu_j = \mu_j^*$. Substracting terms in the above relations and applying this remark, we get

$$\mu_j (s_j - s_j^* - (c_j - c_j^*) x_j) \; + \sum_{i \in J} \big(\mu_i (s_i - r_i - c_i x_i) - \mu_i^* (s_i^* - r_i - c_i x_i) \big) = 0$$

This eliminates r_j, which is the only unknown to the adversary. Since $c_j - c_j^* \neq 0$, $\mathcal{A}^*$ can use this relation to retrieve x_j in the form

$$x_j \; = \; \frac{s_j - s_j^*}{c_j - c_j^*} \; + \; \frac{1}{\mu_j} \sum_{i \in J} \Big(\mu_i (s_i - r_i - c_i x_i) - \mu_i^* (s_i^* - r_i - c_i x_i) \Big) \tag{3.36}$$

Note that since $|Q| = t$, the adversary can use the shares $\{x_i\}_{i \in Q}$ to also recover the combined secret key by means of Shamir's reconstruction formula.

4. Security notions

We specify the threat model for the ORST protocol. Formulations are given in terms of attack games within the asymptotic paradigm and negligibility is meant with respect to the bitlength of the underlying group order. In Section 4.1, we outline security against impersonation attacks in the abstract and correlate it with a pragmatic security requirement, namely a distributed knowledge property. In Section 4.1.2, we argue that this property cannot be derived under discrete-logarithm (DL) hardness without further assumptions, implying that security against impersontation attacks is most probably irreducible to the latter; however, we observe that the stronger one-more discrete-logarithm (OMDL) assumption is sufficient for deriving the knowledge property or, equivalently, yields the missing assumptions for a formal security proof based on fork and extraction. In Section 4.1.3 we examine some direct consequences of OMDL hardness and narrow our attack model so that the sufficient conditions of extractability are satisfied. The working model for black-box impersonation attacks is presented in Section 4.2 and the OMDL hardness assumption (cf. [4], [2]) is summarized in Section 4.3.

4.1. **Threat model.** The current standard for identification protocols is security against active impersonation attacks. This refers to adversaries who not only passively intercept identification sessions before mounting their actual attack, but also engage in them actively by impersonating a legitimate verifier (e.g., cloning a website in order to collect information about the user's credentials); after several such interactions, the adversary switches roles and attempts to identify itself against some verifier as a legitimate user. We will not pursue this path, focusing on adversaries who only intercept identification sessions and potentially exert some control over the network. Once the relation between threshold and ordinary Schnorr indentification has been clarified from the provability viewpoint, security against active adversaries should be straightforward to establish using techniques similar to those of [4] due to the decoupled structure of the threshold protocol.

4.1.1. *General threat model.* In most general terms, the ORST threat model consists of an adversary $\mathcal{A}$ who compromises the keys of up to $t-1$ shareholders and attempts to impersonate some coalition $Q \subset \{1, \ldots, n\}$, i.e., fabricate a proof $\{(u_i, s_i)\}_{i \in Q}$ that verifies. It does not make sense to consider adaptive adversaries because the proving phase consists of decoupled packets sent to the verifier; since no ephemeral quantities are exchanged between the shareholders, $\mathcal{A}$ has no chance to collect potentially useful information and decide which of them to corrupt adaptively. We can thus confine ourselves to the static model, assuming that the attacker decides from the outset the set of corrupted parties. Adaptive corruption during the execution of an equivalent DKG protocol in place of Shamir's sharing during the initial setup belongs to the threat model of the specific use case.

One could further assume that $\mathcal{A}$ knows the public keys of the individual shareholders; however, these are not used anyhow, although trivially inferred from protocol transcripts, cf. (3.34). Letting $\mathcal{A}$ know them from the outset corresponds to a scenario where the public shares become advertised within the group for at least some initial timespan (e.g., for mutual recognition and verification purposes) and $\mathcal{A}$ compromises the device of a shareholder who has not yet erased them. Still, the security proof of Section 5.6 remains intact under this stronger assumption (cf. Remark 5.19), indicating that an adversary who controls devices in the erasure-free model has no greater advantage than an adversary who only steals keys.

Let $(x_1, \ldots, x_n;\, y)$ be the given (n, t)-sharing of some secret x and $\mathcal{A}$ be an adversary who knows the secret shares $\{x_i\}_{i \in J}$ with $|J| = t-1$. Let $Q \subset \{1, \ldots, n\}$ be the sharholders that fall victim to impersonation under a purportedly valid proof $\{(u_i, s_i)\}_{i \in Q}$ fabricated by $\mathcal{A}$. Without loss of generality, $Q \subset J$ or $J \subset Q$. Indeed, since the black-box adversary knows nothing special about the uncorrupted shareholders, the choice of Q is irrelevant (except for its size) in the extent to which Q intersects with $\{1, \ldots, n\} \setminus J$; the adversary can change Q without this affecting its strategy, provided that $|Q|$ is preserved and its original intersection with J remains included. We conclude with the following general attack model.

- *Attack statement and corruption*:
  - $\mathcal{A}$ chooses $J \subset \{1, \dots, n\}$ with $|J| = t - 1$.
  - $\mathcal{A}$ chooses $Q \subset \{1, \dots, n\}$ with $Q \subset J$ or $J \subset Q$ and sends it along with J to its challenger $\mathcal{C}$.
  - $\mathcal{C}$ runs $(x_1, \dots, x_n; y) \leftarrow D(x)$, $x \leftarrow_{\$} \mathbb{Z}_q$ and sends $y, \{x_i\}_{i \in J}$ to $\mathcal{A}$.
- *Impersonation*: $\mathcal{A}$ outputs a proof of the form $\sigma = \{(u_i, s_i)\}_{i \in Q}$.

$\mathcal{A}$ wins if σ is valid against y.

Note that successful impersonation is equivalent to condition (3.20) for σ.

ORST security against impersonation attacks should mean that every polynomial-time adversary wins the above game with negligible advantage. The pragmatic requirement behind is that, if $\mathcal{A}$ outputs a proof $\{(u_i, s_i)\}_{i \in Q}$ that verifies, then it should necessarily know x_j for some $j \notin J$ (and consequently x); or, in other words, impersonation should be infeasible without knowledge of at least t secret shares. We will refer to this design requirement as the *knowledge property*.

This is instance of a more general pattern. Security notions for Schnorr-based protocols are naturally intermediated by some contextual knowledge property like above, which is the pragmatic security requirement behind. Consequently, a security reduction should (most probably) proceed by recovery of some contextual key in case of successful impersonation; by *knowing* this key, the adversary solves the DL or related hard problem, implying that it cannot have succeded (with non-negligible chances) in its impersonation attack. Key-recovery proceeds as follows. By rewinding the successful adversary at some special point and letting it repeat its attempt, the rewinder *extracts* the key from the partially differing transcripts of the two successful attacks. Security reduces to the hardness assumption only because extraction yields the solution to the underlying hard problem under any circumstances that allow for rewinding; if extraction is not possible under certain circumstances, security is (most probably) unprovable without further assumptions.

Regarding ORST, the relevant extraction formula for x_j is evidently equation (3.36) from the key-recovery attack of Section 3.3.3. We will see (cf. Section 4.1.2) that this formula is insufficient for retrieving x_j under plausible circumstances, indicating that ORST security is (most probably) irreducible to DL hardness alone. We need extra assumptions in order for security to become provable or, what is equivalent, reduce it to an easier problem than DL.

4.1.2. *Extractability and* $(t-1)$-OMDL *hardness.* Consider the following situation with $Q = J \cup \{j\}$ and $|J| = t - 1$. The shareholders $i \in Q$ engage in an multi-round session (cf. Section 3.2) with an interactive verifier (cf. Remark 3.7), where an adversary $\mathcal{A}$ acts as the acclaimed j-th shareholder. We further assume that $\mathcal{A}$ knows the shares $\{x_i\}_{i \in J}$ and can intercept the communication of the other shareholders. In particular, $\mathcal{A}$ captures u_i, c_i, $i \in J$. Next, $\mathcal{A}$ fabricates a commitment u_j according to some-black-box strategy and sends it to the verifier, who responds with c_j. Subsequently, the shareholders perform some multi-party computation to generate responses s_i, $i \in Q$. Specifically, $\mathcal{A}$ fabricates a response s_j according to some black-box strategy and sends it to the verifier. If the protocol is secure, then the transcript u_i, c_i, s_i, $i \in Q$, should verify with at most negligible chances, even though $\mathcal{A}$ knows $x_i, u_i, c_i,\ i \in J$, while fabricating u_j and s_j. Pragmatically, if the transcript verifies, then $\mathcal{A}$ should necessarily know x_j.

Extraction proceeds as follows. Let $\mathcal{A}^*$ be an algorithm with rewindable black-box access to $\mathcal{A}$. If $\mathcal{A}$ fabricates $u_j,\ s_j$ such that the transcript $u_i, c_i, s_i,\ i \in Q$, verifies with probability ε, then $\mathcal{A}^*$ rewinds $\mathcal{A}$ exactly before sending u_j and lets it repeat its impersonation attempt. By the forking lemma (cf. [3], or the reset lemma [4]), $\mathcal{A}$ succeeds again with probability $\varepsilon^* \approx \varepsilon^2$. Let $u_i^*, c_i^*, s_i^*,\ i \in Q$, be the trancript of the second impesonation attack; since rewinding occurs at the last commitment, we have

$$(\,u_i^* = u_i \ \wedge\ c_i^* = c_i\,)\,\forall i \in J \ \wedge\ u_j^* = u_j \ \wedge\ c_j^* \neq c_j\,, \tag{4.1}$$

where the last inequality holds with overwhelming probability true. Extractability means that $\mathcal{A}^*$ should be able to recover x_j from the transcripts of the two impersonation attacks and the known shares. We contend that this is (most probably) impossible *without* further assumptions. Condition (4.1) implies that the computations of Section 3.3.3 apply in the present context, meaning that the relevant extraction formula is the same as (3.36). However, $\mathcal{A}^*$ cannot know the logarithms $r_i,\ i \in J$, of the intercepted commitments and consequently cannot use this formula to recover x_j. Extraction is not possible in the present context.

Since there exist circumstances where extraction cannot complete without further assumptions, security is (most probably) unprovable without further assumptions as well. In particular, ORST security against impersonation attacks is (most probably) *not* reducible to DL hardness, at least not until a different extraction formula is discovered. Nevertheless, $\mathcal{A}^*$ would indeed be able to extract x_j if it could somehow see the logarithms of the intercepted commitments, e.g., if it controlled the respective shareholders' devices or could at least subvert their random number generator or other security sensitive module. We can codify such scenarios under the assumption that $\mathcal{A}^*$ is allowed to issue at most $t-1$ queries to a hypothetical DL oracle; extraction of x_j by means of (3.36) would then become possible, suggesting that security against impersonation attacks might as well be reducible to the $(t-1)$-OMDL hardness assumption (cf. Section 4.3). Note that $(t-1)$-OMDL is indeed an easier problem than DL, plausibly yielding the missing assumptions needed for security provability.

4.1.3. *Implications of* $(t-1)$-OMDL *hardness.* The black-box pattern of the previous section is rather restricted as compared to the general attack model of Section 4.1.1. This is because $\mathcal{A}$ does not influence the compromised shareholders, mounting its actual attack only after collecting their transcripts; in particular, the fabrication of u_j and s_j is deferred until after $u_i, c_i,\ i \in J$, have resolved, in order for extractability failure to be demonstrated in the simplest possible way (i.e., rewinding instead of some more complicated forking argument). The same argument indicates in which way security becomes provable under the stronger $(t-1)$-OMDL hardness assumption. Specifically, in order to be able to apply extraction, $\mathcal{A}^*$ should at least reproduce the conditions of Remark 3.3.3 and let $\mathcal{A}$ operate in that context. While this remains more restricted than the general attack model, it generalizes by far the restricted attack pattern of Section 4.1.2, allowing $\mathcal{A}$ to fabricate u_j before the rest shareholders send their commitments and also mount its own responses by obstructing their communication. We propose the following preliminary attack model for $\mathcal{A}$, representing a tradeoff between the general attack model of Section 4.1.1 and security provability under the $(t-1)$-OMDL hardness assumption.

- *Attack statement and corruption*:
  - $\mathcal{A}$ chooses $J \subset \{1, \ldots, n\}$ with $|J| = t-1$ and sends it along with some $Q \subset \{1, \ldots, n\}$ to its challenger $\mathcal{C}$.
  - $\mathcal{A}$ chooses $Q \subset \{1, \ldots, n\}$ with $Q \subset J$ or $J \subset Q$ and sends it along with J to its challenger $\mathcal{C}$.
  - $\mathcal{C}$ runs $(x_1, \ldots, x_n; y) \leftarrow D(x)$, $x \leftarrow_\$ \mathbb{Z}_q$ and sends $y, \{x_i\}_{i \in J}$ to $\mathcal{A}$.
- *Interception*: $\mathcal{C}$ samples $u_i \leftarrow_\$ \mathbb{G}$, $i \in J$ and sends $\{u_i\}_{i \in J}$ to $\mathcal{A}$.
- *Impersonation*: $\mathcal{A}$ outputs a proof of the form $\{(u_i, s_i)\}_{i \in Q}$.

$\mathcal{A}$ wins if $\{(u_i, s_i)\}_{i \in Q}$ verifies against y.

ORST security against impersonation attacks should mean that every polynomial-time adversary wins this game with negligible advantage.

Observe that no restrictions are yet imposed on $|Q|$, allowing the adversary to fabricate a valid proof with less than t components. This is plausible because proofs of this kind are known to exist in abundance (cf. Remarks 3.26 and 3.29). However, under the $(t-1)$-OMDL hardness assumption, fabricating such proofs can be ruled out as infeasible. Let $\mathcal{A}$ be a polynomial-time adversary that wins with $Q \subset J$ (i.e., $|Q| < t$). Write $J = \{i_1, \ldots i_{t-1}\}$ with

$$i_1 < \cdots < i_{t-1},$$

so that $Q = \{i_1, \ldots, i_m\}$ for some $m \in \{1, \ldots, t-1\}$. We construct a $(t-1)$-OMDL adversary $\mathcal{A}^*$ as a wrapper of $\mathcal{A}$ with equal time complexity and advantage. When given $w_0, \ldots, w_{t-1}$ by its challenger (cf. Game 4.6), $\mathcal{A}^*$ sets $y \leftarrow w_0$ and $u_{i_k} \leftarrow w_k$ for $1 \le k \le t-1$. Next, $\mathcal{A}^*$ samples $x_i \leftarrow_\$ \mathbb{Z}_q$ for $i \in J$ and forwards $y, \{x_i\}_{i \in J}$ to $\mathcal{A}$. By the security property of Shamir's secret sharing (cf. Remark A.5), y and $\{x_i\}_{i \in J}$ determine a (n, t)-sharing $x_1, \ldots, x_n$ of some $x \equiv \log y$ indistinguishably, meaning that $\mathcal{A}^*$ simulates perfectly the corruption phase of the attack game. Subsequently, $\mathcal{A}^*$ sends $\{u_i\}_{i \in J}$ to $\mathcal{A}$ and lets it proceed to impersonation; since its challenger has been perfectly simulated, $\mathcal{A}$ outputs a valid proof $\{(u_i, s_i)\}_{i \in Q}$ with probability equal to its advantage. Condition (3.20) is then satisfied and $\mathcal{A}^*$ can recover x by issuing at most $t-1$ discrete-logarithm queries. More accurately, $\mathcal{A}^*$ queries

$$r_{i_k} \leftarrow \mathcal{O}^{\mathsf{dlog}}_{\mathbb{G}}(u_{i_k}), \quad 1 \le k \le t-1,$$

and uses r_{i_k}, $1 \le k \le m$ to compute x. Finally, $\mathcal{A}^*$ sets $z_0 \leftarrow x$, and $z_k \leftarrow r_{i_k}$ for $1 \le k \le t-1$, and wins by forwarding $z_0, \ldots, z_{t-1}$ to its challenger. In short, if $\mathcal{A}$ succeeds with non-negligible probability, then $\mathcal{A}^*$ solves the $(t-1)$-OMDL problem with equal advantage, which contradicts the hardness assumption. This straight-line reduction shows that we can restrict attention to the case $J \subsetneq Q$ (i.e. $|Q| \ge t$), excluding the possibility of impersonating less than t shareholders without knowing x. This is evidently a security requirement for the ORST protocol, captured by the $(t-1)$-OMDL hardness assumption; see also Remarks 3.26 and 3.29.

4.2. **Security against impersonation attacks.** We conclude with the following working model for impersonation attacks against $\mathsf{ORST}_{\mathbb{G}, n, t}$, $1 \le t \le n < q$.

Game 4.2. ($\mathsf{ORST}^{\mathbf{imp}}_{\mathbb{G}, n, t}$ – *Impersonation*) Given adversary $\mathcal{A}$ and challenger $\mathcal{C}$,

- *Attack statement and corruption*:
  - $\mathcal{A}$ chooses $J \subset \{1, \ldots, n\}$ with $|J| = t - 1$ and sends it along with some $j \in \{1, \ldots, n\} \setminus J$ to the challenger. Denote $Q = J \cup \{j\}$.
  - $\mathcal{C}$ runs $(x_1, \ldots, x_n; y) \leftarrow D(x),\ x \leftarrow_\$ \mathbb{Z}_q$ and sends $y, \{x_i\}_{i \in J}$ to $\mathcal{A}$.
- *Interception*: $\mathcal{C}$ samples $u_i \leftarrow_\$ \mathbb{G},\ i \in J$ and sends $\{u_i\}_{i \in J}$ to $\mathcal{A}$.
- *Impersonation*: $\mathcal{A}$ outputs a proof of the form $\sigma = \{(u_i, s_i)\}_{i \in Q}$.

$\mathcal{A}$ wins if $\mathcal{V}(y, \sigma) = \mathsf{true}$.

The probability that $\mathcal{A}$ wins Game 4.2 is denoted by

$$\mathsf{Adv}^{\mathsf{ORST}}_{\mathbb{G},n,t}[\mathcal{A}]. \tag{4.3}$$

ORST *security against impersonation attacks* (*in* $\mathbb{G}$) should mean that (4.3) is negligible for every choice of n, t and every polynomial-time adversary $\mathcal{A}$.

Remark 4.4. (*Adversarial components*) By the security property of Shamir's secret sharing (cf. Remark A.5), the parameters j and J (and consequently Q) may be regarded as fixed for $\mathcal{A}$. In particular, we can regard $\mathcal{A}$ as a pair of non-interactive algorithms $(\mathcal{A}_0, \bar{\mathcal{A}})$ of the form

$$(j, J) \leftarrow \mathcal{A}_0(\cdot), \quad \{(u_i, s_i)\}_{i \in Q} \leftarrow \bar{\mathcal{A}}(y, \{x_i\}_{i \in J}, \{u_i\}_{i \in J}),$$

where $\bar{\mathcal{A}}$ consumes $\mathcal{A}$'s view during the impersonation phase in order to fabricate u_j and $\{s_i\}_{i \in Q}$ such that $\{(u_i, s_i)\}_{i \in Q}$ verifies. Note that, when using $\mathcal{A}$ in a black-box reduction, the logarithm of the fabricated commitment u_j cannot be assumed to be known and we will normally have to eliminate it from our computations.

Remark 4.5. (*Deterministic adversaries*) For every adversary $\mathcal{A}$ attacking Game 4.2, there exists a deterministic advesary $\mathcal{A}'$ of equal time complexity such that

$$\mathsf{Adv}^{\mathsf{ORST}}_{\mathbb{G},n,t}[\mathcal{A}'] \geq \mathsf{Adv}^{\mathsf{ORST}}_{\mathbb{G},n,t}[\mathcal{A}].$$

This follows from a generic argument regarding any adversary $\mathcal{A}$ attacking a sufficiently simple game G. Consider the residual determinisic strategies $\mathcal{A}_1, \ldots, \mathcal{A}_m$ obtained by fixing the possible values of $\mathcal{A}$'s random coins and let p_i denote the probability that $\mathcal{A}$ runs $\mathcal{A}_i$ (which is itself an adversary for G); we then have

$$\mathsf{Adv}^{\mathsf{G}}[\mathcal{A}] = \sum_{i=1}^{m} p_i \cdot \mathsf{Adv}^{\mathsf{G}}[\mathcal{A}_i]$$

Since $p_1 + \cdots + p_m = 1$, we conclude $\mathsf{Adv}^{\mathsf{G}}[\mathcal{A}_i] \geq \mathsf{Adv}^{\mathsf{G}}[\mathcal{A}]$ for some $i \in \{1, \ldots, m\}$.

4.3. **One-more discrete-logarithm (OMDL) hardness.** The OMDL problem family is a parametrized extension of the discrete-logarithm (DL) problem, yielding a sequence of increasingly easier related problems. It was introduced in [4] and [2] as a way to capture design properties that pragmatically guarantee security under DL hardness while most probably being themselves irreducible to the latter. Note that reducing security to an easier problem, i.e., a stronger hardness assumption, is usually equivalent to assumptions regarding the black-box adversary, i.e., a narrower threat model. This represents a tradeoff between the higher plausibility of the weaker assumption and having a security proof at hand, which may otherwise be intractable. Under the extra assumptions a formal security proof may become possible, while a proof under the weaker assumption alone may not even exist; since the solvability of the harder problem implies solvability of the easier one, the restricted proof can be regarded as evidence that security holds true under the weaker assumption. In our case, the pragmatic requirement that cannot be reduced to DL hardness is the knowledge property from Section 4.1.

By *discrete-logarithm* (DL) *oracle* for a cyclic group $\mathbb{G} = \langle g \rangle$ is meant a hypothetical polynomial-time procedure of the form $z \leftarrow \mathcal{O}^{\mathsf{dlog}}_{\mathbb{G}}(u)$, $u \in \mathbb{G}$, such that $u = g^z$. Given $\mathcal{O}^{\mathsf{dlog}}_{\mathbb{G}}$ and an integer $t \geq 1$, we consider the following problem.

Game 4.6. ($\mathsf{OMDL}_{\mathbb{G},\,t-1}$ – *One-more discrete-logarithm problem*) Given an adversary $\mathcal{A}$ with challenger $\mathcal{C}$,

- $\mathcal{C}$ samples $w_i \leftarrow_{\$} \mathbb{G},\ 0 \leq i \leq t-1$ and sends them to $\mathcal{A}$
- $\mathcal{A}$ outputs $z_0, \ldots, z_{t-1}$

$\mathcal{A}$ wins if $\bigwedge_{i=0}^{t-1} w_i = g^{z_i}$ and it has issued at most $t-1$ queries to $\mathcal{O}^{\mathsf{dlog}}_{\mathbb{G}}$.

The probability that $\mathcal{A}$ solves $\mathsf{OMDL}_{\mathbb{G},\,t-1}$ is denoted by

$$\mathsf{Adv}^{\mathsf{OMDL}}_{\mathbb{G},\,t-1}[\mathcal{A}]. \tag{4.7}$$

$(t-1)$-OMDL *hardness in* $\mathbb{G}$ is the assumption that (4.7) is negligible for every polynomial-time adversary. Evidently, 0-OMDL is equivalent to DL. If an adversary solves $\mathsf{ORST}_{\mathbb{G},\,t-1}$ for $t < t^*$, then it trivially solves $\mathsf{ORST}_{\mathbb{G},\,t^*-1}$, i.e., (t^*-1)-ODML is generally an easier problem than $(t-1)$-OMDL. In particular, $(t-1)$-OMDL, $t > 1$ is generally a stronger assumption than DL.

5. Security proof

In this section, we prove ORST security against impersonation attacks (cf. Section 4.2) under OMDL hardness (cf. Section 4.3) in the random oracle model. The rationale behind the attack model is found in Section 4.1.

Our reduction generalizes the rewinding argument of the non-distributed case in the threshold setting in order to extract the unknown share x_j. This is a non-trivial generalization. As indicated by the key-recovery attack of Section 3.3.3, the extractor comes with a residue stemming from the $t-1$ degrees of freedom in purturbing a valid proof with fixed commitments (cf. Remark 3.26); it becomes useful only if the logarithms of the intercepted commitments are known to the reduction algorithm, which is the reason for reducing to $(t-1)$-OMDL hardness. Moreover, extraction presupposes the fixture of all but one challenges, making rewinding useless in the threshold setting. We overcome this obstacle by employing the Local Forking Lemma (cf. Appendix B) and ensuring that the conditions for its applicability are satisfied. The attack simulator is presented in Section 5.4 and its components are developed in Sections 5.1 to 5.3.

5.1. **Index mapping formalism.** Given $J \subset \{1, \ldots, n\}$, we can represent any collection $\alpha_1, \ldots, \alpha_m$ with $|J| = m$ as $\{\beta_i\}_{i \in J}$ using the Zip operator below.

Algorithm 1 $\mathsf{Zip}(\alpha_1, \ldots, \alpha_m;\, J)$

$\{i_1, \ldots, i_m\} \leftarrow J$ with $i_1 < \cdots < i_m$
for $k = 1, \ldots, m$
  $\beta_{i_k} \leftarrow \alpha_k$
return $\{\beta_i\}_{i \in J}$

Algorithm 2 $\mathsf{Zip}^{-1}(\{\beta_i\}_{i \in J})$

$\{i_1, \ldots, i_m\} \leftarrow J$ with $i_1 < \cdots < i_m$
for $k = 1, \ldots, m$
  $\alpha_k \leftarrow \beta_{i_k}$
return $\alpha_1, \ldots, \alpha_m$

It should be clear that Zip preserves the uniform sampling.

Lemma 5.1. Given $J \subset \{1, \ldots, n\}$, $|J| = m \geq 1$, the probability distribution

$$\{\,\mathsf{Zip}(\alpha_1, \ldots, \alpha_m;\, J) : \alpha_i \leftarrow_\$ A,\ i = 1, \ldots, m\}$$

is identical to the distribution $\{\{\beta_i\}_{i \in J} : \beta_i \leftarrow_\$ A,\ i \in J\}$.

Proof. Follows directly from the definition of Zip. □

5.2. **Shamir sharing simulation.** Let $y^* \in \mathbb{G}$ and $x_1^*, \ldots, x_{t-1}^* \in \mathbb{Z}_q$, $t \leq n$. Given $J \subset \{1, \ldots n\}$ with $|J| = t-1$ and $j \in \{1, \ldots, n\} \setminus J$, we define

$$(\{x_i\}_{i \in J}, y) \leftarrow \mathcal{D}(y^*, x_1^*, \ldots, x_{t-1}^*\,; j\,, J) \tag{5.2}$$

to be Algorithm 3.

Algorithm 3 $\mathcal{D}(y^*, x_1^*, \ldots, x_{t-1}^*\,; j\,, J)$

1: Set $y_j \leftarrow y^*$
2: $\{x_i\}_{i \in J} \leftarrow \mathsf{Zip}(x_1^*, \ldots, x_{t-1}^*)$
3: Set $y_i \leftarrow g^{x_i},\ i \in J$
4: $Q \leftarrow J \cup \{\,j\,\}$
5: $y \leftarrow \prod_{i \in Q} y_i^{\lambda_i}$
6: **return** $\{x_i\}_{i \in J}, y$

Its properties are summarlized as follows.

Lemma 5.3. *Let $J \subset \{1, \ldots, n\}$, $|J| = t-1$ with $t \leq n$, and $j \in \{1, \ldots, n\} \setminus J$. Given* (5.2)*, there exists a unique (n, t)-sharing $(x_1, \ldots, x_n; y)$ such that*

$$\{x_i\}_{i \in J} = \mathsf{Zip}(x_1^*, \ldots, x_{t-1}^*)\ \wedge\ y^* = g^{x_j} \tag{5.4}$$

Moreover, the probability distribution

$$\{(\{x_i\}_{i \in J}, y) : \left\{\begin{matrix} (\{x_i\}_{i \in J}; y) \leftarrow \mathcal{D}(y^*, x_1^*, \ldots, x_{t-1}^*\,; j\,, J) \\ y^* \leftarrow_\$ \mathbb{G},\quad x_i^* \leftarrow_\$ \mathbb{Z}_q,\ 1 \leq i \leq t-1 \end{matrix}\right\}\}$$

is identical to $\{(\{x_i\}_{i \in J}, y) : (x_1, \ldots, x_n; y) \leftarrow D(x),\ x \leftarrow_\$ \mathbb{Z}_q\}$.

Proof. By the elementary properties of interpolation, $x_1^*, \dots, x_{t-1}^*$ and y^* uniquely determine a sharing subject to the constraint (5.4), with the outputs of $\mathcal{D}$ mapping to those of D bijectively. Equality of distributions follows from the security of the Shamir's sharing (cf. Remark A.5). □

5.3. **Hash oracles.** When querying a random oracle H (cf. Algorithm 5) in a simulation context, we usually need to keep track of distinct oracle queries. Given a finite tape of preallocated hashes $h = [h_1 \dots, h_m]$ and initially empty lookup tables Map_1, Map_2, we define the *hash oracle* $\mathcal{O}^{\mathsf{hash}}$ to be Algorithm 4.

Algorithm 4 $\mathcal{O}^{\mathsf{hash}}(u)$

```
if u ∉ Dom(Map_1)
    c ← h[ν]
    Map_1[u] ← c
    Map_2[u] ← ν
    ν ← ν + 1
else
    c ← Map_1[u]
return c
```

Algorithm 5 $H(u)$

```
if u ∉ Dom(Map)
    c ←$ C
    Map[u] ← c
else
    c ← Map[u]
return c
```

That is, $\mathcal{O}^{\mathsf{hash}}$ operates like a random oracle with lookup table Map_1 but draws hashes from h instead of sampling them uniformly; upon inserting a fresh hash to Map_1, it takes care to (i) update the tape pointer so that the next predefined hash be available on demand and (ii) insert the current pointer value to Map_2 so that the respective query can be easily tracked by order. Evidently, if $h_1, \dots, h_n$ are uniformly sampled from the codomain of H, then $\mathcal{O}^{\mathsf{hash}}$ cannot be distinguished from H. We state this remark formally for the sake of reference.

Lemma 5.5. (*Random oracle simulation*) *Given a random oracle H with codomain C and empty lookup table* Map, *initialize a hash oracle $\mathcal{O}^{\mathsf{hash}}$ over an empty tape h of size $m \geq 1$ and initially empty lookup tables* Map_1 *and* Map_1. *For any distinct elements $u_1, \dots, u_m$, the probability distribution*

$$\{(c_1, \dots, c_m) : \ c_\nu \leftarrow \mathcal{O}^{\mathsf{hash}}(u_\nu), \ h[\nu] \leftarrow_\$ C, \ \nu = 1, \dots, m\}$$

is identical to the distribution $\{(c_1, \dots, c_m) : c_\nu \leftarrow H(u_\nu), \ \nu = 1, \dots, m\}$.

Proof. Follows directly from the definition of $\mathcal{O}^{\mathsf{hash}}$. □

5.4. **Simulated impersonation attack.** We define an algorithm $\mathcal{S}[\mathcal{A}]$ that simulates Game 4.2 for the adversary $\mathcal{A}$, provided that H is modelled as a random oracle. Roughly speaking, $\mathcal{S}[\mathcal{A}]$ accepts the ingredients of $\mathcal{A}$'s view and outputs the fabricated proof along with a success index for $\mathcal{A}$'s impersonation attempt. The simulation is perfect meaning that, if run with uniform input, the probability of success equals the adversary's advantage of winning the real game.

More accurately, given a black-box adversary $\mathcal{A}$ to game $\mathsf{ORST}_{\mathbb{G},n,t}$, the desired simulation is a wrapper of $\mathcal{A}$ of the form

$$(k, (j, \sigma)) \ \leftarrow \ \mathcal{S}[\mathcal{A}](y^*, x_1^*, \dots, x_{t-1}^*, w_1, \dots, w_{t-1}; h_1, \dots, h_m),$$

arranging its input values appropriately in order to emulate $\mathcal{A}$'s execution environment. The values $y^*, x_1^*, \ldots, x_{t-1}^*$ will uniquely determine the attacked (n,t)-sharing, yielding the corrupted shares in the process. Similarly, $w_1, \ldots, w_{t-1}$ will be used as the intercepted commitments. Next, the embedded replica of $\mathcal{A}$ consumes its view in order to fabricate a purportedly valid proof σ with the difference that, instead of invoking H directly, it queries an embedded hash oracle $\mathcal{O}^{\mathsf{hash}}$ with pre-allocated hashes $h_1, \ldots, h_m$ (cf. Section 5.3). If needed, $\mathcal{A}$ may be reprogrammed without loss of generality to query hashes for all the commitments it sees (including the one fabricated by itself); in this case $m \geq 1$. Finally, an ORST-verifier $\mathcal{V}$ is included with access to the same hash oracle for verifying the output of the embedded adversary. The exact procedure is shown in Algorithm 6.

Algorithm 6 $\mathcal{S}[\mathcal{A}](y^*, x_1^*, \ldots, x_{t-1}^*, w_1, \ldots, w_{t-1}; h_1, \ldots, h_m)$

1: $h \leftarrow [h_1, \ldots, h_m],\ \nu \leftarrow 1,\ \mathsf{Map}_1 = \varnothing,\ \mathsf{Map}_2 = \varnothing$ // *Setup* $\mathcal{O}^{\mathsf{hash}}$
2: $(j, J) \leftarrow \mathcal{A}_0(\cdot)$ // *Attack statement*
3: $(\{x_i\}_{i \in J}, y) \leftarrow \mathcal{D}(y^*, x_1^*, \ldots, x_{t-1}^*; j, J)$ // *Corruption*
4: $\{u_i\}_{i \in J} \leftarrow \mathsf{Zip}(w_1, \ldots, w_{t-1})$ // *Interception*
5: $\{(u_i, s_i)\}_{i \in Q} \leftarrow \bar{\mathcal{A}}(y, \{x_i\}_{i \in J}, \{u_i\}_{i \in J})$ // *Impersonation*
6: $res \leftarrow \mathcal{V}(y, \{(u_i, s_i)\}_{i \in Q})$
7: **for** $i \in Q$
8: $\quad c_i \leftarrow \mathsf{Map}_1[u_i]$
9: $\sigma \leftarrow \{(u_i, c_i, s_i)\}_{i \in Q}$
10: $k \leftarrow \mathsf{Map}_2[u_j]$
11: **if** $res = \mathsf{true}$
12: $\quad$ **return** $(k, (j, \sigma))$
13: **return** $(0, (j, \sigma))$

$\mathcal{O}^{\mathsf{hash}}$ is first initialized with tape $[h_1, \ldots, h_m]$ and its lookup tables set to empty. After $\mathcal{A}$ makes its attack statement, the virtual challenger accordingly simulates the unique sharing $(x_1, \ldots, x_n;\, y) \leftarrow D(x)$, $x \leftarrow \mathbb{Z}_q$ subject to the constraint (5.4); in particular, $x_j = \log y^*$. It then generates commitments $\{u_i\}_{i \in J}$ and feeds them along with y and $\{x_i\}_{i \in J}$ to $\mathcal{A}$, who proceeds to impersonation and generates a proof of the expected form; in particular, $Q = J \cup \{j\}$. Note that all oracle queries are implicitly issued at step 5, including the challenges $c_i \leftarrow \mathcal{O}^{\mathsf{hash}}(u_i)$, $i \in Q$, which are collected and attached to the fabricated proof for later usage (steps 7-9). We also locate the unique $k \in \{1, \ldots, m\}$ for which $c_j = h[k]$ (step 10), indicating that the fabricated commitment was submitted at the k-th distinct query. If $\mathcal{A}$ succeeds in its impersonation attempt, the simulation returns $(k, (j, \sigma))$; otherwise, it returns $(0, (j, \sigma))$ to indicate failure.

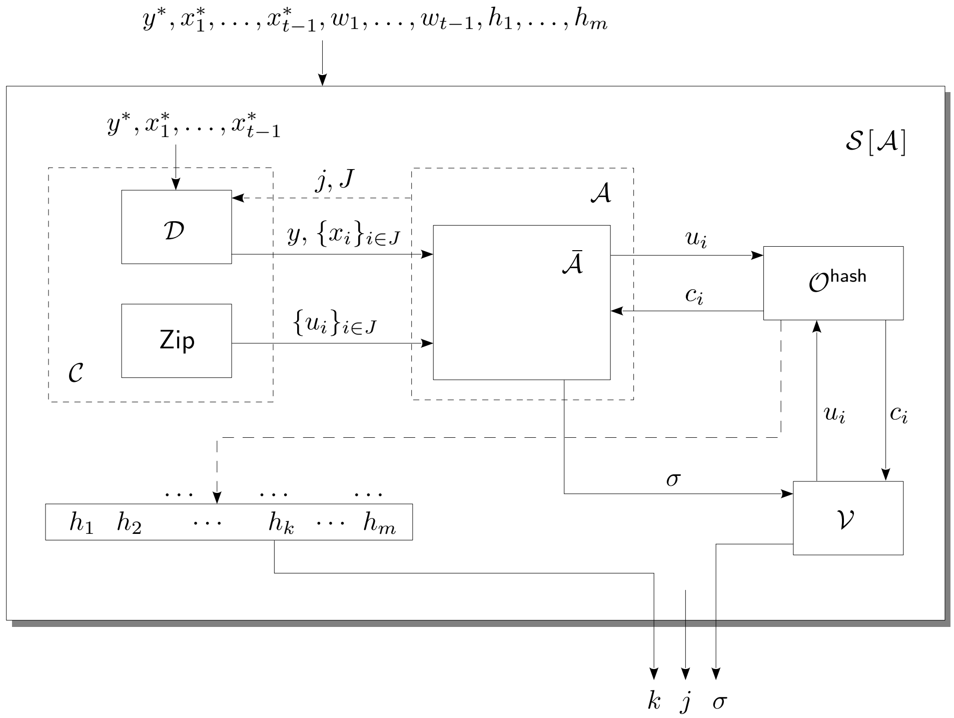

FIGURE 1. The $\mathsf{ORST}^{\mathbf{imp}}_{\mathbb{G},n,t}$ simulator

Proposition 5.6. ($\mathsf{ORST}^{\mathbf{imp}}_{\mathbb{G},n,t}$ – *Perfect simulation*) *If H is modelled as a random oracle, then $\mathcal{S}[\mathcal{A}]$ simulates Game 4.2 perfectly if invoked with uniformly random input, in the sense that the real and simulated game executions are indistinguishable from the adversary's viewpoint. More accurately, the probability distribution of $\mathcal{A}$'s view in the real attack game is identical to the distribution of $\bar{\mathcal{A}}$'s input in*

$$\begin{aligned} (k,(j,\sigma)) \leftarrow \mathcal{S}[\mathcal{A}](y^*, x_1^*, \ldots, x_{t-1}^*, w_1, \ldots, w_{t-1}; h_1, \ldots, h_m), \\ y^* \leftarrow_\$ \mathbb{G}, \quad x_i^* \leftarrow_\$ \mathbb{Z}_q, \quad w_i \leftarrow_\$ \mathbb{G}, \quad 1 \le i \le t-1, \\ h_\nu \leftarrow_\$ \mathbb{Z}_q,\ 1 \le \nu \le m \end{aligned} \tag{5.7}$$

In the context of this simulation, if $\bar{\mathcal{A}}$ succeeds in its impersonation attempt, i.e., $k \ge 1$, the input value h_k maps to $c_j \equiv H(u_j)$ of the real game execution.

Proof. In the real game execution, the adversarial view consists of the corrupted shares $\{x_i\}_{i \in J}$ along with the combined public key y, the intercepted commitments $\{u_i\}_{i \in Q}$ and the hashes queried during the impersonation phase. These three blocks are independent from each other; clearly, the first and second blocks are independent, while the third block is independent from the others because H is a random oracle. It thus suffices to check that the respective blocks have identical distributions in the simulated execution (5.7). The first block is perfectly simulated due to Lemma 5.3. The second block is also perfectly simulated due to Lemma 5.1.

Finally, the third block is perfectly simulated due to Lemma 5.5. The fact that $h_k = h[k]$ corresponds to $H(u_j)$ follows directly from step 10, Algorithm 6. $\square$

We denote by $\mathsf{acc}(\mathcal{S}[\mathcal{A}])$ the probability that the embedded adversary succeeds in the context of a simulated attack when run with uniformly random input (cf. (5.7)). This is related to the real adversarial advantage as follows.

Corollary 5.8. *Given an adversary $\mathcal{A}$ for Game 4.2, there holds*

$$\mathsf{acc}(\mathcal{S}[\mathcal{A}]) \;=\; \mathsf{Adv}^{\mathsf{ORST}}_{\mathbb{G},n,t}[\mathcal{A}] \tag{5.9}$$

Proof. Direct consequence of Proposition 5.6. $\square$

5.5. **Local fork and extraction.** In the non-distributed case $t = 1$, the simulator $\mathcal{S}[\mathcal{A}]$ yields the execution environment of the rewindable adversary $\mathcal{A}$ that is usually employed to reduce the security of ordinary Schnorr identification to DL hardness in the random oracle model. In the non-distributed case, the extractor is generalized as (3.36) provided that the forked attack runs with the same hashes except for $c_j \leftarrow H(u_j)$. This implies that the rewinding argument cannot be effective in the threshold setting; indeed, resetting the attack before $\mathcal{A}$ sends u_j does not guarantee that subsequent oracle responses remain the same and, in fact, they differ almost certainly. Nevertheless, when invoking $\mathcal{S}[\mathcal{A}]$ we predefine the hashes $h_1, \ldots, h_m$ returned by the embedded oracle in respective order, which allows to rerun the simulator with the same hashes except for $c_j = h_k$. In this case, the Local Forking Lemma (cf. Lemma B.5) becomes applicable in place of the general forking lemma (cf. [3] or reset lemma, [4]), provided that $\mathcal{A}$ is deterministic.

Recall that the local fork of $\mathcal{S}[\mathcal{A}]$ is an algorithm of the form

$$(b, \mathsf{out}) \;\leftarrow\; \mathsf{LocalFork}_{\mathcal{S}[\mathcal{A}]}(y^*, x_1^*, \ldots, x_{t-1}^*, w_1, \ldots, w_{t-1})$$

(cf. Appendix B) operating as follows. If the embedded adversary $\mathcal{A}$ succeeds in its first impersonation attempt, the fork reexecutes the simulation with the same input values except for the k-th predefined hash, which is afresh uniformly sampled as h_k^*; otherwise, the fork aborts ($b = 0$). When the second simulation completes, the fork checks if $\mathcal{A}$ succeeds again under the condition $h_k^* \neq h_k$; in this case, it returns the successive outputs $\tau \equiv (j, \sigma)$ and $\tau^* \equiv (j, \sigma^*)$ of both simulations; otherwise, it aborts ($b = 0$).

Evidently, $\mathsf{forkAcc}(\mathcal{S}[\mathcal{A}])$ (cf. def. (B.4)) expresses here the probability that $\mathcal{A}$ succeeds both in its first and forked impersonation attempt when consuming the same uniformly random view (except for the k-th distinct hash). In this case, the outputs of the two simulated attacks are correlated as follows.

Proposition 5.10. (*Local fork and extraction*) *Let $\mathcal{A}$ be a deterministic adversary of Game 4.2 and $\tau = (j, \sigma)$, $\tau^* = (j, \sigma^*)$ be outputs of a non-aborting execution*

$$(1, (\tau, \tau^*)) \;\leftarrow\; \mathsf{LocalFork}_{\mathcal{S}[\mathcal{A}]}(y^*, x_1^*, \ldots, x_{t-1}^*, w_1, \ldots, w_{t-1})$$

Then σ, σ^* are of the form

$$\{(u_i, c_i, s_i)\}_{i \in Q},\;\; \{(u_i, c_i^*, s_i^*)\}_{i \in Q} \tag{5.11}$$

respectively and subject to the constraint

$$c_j \neq c_j^* \ \wedge \ c_i = c_i^*,\, \forall i \in Q \setminus \{j\}. \tag{5.12}$$

Moreover, the extraction formula

$$x_j \ = \ \frac{s_j - s_j^*}{c_j - c_j^*} \ + \ \frac{1}{\mu_j} \sum_{i \in J} \Big(\mu_i (s_i - r_i - c_i x_i) - \mu_i^* (s_i^* - r_i - c_i x_i) \Big) \tag{5.13}$$

for $x_j \equiv \log y^*$ holds true, where $r_i \equiv \log u_i$ and $\{\mu_i\}_{i \in Q}$, $\{\mu_i^*\}_{i \in Q}$ are the weights of σ and σ^* respectively.

Proof. Let $\sigma = \{(u_i, c_i, s_i)\}_{i \in Q}$, $\sigma^* = \{u_i^*, c_i^*, s_i^*\}_{i \in Q}$ so that $u_i^* = u_i, \forall i \neq j$. Fabricating the j-th commitment completes before submitting it to the hash oracle and is thus unaffected by the k-th hash. Since $\mathcal{A}$ is deterministic and its view is the same in both simulations (apart from the k-th hash), we conclude that $u_j = u_j^*$. The constraint regarding challenges follows because $c_j^* = h_k^* \neq h_k = c_j$ and the rest are drawn from $\{h_\nu\}_{\nu \neq k}$. We proceed to the extraction formula. Since $c_i = c_i^*$ for $i \neq j$, by definition of weights we get

$$\mu_j = \mu_j^* \tag{5.14}$$

(cf. Definition 3.8). Since both σ, σ^* are valid, Corollary 3.30 implies

$$\sum_{i \in Q} \mu_i (s_i - r_i - c_i x_i) = 0\,, \quad \sum_{i \in Q} \mu_i^* (s_i^* - r_i - c_i^* x_i) = 0\,.$$

Substracting terms and applying (5.14) yields

$$\mu_j (s_j - s_j^* - (c_j - c_j^*) x_j) \ + \sum_{i \in J} \left(\mu_i (s_i - r_i - c_i x_i) - \mu_i^* (s_i^* - r_i - c_i x_i) \right) = 0\,.$$

Extraction follows because $c_j - c_j^* \neq 0$. □

5.6. **Main theorem.** We proceed to the final reduction argument.

Theorem 5.15. (ORST – *Security against impersonation attacks*) $\mathsf{ORST}_{\mathbb{G},n,t}$ *is secure against impersonation attacks under $(t-1)$-OMDL hardness in the random oracle model. In particular, if H is modelled as a random oracle, for every polynomial-time adversary $\mathcal{A}$ that attacks $\mathsf{ORST}^{\mathbf{imp}}_{\mathbb{G},n,t}$ there exists a polynomial-time adversary $\mathcal{A}^*$ that solves $\mathsf{OMDL}_{\mathbb{G},t-1}$ with advantage*

$$\mathsf{Adv}^{\mathsf{OMDL}}_{\mathbb{G},t-1}[\mathcal{A}^*] \ \geq \ \frac{1}{q_{\mathsf{ro}} + 1} \, \mathsf{Adv}^{\mathsf{ORST}}_{\mathbb{G},n,t}[\mathcal{A}]^2, \tag{5.16}$$

where $q_{\mathsf{ro}} + 1$ is the number of distinct queries issued by $\mathcal{A}$ to the random oracle.

Proof. We construct $\mathcal{A}^*$ as a wrapper of $\mathsf{LocalFork}_{\mathcal{S}[\mathcal{A}]}$; after forking a successful impersonation attack, the wrapper should namely be able to recover the unknown share by issuing at most $t-1$ discrete-logarithm queries and then use it to solve $\mathsf{OMDL}_{\mathbb{G},t-1}$ with comparable chances. We assume without loss of generality that $\mathcal{A}$ is deterministic (cf. Remark 4.5). When given $w_0, w_1, \ldots, w_{t-1}$ by its challenger (cf. Game 4.6), $\mathcal{A}^*$ operates as shown in Algorithm 7. Observe that $\mathcal{A}^*$ aborts if the fork aborts, that is, unless both impersonation attacks by $\mathcal{A}$ are successful; otherwise, the fork fulfills the conditions of Proposition 5.10 and the proofs fabricated by $\mathcal{A}$

are as shown at steps 7 and 8. By definition of $\mathcal{S}[\mathcal{A}]$, the attacks of step 3 unfold against the sharing $(x_1, \ldots, x_n;\, y)$ that is uniquely determined by the constraint

$$\{x_i\}_{i \in J} = \mathsf{Zip}(x_1^*, \ldots, x_{t-1}^*) \ \wedge \ y^* = g^{x_j} \tag{5.17}$$

(cf. Lemma 5.3). Similarly, the intercepted commitments u_i, $i \in J$, comprise of the input values $w_1, \ldots, w_{t-1}$; in particular,

$$\{u_i\}_{i \in J} = \mathsf{Zip}(w_1, \ldots, w_{t-1}) \tag{5.18}$$

(cf. Algorithm 6, step 4). In steps 9-14, $\mathcal{A}^*$ computes the parameters needed for the extraction at step 15. The compromised shares are recomputed at step 13 using the first part of constraint (5.17), while the logarithms of u_i, $i \in J$, are retrieved at step 14 by submitting them to the DL oracle. Since $y^* = w_0$, Proposition 5.10 ensures that the computation of step 15 yields $z_0 = \log w_0$. Similarly, due to (5.18), the operation of step 17 yields $z_i = \log w_i$, $1 \leq i \leq t-1$. In short, $z_0, z_1, \ldots, z_{t-1}$ is the solution to the original problem instance and $\mathcal{A}^*$ has issued at most $t-1$ distinct queries to the DL oracle.

This discussion shows that $\mathcal{A}^*$ solves $\mathsf{OMDL}_{\mathbb{G},t-1}$ exactly if it does not abort. Specifically, since $w_0, w_1, \ldots, w_{t-1}$ are uniformly sampled by its challenger, $\mathcal{A}^*$'s advantage is equal to the probability that

$$(b, \mathsf{out}) \ \leftarrow \ \mathsf{LocalFork}_{\mathcal{S}[\mathcal{A}]}(y^*, x_1^*, \ldots, x_{t-1}^*, w_1, \ldots, w_{t-1})$$

is a non-aborting execution when run with uniform input; or, in other words,

$$\mathsf{Adv}^{\mathsf{OMDL}}_{\mathbb{G},t-1}[\mathcal{A}^*] \ = \mathsf{forkAcc}(\mathcal{S}[\mathcal{A}])$$

(cf. steps 1-2 and (B.4)). Let $m = q_{\mathsf{ro}} + 1$ be the number of distinct queries issued by $\mathcal{A}$ to $\mathcal{O}^{\mathsf{hash}}$ in the context of $\mathcal{S}[\mathcal{A}]$. The desired inequality follows from the Local Forking Lemma and Corollary 5.8. $\square$

Remark 5.19. (*Leaked public shares*) The adversary $\mathcal{A}$ of Game 4.2 is not given the public shares $y_i = g^{x_i}$, $1 \leq i \leq n$. This is plausible because these are not advertised or used in the ORST protocol. On the other hand, a real-life attacker has most probably knowledge of these shares before mounting its actual impersonation attack. For example, a man-in-the middle who intercepts multiple identification sessions can infer the public shares from the captured proof packets in the form (3.34); or, if the shareholders store the public shares locally for, say, mutual verification purposes, an attacker who compromises the device of a single shareholder can learn them immediately (if not erased). However, the proof of Theorem 5.15 remains intact under this stronger assumption. Supposing that the y_i's must be given to $\bar{\mathcal{A}}$ during the corruption phase of the simulated impersonation attack (cf. steps 3-5, Algorithm 6), these can be easily computed as

$$y_k \leftarrow \sqrt[\lambda_k^k]{y\Big(\prod_{i \in J} y_i^{\lambda_i^k}\Big)^{-1}}, \ \ k \in \{1, \ldots, n\} \setminus J,$$

where $y_i \equiv g^{x_i}$ for $i \in J$ and $\{\lambda_i^k\}_i$ are the Lagrange interpolation coefficients for $J \cup \{k\}$ (cf. Remark A.2).

Algorithm 7 $\mathcal{A}^*(w_0, w_1, \ldots, w_{t-1})$

Set $y^* \leftarrow w_0$

Set $x_i^* \leftarrow_\$ \mathbb{Z}_q$, $1 \le i \le t-1$

$(b, \mathsf{out}) \leftarrow \mathsf{LocalFork}_{\mathcal{S}[\mathcal{A}]}(y^*, x_1^*, \ldots, x_{t-1}^*, w_1, \ldots, w_{t-1})$

if $b = 0$

  return $\perp$

Parse $(\tau, \tau^*) \leftarrow \mathsf{out}$

Parse $(j, \{(u_i, c_i, s_i)\}_{i \in Q}) \leftarrow \tau$

Parse $(j, \{(u_i, c_i^*, s_i^*)\}_{i \in Q}) \leftarrow \tau^*$

$J \leftarrow Q \setminus \{j\}$

Compute the Lagrange interpolation coefficients $\{\lambda_i\}_{i \in Q}$

Compute

$$\mu_j \leftarrow \lambda_j \prod_{i \in J} c_i$$

for $i \in J$, compute

$$\mu_i \leftarrow \lambda_i \prod_{k \in Q \setminus \{i\}} c_k, \qquad \mu_i^* \leftarrow \lambda_i \prod_{k \in Q \setminus \{i\}} c_k^*$$

$\{x_i\}_{i \in J} \leftarrow \mathsf{Zip}(x_1^*, \ldots, x_{t-1}^*; J)$

Query $r_i \leftarrow \mathcal{O}_{\mathbb{G}}^{\mathsf{dlog}}(u_i),\ i \in J$ // *max* $t-1$ *queries*

Compute

$$x_j \leftarrow \frac{s_j - s_j^*}{c_j - c_j^*} + \frac{1}{\mu_j} \sum_{i \in J} \Big(\mu_i(s_i - r_i - c_i x_i) - \mu_i^*(s_i^* - r_i - c_i x_i)\Big)$$

Set $z_0 \leftarrow x_j$

$(z_1, \ldots, z_{t-1}) \leftarrow \mathsf{Zip}^{-1}(\{r_i\}_{i \in J})$

return $z_0, z_1, \ldots, z_{t-1}$

Appendix A. Shamir's secret sharing

We list for reference some basic facts regarding the secret sharing scheme introduced by Shamir [11]. Given a cyclic group $\mathbb{G} = \langle g \rangle$ of order q, we denote by $(\mathbb{Z}_q)_{t-1}[X]$ the ring of polynomials of degree at most $t-1$ over $\mathbb{Z}_q$.

Definition A.1. The *Shamir* (n,t)*-sharing scheme*, $1 \le t \le n < q$, is the probabilistic algorithm $(x_1, \ldots, x_n;\, y) \leftarrow D(x)$, $x \in \mathbb{Z}_q$ defined as

$$y \leftarrow g^x, \quad x_i \leftarrow f(x_i), \quad 1 \leq i \leq n, \quad f \leftarrow_\$ \left(\mathbb{Z}_q\right)_{t-1}[X] \text{ with } f(0) = x$$

The space of possible Shamir (n, t)-sharings of x is denoted by

$$D_x = \{(x_1, \ldots, x_n;\, y) \mid (x_1, \ldots, x_n;\, y) \leftarrow D(x)\}$$

Remark A.2. (*Reconstruction formula*) Given $(x_1, \ldots, x_n;\, y) \in D_x$, the condition

$$x = \sum_{i \in Q} \lambda_i x_i \iff |Q| \geq t \iff y = \prod_{i \in Q} y_i^{\lambda_i}, \tag{A.3}$$

holds true, where $\{\lambda_i\}_{i \in Q}$ are the Lagrange interpolation coefficients and $y_i \equiv g^{x_i}$ are the public shares.

Remark A.4. ($t-1$ *degrees of freedom*) Given $J \subset \{1, \ldots, n\}$ with $|J| = t - 1$, pick $x_i^* \in \mathbb{Z}_q$, $i \in J$. Given $x \in \mathbb{Z}_q$, there is a unique $(x_1, \ldots, x_n;\, y) \in D_x$ such that

$$x_i = x_i^*, \, \forall i \in J$$

Remark A.5. (*Security of Shamir's sharing*) Given x, $x^* \in \mathbb{Z}_q$ and $J \subset \{1, \ldots, n\}$ with $|J| = t - 1$, the probability distributions

$$\{\{x_i\}_{i \in J} : (x_1, \ldots, x_n;\; y) \leftarrow D(x)\}, \quad \{\{x_i\}_{i \in J} : (x_1, \ldots, x_n;\; y) \leftarrow D(x^*)\}$$

are indistinguishable.

Appendix B. Local forking lemma

The Forking Lemma was introduced by Pointcheval and Stern ([8], [9]) as part of the rewinding methodology for proving the security of Fiat-Shamir based signature schemes in the random oracle. It was later elaborated as General Forking Lemma by Bellare and Neven [3], who abstracted away the signature specific details. The Local Forking Lemma is a variance by [1] which dispenses with rewinding completely, allowing to fork a (deterministic) process by reprogramming the oracle at a single location as opposed to every location after the fork.

Consider a deterministic algorithm of the form

$$\mathcal{S} : \Theta \times C^m \to \{0, 1, \ldots, m\} \times T, \quad (k, \tau) \leftarrow \mathcal{S}(\theta;\, h_1, \ldots, h_m), \tag{B.1}$$

running internally a hash oracle with preallocated hashes $h_1, \ldots, h_m$ in respective order (cf. Section 5.3). Think of θ as the random coins, k as the *main* output and τ as the *side* output. If $k \geq 1$, then the main output can be thought of as indicating some peculiarity of the k-th distinct oracle query regarding the side output, while the case $k = 0$ is interpreted as failure. We denote by

$$\mathsf{acc}(\mathcal{S}) = Pr[k \geq 1 : \left\{ \begin{array}{c} (k, \tau) \leftarrow \mathcal{S}(\theta;\, h_1, \ldots, h_m), \quad \theta \leftarrow_\$ \Theta \\ h_\nu \leftarrow_\$ C, \quad 1 \leq \nu \leq m \end{array} \right\}] \tag{B.2}$$

the probability that $\mathcal{S}$ accepts taken over the possible coins and oracle outputs.

Definition B.3. The *local fork of* $\mathcal{S}$ is Algorithm 8.

That is, if $\mathcal{S}$ accepts, then the fork reruns it with the same input except for the k-th hash, which is afresh uniformly sampled; otherwise it aborts. When the second execution completes, the fork checks whether $\mathcal{S}$ accepts again under the constraint $h_k^* \neq h_k$ and returns both side outputs; otherwise, it aborts. We denote by

$$\mathsf{forkAcc}(\mathcal{S}) \;=\; Pr\,[\,b = 1 :\; (b, \mathsf{out}) \leftarrow \mathsf{LocalFork}_{\mathcal{S}}(\theta),\;\; \theta \leftarrow_{\$} \Theta\,] \tag{B.4}$$

the probability that the local fork does not abort taken over the possible random coins of $\mathcal{S}$. The local forking lemma asserts the following lower bound.

Lemma B.5. (*Local Forking Lemma*, [1]) *For every deterministic algorithm $\mathcal{S}$ as in* (B.1) *issuing m distinct queries to a hash oracle with tape $[h_1, \dots, h_m]$,*

$$\mathsf{forkAcc}(\mathcal{S}) \;\geq\; \frac{1}{m}\, \mathsf{acc}(\mathcal{S})^{\,2} \tag{B.6}$$

Algorithm 8 $\mathsf{LocalFork}_{\mathcal{S}}(\theta)$

$h_\nu \leftarrow_{\$} C, \quad 1 \leq \nu \leq m$

$(k, \tau) \leftarrow \mathcal{S}(\theta; h_1, \dots, h_m)$

if $k = 0$

  return $(0, \perp)$

$h_k^* \leftarrow_{\$} C$

$(k^*, \tau^*) \leftarrow \mathcal{S}\,(\theta\,; h_1, \dots, h_{k-1}, h_k^*, h_{k+1}, \dots h_m)$

if $k^* = k \;\wedge\; h_k \neq h_k^*$

  return $(1, (\tau, \tau^*))$

return $(0, \perp)$

References

[1] Mihir Bellare, Wei Dai, and Lucy Li. The local forking lemma and its application to deterministic encryption. Cryptology ePrint Archive, Paper 2019/1017, 2019.

[2] Mihir Bellare, Chanathip Namprempre, David Pointcheval, and Michael Semanko. The one-more-rsa-inversion problems and the security of chaum's blind signature scheme. *J. Cryptology*, 16:185–215, 2003.

[3] Mihir Bellare and Gregory Neven. Multi-signatures in the plain public-key model and a general forking lemma. In *Conference on Computer and Communications Security*, 2006.

[4] Mihir Bellare and Adriana Palacio. Gq and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks. In *Advances in Cryptology - CRYPTO 2002, 22nd Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 2002, Proceedings*, volume 2442 of *Lecture Notes in Computer Science*, pages 162–177. Springer, 2002.

[5] Marcel Keller, Gert Læssøe Mikkelsen, and Andy Rupp. Efficient threshold zero-knowledge with applications to user-centric protocols. Cryptology ePrint Archive, Paper 2012/306, 2012.

[6] Chelsea Komlo and Ian Goldberg. FROST: Flexible round-optimized schnorr threshold signatures. Cryptology ePrint Archive, Paper 2020/852, 2020.

[7] Ueli M. Maurer. Unifying zero-knowledge proofs of knowledge. In Bart Preneel, editor, *Progress in Cryptology - AFRICACRYPT 2009, Second International Conference on Cryptology in Africa, Gammarth, Tunisia, June 21-25, 2009. Proceedings*, volume 5580 of *Lecture Notes in Computer Science*, pages 272–286. Springer, 2009.

[8] David Pointcheval and Jacques Stern. Security proofs for signature schemes. In *Advances in Cryptology - EUROCRYPT '96, International Conference on the Theory and Application of Cryptographic Techniques, Saragossa, Spain, May 12-16, 1996, Proceeding*, volume 1070 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 1996.

[9] David Pointcheval and Jacques Stern. Security arguments for digital signatures and blind signatures. *J. Cryptology*, 13:361–396, 2000.

[10] Claus-Peter Schnorr. Efficient signature generation by smart cards. *Journal of Cryptology*, 4:161–174, 2004.

[11] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[12] Douglas R. Stinson and Reto Strobl. Provably secure distributed schnorr signatures and a (t, n) threshold scheme for implicit certificates. In Vijay Varadharajan and Yi Mu, editors, *Information Security and Privacy*, pages 417–434, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.