Continuously Non-Malleable Secret Sharing for General Access Structures

Gianluca Brian[1], Antonio Faonio[2], and Daniele Venturi[1]

[1]*Department of Computer Science, Sapienza University of Rome, Rome, Italy*
[2]*IMDEA Software Institute, Madrid, Spain*

May 31, 2019

Abstract

We study leakage-resilient continuously non-malleable secret sharing, as recently introduced by Faonio and Venturi (CRYPTO 2019). In this setting, an attacker can continuously tamper and leak from a target secret sharing of some message, with the goal of producing a modified set of shares that reconstructs to a message related to the originally shared value. Our contributions are two fold.

- In the plain model, assuming one-to-one one-way functions, we show how to obtain noisy-leakage-resilient continuous non-malleability for arbitrary access structures, in case the attacker can continuously leak from and tamper with all of the shares independently.
- In the common reference string model, we show how to obtain a new flavor of security which we dub bounded-leakage-resilient continuous non-malleability under joint k-selective partitioning. In this model, the attacker is allowed to partition the target n shares into k non-overlapping subsets, and then can continuously leak from and tamper with the shares within each subset jointly. Our construction works for arbitrary access structures, and assuming (doubly enhanced) trapdoor permutations and collision-resistant hash functions, we achieve a concrete instantiation for $k \in O(n/\log n)$.

Prior to our work, there was no secret sharing scheme achieving continuous non-malleability against joint tampering, and the only known scheme for independent tampering was tailored to threshold access structures.

Keywords: Secret sharing, Non-malleability, Leakage resilience.

Contents

1 Introduction

A non-malleable secret sharing for an access structure $\mathcal{A}$ over n parties allows to share a secret message m into n shares $s = (s_1, \ldots, s_n)$, in such a way that the following properties are guaranteed.

Privacy: No attacker given the shares belonging to an arbitrary unauthorized subset $\mathcal{U} \notin \mathcal{A}$ of the players can infer any information on m.

Non-malleability: No attacker tampering with all of the shares via some function $f \in \mathcal{F}$ within some family of allowed[1] modifications can generate a mauled secret sharing $\tilde{s} = f(s)$ that reconstructs to $\tilde{m} \neq m$ related to m.

Sometimes, non-malleability is considered together with leakage resilience. This means that the attacker can additionally leak partial information $g(s)$ from all of the shares (via functions $g \in \mathcal{G}$) before launching a tampering attack. Leakage resilience typically comes in one of two flavors: *bounded leakage* (i.e,. there is a fixed upper bound on the maximum amount of information retrieved from the shares) or *noisy leakage* (i.e., the length of the retrieved information is arbitrary as long as it does not decrease the entropy of the shares by too much).

In this work we focus on leakage-resilient *continuous* non-malleability with *adaptive concurrent reconstruction*, as recently introduced by Faonio and Venturi [FV19].[2] Here, the attacker can (leak from and) tamper poly-many times with a target secret sharing using functions $f^{(q)} \in \mathcal{F}$ as above, and for each tampering query q it can also choose adaptively the reconstruction set $\mathcal{T}^{(q)} \in \mathcal{A}$ used to determine the reconstructed message. There are only two limitations: First, the attacker is computationally bounded; second, the experiment stops (we say it "self-destructs") after the first tampering query yielding an invalid set of shares. Both limitations are *inherent* for continuous non-malleability [FMNV14, BS18, FV19].

The only known scheme achieving such a strong flavor of non-malleability is the one by Faonio and Venturi, which tolerates the families $\mathcal{F}$ and $\mathcal{G}$ of *independent tampering/leakage*, i.e. for each query q we have $f^{(q)} = (f_1^{(q)}, \ldots, f_n^{(q)}) \in \mathcal{F}$ where $f_i^{(q)}$ gets as input the i-th share (and similarly $g^{(q)} = (g_1^{(q)}, \ldots, g_n^{(q)}) \in \mathcal{G}$). The access structure $\mathcal{A}$ supported by their construction is the τ-threshold access structure—i.e., any subset of at most τ players has no information about the message—with the caveat that reconstruction works with at least $\tau + 2$ shares, namely a *ramp* secret sharing, thus leaving a minimal gap between the reconstruction and privacy threshold. The following natural question arise:

Problem 1. Can we obtain leakage-resilient continuously non-malleable secret sharing against independent leakage/tampering, for general access structures?

Another open question is whether leakage-resilient continuous non-malleability is achievable for stronger tampering and leakage families $\mathcal{F}, \mathcal{G}$, e.g. in case the attacker can leak from and manipulate subsets of the shares jointly.

Problem 2. Can we obtain leakage-resilient continuously non-malleable secret sharing against joint leakage/tampering?

1.1 Our Contributions

We make significant progress towards solving the above problems. In particular, our first contribution is a positive answer to Problem 1:

[1] It is easy to see that non-malleability is impossible for arbitrary (polynomial-time) tampering.

[2] From now on, we omit to explicitly mention the feature of adaptive concurrent reconstruction and simply talk about continuous non-malleability.

Theorem 1 (Informal)**.** *Assuming one-to-one one-way functions, for any access structure $\mathcal{A}$ over n parties there exists a noisy-leakage-resilient continuously non-malleable secret sharing scheme realizing $\mathcal{A}$ against independent leakage and tampering, in the plain model.*

Our second contribution is a positive answer to Problem 2 assuming trusted setup, in the form of a common reference string (CRS). More in details, we put forward a new security notion for secret sharing dubbed continuous non-malleability under *joint k-selective partitioning*. This roughly means that the attacker, after seeing the CRS, must commit to a partition of the set $[n]$ into k (non-overlapping) subsets $(\mathcal{B}_1, \ldots, \mathcal{B}_k)$; hence, the adversary can jointly, and continuously, tamper with and leak from each collection $s_{\mathcal{B}_i}$ of the shares.[3]

Theorem 2 (Informal)**.** *Assuming (doubly-enhanced) trapdoor permutations and collision-resistant hash functions, for any access structure $\mathcal{A}$ over n parties there exists a bounded-leakage-resilient continuously non-malleable secret sharing scheme realizing $\mathcal{A}$ against joint leakage and tampering under $O(n/\log n)$-selective partitioning, in the CRS model.*

1.2 Related Work

Non-malleable secret sharing was introduced by Goyal and Kumar [GK18a]. For any $\tau \leq n$, they showed how to realize τ-threshold access structures, against one-time tampering with either all of the shares independently, or jointly after partitioning the players into two non-overlapping subsets of size at most[4] $\tau - 1$. In a subsequent work [GK18b], the same authors show how to extend the result for independent tampering to the case of arbitrary access structures; additionally, for the case of joint tampering, they provide a new scheme realizing the n-threshold access structure (i.e., an n-out-of-n secret sharing) in a stronger model where the attacker can partition the players into two possibly overlapping subsets of size at most $n - 1$. Srinivasan and Vasudevan [SV18] built the first non-malleable secret sharing schemes for general access structures against independent tampering, with non-zero rate[5] (in fact, even constant rate in case of threshold access structures). Chattopadhyay *et al.* [CL18] construct non-malleable secret sharing for threshold access structures, against affine tampering composed with joint split-state tampering. Lin *et al.* [LCG+19] consider non-malleability against affine tampering in an adaptive setting where the adversary gets to see an unauthorized subset of the shares before launching a single tampering attack.

Badrinarayanan and Srinivasan [BS18] generalize non-malleability to p-time tampering attacks, where p is an a-priori upper bound on the number of tampering queries the adversary can ask. For each attempt, however, the reconstruction set $\mathcal{T}$ must be chosen in advance at the beginning of the experiment. In this model, they show how to realize arbitrary access structures against independent tampering with all of the shares. Aggarwal *et al.* [ADN+18] were the first to consider p-time non-malleability under *non-adaptive* concurrent reconstruction, i.e. the attacker now can specify a different reconstruction set $\mathcal{T}^{(q)}$ during the q-th tampering query, although the sequence of subsets $\mathcal{T}^{(1)}, \ldots, \mathcal{T}^{(p)}$ must be chosen non-adaptively. Kumar, Meka, and Sahai [KMS18] pioneered bounded-leakage-resilient one-time non-malleable secret sharing for general access structures, against independent leakage and tampering with all of the shares.

In the special case of 2-threshold access structures over $n = 2$ parties, the notion of (leakage-resilient) non-malleable secret sharing collapses to that of split-state (leakage-resilient) non-

[3] The only restriction is that no subset in the partition can contain an authorized set of players, otherwise trivial attacks are possible.

[4] An additional (artificial) requirement is that the size of the two subsets must be different in order for their technique to work.

[5] The rate refers to the asymptotic ratio between the maximal length of a share and that of the message.

malleable codes [DPW10, LL12, DKO13, ADL14, CG14, FMNV14, ADKO15b, ADKO15a, AAG+16, Li17, FNSV18, OPVV18, CFV19].

1.3 Organization

In §2 we recall the definition of secret sharing schemes for general access structure; all our constructions rely on standard cryptographic primitives, which we recall in §A (together with some basic notation). The new model of continuous tampering under selective partitioning is presented in §3.

Our main constructions appear in §4 (for joint tampering in the CRS model) and §5–§6 (for independent tampering in the plain model), respectively; there, we also explain how to instantiate these constructions with concrete building blocks, thus establishing Thm. 1 and Thm. 2. Finally, in §7, we conclude the paper with a list of open problems and interesting directions for further research.

2 Secret Sharing Schemes

An n-party secret sharing scheme Σ in the common reference string (CRS) model consists of polynomial-time algorithms $(\mathsf{Init}, \mathsf{Share}, \mathsf{Rec})$ specified as follows: (i) The randomized initialization algorithm Init takes as input the security parameter 1^λ, and outputs a CRS $\omega \in \{0,1\}^*$; (ii) The randomized sharing algorithm Share takes as input a CRS $\omega \in \{0,1\}^*$ and a message $m \in \mathcal{M}$, and outputs n shares $s_1, \ldots, s_n$ where each $s_i \in \mathcal{S}_i$; (iii) The deterministic algorithm Rec takes as input a CRS $\omega \in \{0,1\}^*$ and a certain number of candidate shares, and outputs a value in $\mathcal{M} \cup \{\bot\}$. Given $s = (s_1, \ldots, s_n)$ and a subset $\mathcal{I} \subseteq [n]$, we often write $s_\mathcal{I}$ to denote the shares $(s_i)_{i \in \mathcal{I}}$.

The subset of parties allowed to reconstruct the secrets by pulling their shares together form the so-called access structure.

Definition 1 (Access structure)**.** We say $\mathcal{A}$ is an access structure for n parties if $\mathcal{A}$ is a monotone class of subsets of $[n]$, i.e., if $\mathcal{I}_1 \in \mathcal{A}$ and $\mathcal{I}_1 \subseteq \mathcal{I}_2$, then $\mathcal{I}_2 \in \mathcal{A}$. We call sets $\mathcal{I} \in \mathcal{A}$ authorized or qualified, and unauthorized or unqualified otherwise.

Intuitively, a secure secret sharing scheme must be such that all qualified subsets of players can efficiently reconstruct the secret, whereas all unqualified subset have no information (possibly in a computational sense) about the secret.

Definition 2 (Secret sharing scheme)**.** Let $n \in \mathbb{N}$, and $\mathcal{A}$ be an access structure for n parties. We say that $\Sigma = (\mathsf{Init}, \mathsf{Share}, \mathsf{Rec})$ is a secret sharing scheme realizing access structure $\mathcal{A}$ in the CRS model, with message space $\mathcal{M}$ and share space $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$, if it is an n-party secret sharing in the CRS model with the following properties.

(i) **Correctness:** For all $\lambda \in \mathbb{N}$, all $\omega \in \mathsf{Init}(1^\lambda)$, all messages $m \in \mathcal{M}$, and for all subsets $\mathcal{I} \in \mathcal{A}$, we have that $\mathsf{Rec}(\omega, (\mathsf{Share}(\omega, m))_\mathcal{I}) = m$, with overwhelming probability over the randomness of the sharing algorithm.

(ii) **Privacy:** For all PPT adversaries $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$, we have

$$\{\mathbf{Privacy}_{\Sigma,\mathsf{A}}(\lambda, 0)\}_{\lambda \in \mathbb{N}} \approx_c \{\mathbf{Privacy}_{\Sigma,\mathsf{A}}(\lambda, 1)\}_{\lambda \in \mathbb{N}},$$

where the experiment $\mathbf{Privacy}_{\Sigma,\mathsf{A}}(\lambda, b)$ is defined by

$$\mathbf{Privacy}_{\Sigma,\mathsf{A}}(\lambda, b) := \left\{ \begin{array}{c} \omega \leftarrow\!\!\$\ \mathsf{Init}(1^\lambda); (m_0, m_1, \mathcal{U} \notin \mathcal{A}, \alpha_1) \leftarrow\!\!\$\ \mathsf{A}_1(\omega) \\ s \leftarrow\!\!\$\ \mathsf{Share}(\omega, m_b); b' \leftarrow\!\!\$\ \mathsf{A}_2(\alpha_1, s_\mathcal{U}) \end{array} \right\}.$$

If the above ensembles are statistically close (resp. identically distributed), we speak of *statistical* (resp. *perfect*) privacy.

Moreover, we say that Σ is a secret sharing scheme realizing access structure $\mathcal{A}$ in the plain model, if for all $\lambda \in \mathbb{N}$ algorithm Init simply returns $\omega = 1^\lambda$.

Remark 1. *In the plain model, the above definition of privacy is equivalent to saying that for all pairs of messages $m_0, m_1 \in \mathcal{M}$, and for all unqualified subsets $\mathcal{U} \notin \mathcal{A}$, it holds that*

$$\{(\mathsf{Share}(1^\lambda, m_0))_\mathcal{U}\}_{\lambda\in\mathbb{N}} \approx_c \{(\mathsf{Share}(1^\lambda, m_1))_\mathcal{U}\}_{\lambda\in\mathbb{N}}.$$

3 Continuous Tampering under Selective Partitioning

In this section we define a new notion of non-malleability against joint memory tampering and leakage for secret sharing. Our definition generalizes the one in [FV19] which was tailored to threshold access structures and to individual leakage/tampering from the shares.

Very roughly, in our model the attacker is allowed to partition the set of share holders in k, non-overlapping, subsets covering the entire set $[n]$. This is formalized through the notion of a k-partition.

Definition 3 (k-partition)**.** Let $n, k \in \mathbb{N}$. We call $\mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_k)$ a k-partition of $[n]$ when:

(i) $\bigcup_{i=1}^k \mathcal{B}_i = [n]$;
(ii) $\forall i_1, i_2 \in [k]$, with $i_1 \neq i_2$, we have $\mathcal{B}_{i_1} \cap \mathcal{B}_{i_2} = \emptyset$.

3.1 The Definition

To define non-malleability, we consider an attacker A playing the following game. At the beginning of the experiment, A chooses two messages m_0, m_1 possibly depending on the CRS ω of the underlying secret sharing scheme, and a k-partition $(\mathcal{B}_1, \ldots, \mathcal{B}_k)$ of the set $[n]$. Hence, the adversary interacts with a target secret sharing $s = (s_1, \ldots, s_n)$ of either m_0 or m_1, via the following queries:

- **Leakage queries.** For each $j \in [k]$, the attacker can leak jointly from the shares $s_{\mathcal{B}_j}$. This can be done repeatedly and in an adaptive fashion, the only limitation being that the overall amount of leakage on each subset is at most $\ell \in \mathbb{N}$ bits.
- **Tampering queries.** For each $j \in [k]$, the attacker can tamper jointly the shares $s_{\mathcal{B}_j}$. Each such query yields mauled shares $(\tilde{s}_1, \ldots, \tilde{s}_n)$, for which the adversary is allowed to see the corresponding reconstructed message w.r.t. an arbitrary reconstruction set $\mathcal{T} \in \mathcal{A}$ that is also chosen adversarially. This can be done for at most $p \in \mathbb{N}$ times, and in an adaptive fashion.

The above naturally yields a notion of bounded-leakage and joint-tampering admissible adversary, as defined below. Note that, in order to rule out trivial attacks, we must require that the partition $\mathcal{B}$ chosen by the attacker be such that no subset of the partition is an authorized set for the underlying access structure.

Definition 4 (Joint bounded leakage and selective tampering admissible adversaries)**.** Let $n, k, \ell, p \in \mathbb{N}$, and fix an arbitrary message space $\mathcal{M}$, sharing domain $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ and access structure $\mathcal{A}$ for n parties. We say that a (possibly unbounded) adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ is ℓ-bounded-leakage (k, p)-joint-tampering admissible ((k, ℓ, p)-BLTA for short) if it satisfies the following conditions:

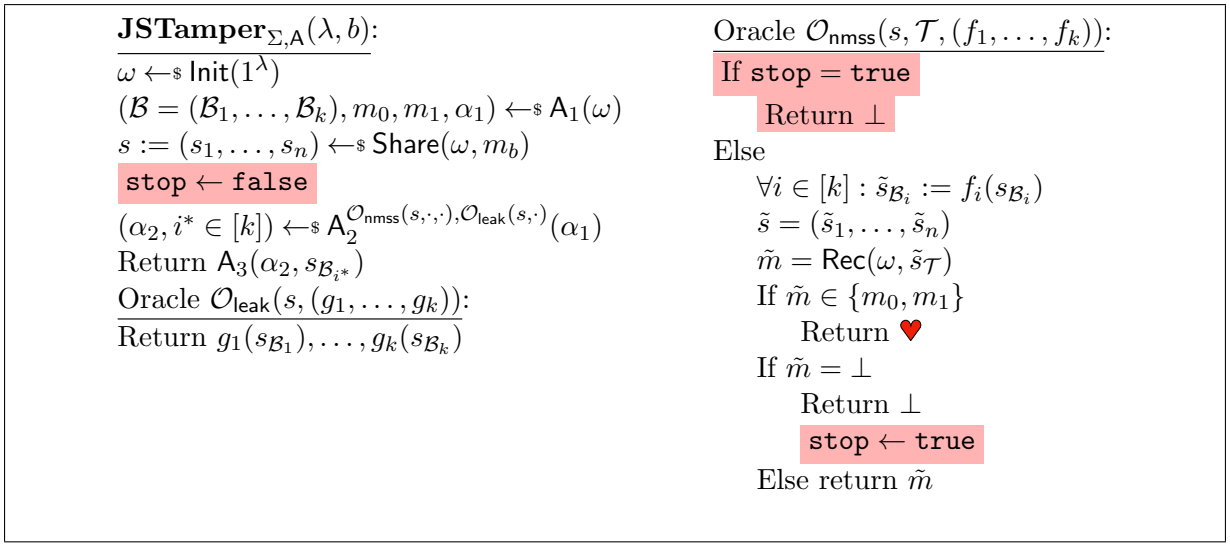

Figure 1: Experiment defining leakage-resilient (continuously) non-malleable secret sharing under adaptive concurrent reconstruction. The instructions boxed in red are considered only for continuous non-malleability, in which case the oracle $\mathcal{O}_{\mathsf{nmss}}$ is implicitly parameterized by the flag `stop`.

(i) A_1 outputs two messages $m_0, m_1 \in \mathcal{M}$ and a k-partition $\mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_k)$ of $[n]$ such that $\forall j \in [k]$ we have $\mathcal{B}_j \notin \mathcal{A}$.

(ii) A_2 outputs a sequence of poly-many leakage queries, chosen adaptively, $(g_1^{(q)}, \ldots, g_k^{(q)})_{q \in \mathtt{poly}(\lambda)}$ such that $\forall j \in [k]$ it holds that $\sum_q |g_j^{(q)}(\cdot)| \leq \ell$, where $g_j^{(q)} : \bigtimes_{i \in \mathcal{B}_j} \mathcal{S}_i \to \{0,1\}^*$.

(iii) A_2 outputs a sequence of p tampering queries, chosen adaptively, $(\mathcal{T}^{(q)}, (f_1^{(q)}, \ldots, f_k^{(q)}))_{q \in [p]}$ such that $\mathcal{T}^{(q)} \in \mathcal{A}$, and $\forall j \in [k]$ it holds that $f_j^{(q)} : \bigtimes_{i \in \mathcal{B}_j} \mathcal{S}_i \to \bigtimes_{i \in \mathcal{B}_j} \mathcal{S}_i$.

Very roughly, leakage-resilient non-malleability states that no admissible adversary as defined above can distinguish whether it is interacting with a secret sharing of m_0 or of m_1. In the definition below, the attacker is further allowed to obtain in full the shares belonging to one of the partitions, at the end of the experiment. This is reminiscent of augmented (leakage-resilient) non-malleability, as considered in [FMNV14, AAG+16, GPR16, CFV19].

Definition 5 (Leakage-resilient non-malleability under selective partitioning)**.** Let $n, k, \ell, p \in \mathbb{N}$ be parameters, and $\mathcal{A}$ be an access structure for n parties. We say that $\Sigma = (\mathsf{Init}, \mathsf{Share}, \mathsf{Rec})$ is an *augmented* ℓ-bounded leakage-resilient p-time non-malleable secret sharing scheme realizing $\mathcal{A}$ under joint k-selective partitioning in the CRS model (resp., in the plain model)—augmented (k, ℓ, p)-BLR-CNMSS for short—if it is an n-party secret sharing scheme realizing $\mathcal{A}$ in the CRS model (resp., in the plain model) as per Def. 2, and additionally for all (k, ℓ, p)-BLTA adversaries $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ we have:

$$\left\{\mathbf{JSTamper}_{\Sigma,\mathsf{A}}(\lambda, 0)\right\}_{\lambda \in \mathbb{N}} \approx_s \left\{\mathbf{JSTamper}_{\Sigma,\mathsf{A}}(\lambda, 1)\right\}_{\lambda \in \mathbb{N}},$$

where, for $b \in \{0,1\}$, experiment $\mathbf{JSTamper}_{\Sigma,\mathsf{A}}(\lambda, b)$ is depicted in Fig. 1.

In case the above definition holds for all $p(\lambda) \in \mathtt{poly}(\lambda)$, but w.r.t. all PPT adversaries A (i.e., $\approx_s$ is replaced with $\approx_c$ in the above equation), we call Σ (augmented, bounded leakage-resilient) continuously non-malleable. As shown by [FV19], already for the simpler case of independent tampering, it is impossible to achieve this notion without assuming self-destruct

(i.e., the oracle $\mathcal{O}_{\mathsf{nmss}}$ must stop answering tampering queries after the first such query yielding an invalid reconstructed message).

It is also well-known that computational security is inherent for obtaining continuously non-malleable secret sharing realizing threshold access structures [BS18]. Unless stated otherwise, when we refer to non-malleable secret sharing in this paper we implicitly assume security holds in the computational setting (both for privacy and non-malleability).

3.2 On Augmented Non-Malleability

When dropping the adversary A_3 from the above definition, we obtain the standard (non-augmented) notion of (leakage-resilient, continuous) non-malleability. The theorem below, however, says that augmented security is essentially for free whenever non-malleability is considered together with leakage resilience.[6] Intuitively, this is because in the reduction we can simply simulate all leakage queries, and then ask a final leakage query which reveals the output guess of an hypothetical distinguisher attacking augmented non-malleability.

Theorem 3. *Let Σ be a $(k, \ell+1, p)$-BLR-CNMSS realizing access structure $\mathcal{A}$ for n parties in the CRS model (resp. plain model). Then, Σ is an* augmented *(k, ℓ, p)-BLR-CNMSS realizing $\mathcal{A}$ in the CRS model (resp. plain model).*

Proof. Without loss of generality, we prove the statement in the CRS model. Let $\mathsf{A}^+ = (\mathsf{A}_1^+, \mathsf{A}_2^+, \mathsf{A}_3^+)$ be a (k, ℓ, p)-BLTA attacker violating Def. 5 for Σ; we construct an adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ breaking the non-augmented variant of Def. 5 for Σ. Attacker A works as follows:

- Upon receiving ω from the challenger, $\mathsf{A}_1(\omega)$ outputs the same tuple $(\mathcal{B}, m_0, m_1)$ as returned by $\mathsf{A}_1^+(\omega)$.
- Upon input a leakage query $(g_1, \ldots, g_k)$ from A_2^+, forward the same query to the target leakage oracle and return the answer to A_2^+.
- Upon input a tampering query $(\mathcal{T}, (f_1, \ldots, f_k))$ from A_2^+, forward the same query to the target tampering oracle and return the answer to A_2^+.
- Let (α_2, i^*) be the final output of A_2^+. Define the leakage function $\hat{g}_{i^*}^{\alpha_2, \mathsf{A}_3^+}$ which hard-wires α_2 and a description of A_3^+, takes as input the shares $s_{\mathcal{B}_{i^*}}$, and returns the decision bit $b' \leftarrow\!\!\$\ \mathsf{A}_3^+(\alpha_2, s_{\mathcal{B}_{i^*}})$.
- Forward $(\varepsilon, \ldots, \hat{g}_{i^*}^{\alpha_2, \mathsf{A}_3^+}, \varepsilon, \ldots, \varepsilon)$ to the target leakage oracle, obtaining a bit b' which is then sent to the challenger.

The statement follows by observing that A's simulation to A^+'s leakage/tampering queries is perfect, and moreover A_2 leaks a total of at most $\ell+1$ bits from each partition of the shares. □

3.3 Related Notions

We finally argue that known definitions from the literature can be cast by either restricting, or slightly tweaking, Def. 5.

Individual leakage and tampering. The definition below restricts the adversary to leak/-tamper from/with each of the shares individually; this is sometimes known as local or independent leakage/tampering. The condition on leakage admissibility, though, is more general, in that the attacker can leak an arbitrary amount of information as long as the total leakage reduces the uncertainty on each share (conditioned on the other shares) by at most ℓ bits.

[6] While we state the theorem for the case of bounded leakage, an identical statement holds in the noisy-leakage setting.

Definition 6 (Independent noisy leakage and tampering admissible adversaries)**.** Let $n, \ell, p \in \mathbb{N}$, and fix an arbitrary message space $\mathcal{M}$, sharing domain $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ and access structure $\mathcal{A}$ for n parties. We say that a (possibly unbounded) adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ is ℓ-noisy-leakage independent-tampering admissible ((n, ℓ, p)-NLTA for short) if it satisfies the following conditions:

(i) A_1 outputs two messages $m_0, m_1 \in \mathcal{M}$ and the partition $\mathcal{B} = (\{1\}, \ldots, \{n\})$.

(ii) A_2 outputs a sequence of poly-many leakage queries (chosen adaptively) $(g_1^{(q)}, \ldots, g_n^{(q)})_{q \in \mathtt{poly}(\lambda)}$ such that $\forall i \in [n]$ we have $g_i^{(q)} : \mathcal{S}_i \to \{0,1\}^*$, and $\forall m \in \mathcal{M}$ it holds that:

$$\widetilde{\mathbb{H}}_\infty \left(\mathbf{S}_i | (\mathbf{S}_j)_{j \neq i}, g_i^{(1)}(\mathbf{S}_i), \cdots, g_i^{(p)}(\mathbf{S}_i) \right) \geq \widetilde{\mathbb{H}}_\infty (\mathbf{S}_i | (\mathbf{S}_j)_{j \neq i}) - \ell,$$

where $(\mathbf{S}_1, \ldots, \mathbf{S}_n)$ is the random variable corresponding to $\mathsf{Share}(\mathsf{Init}(1^\lambda), m)$.

(iii) A_2 outputs a sequence of tampering queries (chosen adaptively) $(\mathcal{T}^{(q)}, (f_1^{(q)}, \ldots, f_n^{(q)}))_{q \in [p]}$ such that $\mathcal{T}^{(q)} \in \mathcal{A}$, and $\forall i \in [n]$ it holds that $f_i^{(q)} : \mathcal{S}_i \to \mathcal{S}_i$.

When restricting Def. 5 to all PPT $(n, \ell, \mathtt{poly}(\lambda))$-NLTA adversaries, we obtain the notion of (augmented) ℓ-noisy leakage-resilient continuously non-malleable secret sharing against individual leakage and tampering (with adaptive concurrent reconstructions) [FV19]. Finally, if we consider $n = 2$ and the threshold access structure with reconstruction parameter $\varrho = 2$ (i.e., both shares are required in order to reconstruct the message), we immediately obtain noisy leakage-resilient continuously non-malleable codes in the split-state model [FMNV14, OPVV18]. In what follows, we write $\mathbf{Tamper}(\lambda, b)$ to denote the random variable in the security experiment of Def. 5 with an (n, ℓ, p)-NLTA adversary.

Leakage-resilient secret sharing. Further, when no tampering is allowed (i.e., $p = 0$), we obtain the notion of leakage-resilient secret sharing [DDV10, KMS18, SV18, ADN+18, NS19] as a special case. In particular, we write $\mathbf{JSLeak}(\lambda, b)$ to denote the random variable in the security experiment of Def. 5 with a $(k, \ell, 0)$-BLTA adversary, and $\mathbf{Leak}(\lambda, b)$ to denote the random variable in the security experiment of Def. 5 with an $(n, \ell, 0)$-NLTA adversary.

Recall that, by Theorem 3, the augmented variant is without loss of generality as long as leakage resilience holds for $\ell \geq 2$.

4 Construction in the CRS Model

4.1 Description of the Scheme

We show how to obtain leakage-resilient continuously non-malleable secret sharing for arbitrary access structures in the CRS model, with security against joint selective partitioning. Our construction combines a commitment scheme $(\mathsf{Gen}, \mathsf{Com})$ (cf. §A.2), a non-interactive proof system $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Ver})$ of knowledge of a committed value that supports labels (cf. §A.3), and an auxiliary n-party secret sharing scheme $\Sigma = (\mathsf{Share}, \mathsf{Rec})$, as depicted in Fig. 2.

The main idea behind the scheme is as follows. The CRS includes the CRS ω for the proof system and the public key pk for the commitment scheme. Given a message $m \in \mathcal{M}$, the sharing procedure first shares m using Share, obtaining shares $(s_1, \ldots, s_n)$. Then, it commits to the i-th share s_i along with the position i using randomness r_i, and finally generates $n-1$ proofs $(\pi_j^i)_{j \neq i}$ for the statement c_i using each time the value $c_j = \mathsf{Com}(pk, j || s_j; r_j)$ as label. The final share of player i consists of s_i, along with the randomness r_i used to obtain c_i and all the values $(c_j)_{j \neq i}$ and $(\pi_j^i)_{j \neq i}$. The reconstruction procedure, given a set of shares $s_{\mathcal{I}}^*$, first checks that

for each $i \in \mathcal{I}$ the commits $(c_j)_{j \neq i}$ contained in each share are all equal, and moreover each c_i is indeed obtained by committing $i||s_i$ with the randomness r_i; further, it checks that all the proofs verify correctly w.r.t. the corresponding statement and label. If any of the above checks fails, the algorithm returns $\perp$ and otherwise it outputs the same as $\mathsf{Rec}(s_{\mathcal{I}})$.

Intuitively, our scheme can be seen as a generalization of the original construction of continuously non-malleable codes in the split-state model from [FMNV14]. In particular, when $n = 2$, the two constructions are identical except for two differences: (i) We commit to each share, whereas [FMNV14] uses a collision-resistant hash function; (ii) We include the position of each share in the commitment. Roughly speaking, the first modification is necessary in order to prove privacy (as hash functions do not necessarily hide their inputs). The second modification is needed in order to avoid that an attacker can permute the shares within one of the partitions, which was not possible in the setting of independent tampering. We establish the following result.

Theorem 4. *Let $n, k \in \mathbb{N}$, and $\mathcal{A}$ be any access structure for n parties. Assume that:*

(i) *Σ is an n-party augmented ℓ-bounded leakage-resilient secret sharing scheme realizing access structure $\mathcal{A}$ under k-selective joint leakage in the plain model;*

(ii) *$(\mathsf{Gen}, \mathsf{Com})$ is a statistically hiding and computationally binding commitment scheme with commitment length $\gamma = O(\lambda)$;*

(iii) *$(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Ver})$ is a true-simulation extractable non-interactive zero-knowledge argument system for the language $\mathcal{L}^{pk}_{\mathsf{com}} = \{c \in \{0,1\}^\gamma : \exists i \in [n], s \in \mathcal{S}_i, r \in \mathcal{R} \text{ s.t. } \mathsf{Com}(pk, i||s; r) = c\}$.*

Then, the secret sharing scheme Σ^* described in Fig. 2 is an n-party augmented ℓ^*-bounded

Let $\Sigma = (\mathsf{Share}, \mathsf{Rec})$ be an auxiliary secret sharing scheme realizing access structure $\mathcal{A}$, with message space $\mathcal{M}$ and share space $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$. Let $(\mathsf{Gen}, \mathsf{Com})$ be a commitment scheme with domain $\{0,1\}^*$, and $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Ver})$ be a non-interactive argument system for the language $\mathcal{L}^{pk}_{\mathsf{com}} = \{c \in \{0,1\}^\gamma : \exists i \in [n], s \in \mathcal{S}_i, r \in \mathcal{R} \text{ s.t. } \mathsf{Com}(pk, i||s; r) = c\}$ that supports labels in $\{0,1\}^\gamma$. Define the following secret sharing scheme $\Sigma^* = (\mathsf{Init}^*, \mathsf{Share}^*, \mathsf{Rec}^*)$ in the CRS model.

Initialization algorithm Init^*: Sample $\omega \leftarrow\!\!\$\ \mathsf{CRSGen}(1^\lambda)$ and $pk \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$, and return $\omega^* = (\omega, pk)$.

Sharing algorithm Share^*: Upon input $\omega^* = (\omega, pk)$ and a value $m \in \mathcal{M}$, compute $(s_1, \ldots, s_n) \leftarrow\!\!\$\ \mathsf{Share}(m)$. For each $i \in [n]$, generate $r_i \leftarrow\!\!\$\ \mathcal{R}$ and define $c_i = \mathsf{Com}(pk, i||s_i; r_i)$. For each $i, j \in [n]$ such that $i \neq j$, define $\pi_i^j \leftarrow\!\!\$\ \mathsf{Prove}(\omega, c_j, (c_i, i||s_i, r_i))$. Return the shares $s^* = (s_1^*, \ldots, s_n^*)$, where for each $i \in [n]$ we set $s_i^* = (s_i, r_i, (c_j)_{j \neq i}, (\pi_j^i)_{j \neq i})$.

Reconstruction algorithm Rec^*: Upon input $\omega^* = (\omega, pk)$ and shares $(s_i^*)_{i \in \mathcal{I}}$ parse $s_i^* = (s_i, r_i, (c_j^i)_{j \neq i}, (\pi_j^i)_{j \neq i})$ for each $i \in \mathcal{I}$. Hence, proceed as follows:

(a) If $\exists i_1, i_2 \in \mathcal{I}$ and $j \in [n]$ such that $c_j^{i_1} \neq c_j^{i_2}$, output $\perp$; else let the input shares be $s_i^* = (s_i, r_i, (c_j)_{j \neq i}, (\pi_j^i)_{j \neq i})$ for each $i \in \mathcal{I}$.

(b) If $\exists i \in \mathcal{I}$ such that $\mathsf{Com}(pk, i||s_i; r_i) \neq c_i$, output $\perp$.

(c) If $\exists i, j \in \mathcal{I}$ such that $i \neq j$ and $\mathsf{Ver}(\omega, c_j, (c_i, \pi_i^j)) = 0$, output $\perp$.

(d) Else, output $\mathsf{Rec}((s_i)_{i \in \mathcal{I}})$.

Figure 2: Leakage-resilient continuously non-malleable secret sharing for arbitrary access structures against selective joint leakage/tampering, in the CRS model.

leakage-resilient continuously non-malleable secret sharing scheme realizing access structure $\mathcal{A}$ under k-selective partitioning with concurrent reconstruction in the CRS model, as long as $\ell = 2\ell^* + n\gamma + O(\lambda \log \lambda)$.

4.2 Proof Overview

Before coming to the proof, we discuss the main intuition behind privacy and continuous non-malleability.

Privacy. In order to show privacy, we need to prove that no PPT attacker A can distinguish between $\mathbf{Privacy}_{\Sigma^*,\mathsf{A}}(1^\lambda, 0)$ and $\mathbf{Privacy}_{\Sigma^*,\mathsf{A}}(1^\lambda, 1)$ (cf. Def. 2). Recall that in this experiment, after seeing the CRS, the attacker can select an unauthorized subset $\mathcal{U}$ along with messages (m_0, m_1), after which it is given the shares $s^*_{\mathcal{U}}$ (either corresponding to message m_0 or to m_1).

Consider a game hop in which we generate the CRS together with a simulation trapdoor, and we replace the proofs π_i^j in the target secret sharing s^* with simulated proofs. A reduction to adaptive multi-theorem zero knowledge of the NIZK (cf. Def. 13 in §A.3) shows that no attacker can distinguish this modified experiment from the original game. Next, we replace the values c_i with commitments to dummy values. A simple hybrid argument, relying on the statistical hiding property of the commitment (cf. Def. 12 in §A.2), shows that this experiment is statistically close to the previous one.

Finally, we reduce to the privacy of Σ. In particular, note that the reduction can locally sample the CRS and compute the values c_i (with exactly the same distribution as in the final experiment).

Non-malleability. The goal is to show that no PPT attacker A can distinguish the experiments $\mathbf{JSTamper}_{\Sigma^*,\mathsf{A}}(\lambda, 0)$ and $\mathbf{JSTamper}_{\Sigma^*,\mathsf{A}}(\lambda, 1)$ (cf. Def. 5). Recall that A, after seeing the CRS, can selectively partition the set $[n]$ into k blocks $\mathcal{B}_1, \ldots, \mathcal{B}_k$, and then jointly leak from and tamper with the shares within each block. The proof proceeds via a hybrid argument, as outlined below.

$\mathbf{Hyb}^1_{\Sigma^*,\mathsf{A}}(\lambda, b)$: In the first hybrid, we modify the distribution of the target secret sharing $s^* = (s_1, \ldots, s_n)$. In particular, we first let the NIZK simulator S_0 program the CRS ω yielding simulation trapdoor ζ and extraction trapdoor ξ, and then we compute each of the proofs π_i^j by running the NIZK simulator S_1 upon input ζ, statement c_i and label c_j.

$\mathbf{Hyb}^2_{\Sigma^*,\mathsf{A}}(\lambda, b)$: In the second hybrid, we modify the way tampering queries are answered. In particular, let $(\mathcal{T}, (f_1, \ldots, f_k))$ be a generic tampering query, and $\tilde{s}^* = (\tilde{s}^*_1, \ldots, \tilde{s}^*_n)$ be the mauled secret sharing after tampering jointly with the shares; here, each $\tilde{s}^*_i$ can be parsed[7] as $\tilde{s}^*_i = (\tilde{s}_i, (\tilde{c}_j)_{j \neq i}, (\tilde{\pi}^i_j)_{j \neq i})$.

The tampering oracle proceeds as follows: (a) In case the set of mauled commitments $(\tilde{c})_{i \in [n]}$ is equal to the set of initial commitments $(c_i)_{i \in [n]}$ considered in their order, the answer is ♥; (b) In case there exist at least two distinct indices i_1, i_2 such that $\tilde{c}_{i_1} = c_{i_2}$, abort; (c) Otherwise, there exists at least one commitment $\tilde{c}_{i^*}$ that is different from all the initial commitments $(c_i)_{i \in [n]}$. In the latter case, we can use the extraction trapdoor and the knowledge extractor of the NIZK to extract[8] from each set $\tilde{s}^*_{\mathcal{B}_i}$ of tampered shares

[7] As in the original experiment, the tampering oracle first checks that the commitments in each of the shares are all equal, that the NIZK proofs are accepting, and that each $\tilde{c}_j$ is indeed a valid commitment of $j||\tilde{s}_j$; in case any of these checks fails, the oracle self-destructs.

[8] More in details, the extraction procedure extracts the missing shares from the proofs whose statement or label corresponds to one of the mauled commitments; since we might extract one of the shares from more than one proof, the procedure further checks that those the extracted shares are all consistent. Cf. Fig. 4 on page 18

the missing shares $\tilde{s}^*_{\mathcal{T}\setminus\mathcal{B}_i}$, which yields a candidate answer for the attacker's tampering query: In case the answers are all consistent for each of the k subsets in the partition, the oracle returns this value, and otherwise it outputs $\perp$ and self-destructs.

As a first step, we argue that $\mathbf{JSTamper}_{\Sigma^*,\mathsf{A}}(\lambda, b)$ and $\mathbf{Hyb}^1_{\Sigma^*,\mathsf{A}}(\lambda, b)$ are computationally close. This follows readily from adaptive multi-theorem zero knowledge of the NIZK (cf. Def. 13 in §A.3), as the only difference between the two experiments is the fact that in the latter the proofs π^i_j are simulated. As a second step, we prove that $\mathbf{Hyb}^1_{\Sigma^*,\mathsf{A}}(\lambda, b)$ and $\mathbf{Hyb}^2_{\Sigma^*,\mathsf{A}}(\lambda, b)$ are also computationally indistinguishable. More in details, we show how to bound the probability that the output of the tampering oracle in the two experiments differs in the above described cases (a), (b) and (c). In case (a), we rely on computational binding (cf. Def. 11 in §A.2) of $(\mathsf{Gen}, \mathsf{Com})$ to argue that if the mauled shares are equal to the original, the underlying sharing is also unchanged. In case (b), we rely again on computational binding of $(\mathsf{Gen}, \mathsf{Com})$ using the fact that each value c_i is obtained by committing to both the share s_i and the position $i \in [n]$, so that permuting the commitments within a given subset of the partition $\mathcal{B}$ yields an invalid output. In case (c), we rely on simulation extractability (cf. Def. 14 in §A.3) to argue that whenever the outcome of a tampering query in the first hybrid is a valid message $\tilde{m} \in \mathcal{M}$, this value must be the same that can be extracted from the proofs associated to the commitments that have been modified; here is where we exploit the fact that the NIZK supports labels, since the extractor works as long as either the statement or the label mauled by the adversary are fresh.

Next, we show that no PPT attacker A can distinguish between $\mathbf{Hyb}^2_{\Sigma^*,\mathsf{A}}(\lambda, 0)$ and $\mathbf{Hyb}^2_{\Sigma^*,\mathsf{A}}(\lambda, 1)$ with better than negligible probability. To this end, we build a reduction $\hat{\mathsf{A}}$ to leakage resilience of the underlying secret sharing Σ. In order to keep the exposition simple, let us first assume that A is not allowed to ask leakage queries. Very roughly, the reduction works as follows.

- **(Simulate the CRS.)** At the beginning, $\hat{\mathsf{A}}$ receives the challenge CRS ω, samples the public key pk, and runs A upon (ω, pk) with fresh randomness r. Upon receiving $(\mathcal{B}, m_0, m_1)$, then $\hat{\mathsf{A}}$ forwards the same tuple to the challenger.
- **(Learn the self-destruct index.)** Note that in the last hybrid, the tampering oracle computes the answers to A's tampering queries by computing a candidate answer for each set $\mathcal{B}_i$ of the partition, and then this value is returned if the candidate values are all consistent (and otherwise a self-destruct is triggered). Since $\hat{\mathsf{A}}$ outputs the same partition $\mathcal{B}$ chosen by A, it can compute the different candidates by running A with hard-wired randomness r inside the leakage oracle,[9] and then use a pairwise independent hash function to determine using a binary search the first index where the candidates differ. By pairwise independence, this strategy yields the index of the query p^* in which A provokes a self-destruct with overwhelming probability, and by leaking at most $O(\lambda \log \lambda)$ bits from each subset.
- **(Play the game.)** Once the self-destruct index p^* is known, the idea is that now $\hat{\mathsf{A}}$ can restart A outside the leakage oracle, with the same randomness r, and answer to the first $p^* - 1$ tampering queries given the shares $\tilde{s}_{\mathcal{B}_i}$ within one of the partitions, after which the answer to all remaining tampering queries is set to be $\perp$, so that $\hat{\mathsf{A}}$ can output the same guess of A and keep its advantage. Here, we rely on the fact that Σ is *augmented* leakage resilient, i.e. leakage resilience still holds even if $\hat{\mathsf{A}}$ is given the shares belonging to one of

for a precise description.

[9] To be more precise, this also requires to leak the initial commitments to each share, to simulate the NIZK proofs as done in the last hybrid, and to hard-wire those values in each leakage query. However, the latter can be done easily by the reduction.

the partitions after the leakage is done. By Thm. 3 (cf. §3.2), this assumption is without loss of generality.

Finally, we show how to remove the simplifying assumption that A cannot leak from the shares. The difficulty when considering leakage is that we cannot run anymore the entire experiment with A inside the leakage oracle, as the answer to A's leakage queries depends on the shares outside a given partition. However, note that in this case we can stop the execution whenever A asks a leakage query and inform the reduction to leak from the other shares whatever information is needed to continue the execution of each copy of A inside the leakage oracle.

This allows to obtain the answers to all leakage queries of A up to a self-destruct occurs. In order to obtain the answers to the remaining queries, we must re-run A inside the leakage oracle and adjust the simulation consistently with the self-destruct index being p^*. In the worst case, this requires $2\ell^*$ bits of leakage from the shares, yielding the final bound of $2\ell^* + n\gamma + O(\lambda \log \lambda)$. At the end, the reduction knows the answer to all leakage queries of A with hard-wired randomness r, and can thus win the game with the challenger in the same way as explained above.

4.3 Security Analysis

Correctness w.r.t. access structure $\mathcal{A}$ is immediate. We thus need to prove that Σ^* satisfies privacy and continuous non-malleability.

Privacy. We need to show that

$$\{\mathbf{Privacy}_{\Sigma^*,\mathsf{A}}(\lambda,0)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{Privacy}_{\Sigma^*,\mathsf{A}}(\lambda,1)\}_{\lambda\in\mathbb{N}}.$$

Below, we describe the experiment $\mathbf{Privacy}_{\Sigma^*,\mathsf{A}}(\lambda,b)$ after expanding the specification of Init^* and Share^*.

- Generate $\omega^* \leftarrow\!\!\$\ \mathsf{Init}^*(1^\lambda)$, where Init^* is computed as follows:
  - upon receiving 1^λ, compute $\omega \leftarrow\!\!\$\ \mathsf{CRSGen}(1^\lambda)$, $pk \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$;
  - output $\omega^* = (\omega, pk)$.
- Run $(\alpha_1, m_0, m_1, \mathcal{U} \notin \mathcal{A}) \leftarrow\!\!\$\ \mathsf{A}_1(\omega^*)$.
- Compute $(s_1^*, \ldots, s_n^*) \leftarrow\!\!\$\ \mathsf{Share}^*(\omega^*, m_b)$, where Share^* is defined as follows:
  - upon receiving (ω^*, m), parse $\omega^* = (\omega, pk)$ and compute $(s_1, \ldots, s_n) \leftarrow\!\!\$\ \mathsf{Share}(m)$;
  - for all $i \in [n]$, generate $r_i \leftarrow\!\!\$\ \mathcal{R}$ and compute $c_i = \mathsf{Com}(pk, i||s_i; r_i)$ and, for all $j \in [n]$ such that $j \neq i$, compute $\pi_i^j = \mathsf{Prove}(\omega, c_j, (c_i, i||s_i, r_i))$;
  - for all $i \in [n]$, write $s_i^* = (s_i, r_i, (c_j)_{j\neq i}, (\pi_j^i)_{j\neq i})$;
  - output $(s_1^*, \ldots, s_n^*)$.
- Run $b' \leftarrow\!\!\$\ \mathsf{A}_2(\alpha_1, (s_u^*)_{u\in\mathcal{U}})$.

Let $\mathsf{S} = (\mathsf{S}_0, \mathsf{S}_1)$ be the simulator for the adaptive multi-theorem zero-knowledge property of Π, and consider the hybrid experiment $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,b)$ in which the algorithms Init^* and Share^* are replaced by Init' and Share', defined below.

- Algorithm Init':
  - upon receiving 1^λ, compute $(\omega, \zeta) \leftarrow\!\!\$\ \mathsf{S}_0(1^\lambda)$, $pk \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$;
  - output $\omega^* = (\omega, pk)$.
- Algorithm Share' (with hard-wired ζ):

– upon receiving (ω^*, m), parse $\omega^* = (\omega, pk)$ and compute $(s_1, \ldots, s_n) \leftarrow\!\!\$\ \mathsf{Share}(m)$;
– for all $i \in [n]$, generate $r_i \leftarrow\!\!\$\ \mathcal{R}$, compute $c_i = \mathsf{Com}(pk, i||s_i; r_i)$, and for all $j \in [n]$ with $j \neq i$ let $\pi_i^j \leftarrow\!\!\$\ \mathsf{S}_1(\zeta, c_j, c_i)$;
– for all $i \in [n]$, write $s_i^* = (s_i, r_i, (c_j)_{j\neq i}, (\pi_j^i)_{j\neq i})$;
– output $(s_1^*, \ldots, s_n^*)$.

Finally, consider the experiment $\mathbf{Hyb}^t_{\Sigma^*,\mathsf{A}}(\lambda, b)$, defined below.

- Generate $(\omega^*, \zeta) \leftarrow\!\!\$\ \mathsf{Init}'(1^\lambda)$.
- Run $(\alpha_1, m_0, m_1, \mathcal{U} \notin \mathcal{A}) \leftarrow\!\!\$\ \mathsf{A}_1(\omega^*)$.
- Compute $(s_1, \ldots, s_n) \leftarrow\!\!\$\ \mathsf{Share}(m_b)$.
- Generate $\hat{m} \leftarrow\!\!\$\ \mathcal{M}$ and compute $(\hat{s}_1, \ldots, \hat{s}_n) \leftarrow\!\!\$\ \mathsf{Share}(\hat{m})$.
- For all $i \in [n]$, generate $r_i \leftarrow\!\!\$\ \mathcal{R}$ and compute c_i as follows. If $i \in \mathcal{U}$ or $i > t$, compute $c_i = \mathsf{Com}(pk, i||s_i, r_i)$; else, compute $c_i = \mathsf{Com}(pk, i||\hat{s}_i, r_i)$.
- For all $i, j \in [n]$ such that $i \neq j$, compute $\pi_i^j \leftarrow\!\!\$\ \mathsf{S}_1(\zeta, c_j, c_i)$.
- For all $i \in \mathcal{U}$, write $s_i^* = (s_i, r_i, (c_j)_{j\neq i}, (\pi_j^i)_{j\neq i})$.
- Run $b' \leftarrow\!\!\$\ \mathsf{A}_2(\alpha_1, (s_u^*)_{u\in\mathcal{U}})$.

Note that, for $t = 0$, the condition $i > t$ is true for all $i \in [n]$, thus all the commitments are computed using the shares from the message m_b and the experiments $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda, b)$ and $\mathbf{Hyb}^0_{\Sigma^*,\mathsf{A}}(\lambda, b)$ are equivalent.

We now prove that the original experiment is computationally close to the first hybrid experiment and, for fixed $b \in \{0, 1\}$, all the hybrid experiments defined above are statistically close. Finally we prove that, for $t = n$, the experiments with $b = 0$ and $b = 1$ are computationally close.

Lemma 1. $\forall b \in \{0, 1\}$: $\{\mathbf{Privacy}_{\Sigma^*,\mathsf{A}}(\lambda, b)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda, b)\}_{\lambda\in\mathbb{N}}$.

Proof. The proof is down to the adaptive multi-theorem zero-knowledge property of the underlying non-interactive zero-knowledge argument system $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Ver})$. By contradiction, assume that there exists a PPT attacker $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ that, for any fixed $b \in \{0, 1\}$, can distinguish between $\{\mathbf{Privacy}_{\Sigma^*,\mathsf{A}}(\lambda, b)\}_{\lambda\in\mathbb{N}}$ and $\{\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda, b)\}_{\lambda\in\mathbb{N}}$ with noticeable advantage. Consider the following PPT distinguisher D attacking the zero-knowledge property of Π.

- Upon receiving a CRS ω generated either by $\mathsf{CRSGen}(1^\lambda)$ or by the simulator $\mathsf{S}_0(1^\lambda)$ (depending on the experiment that D is currently playing), generate $pk \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$ and write $\omega^* = (\omega, pk)$.
- Run $(\alpha_1, m_0, m_1, \mathcal{U}) \leftarrow\!\!\$\ \mathsf{A}_1(\omega^*)$ and compute $s = (s_1, \ldots, s_n) \leftarrow\!\!\$\ \mathsf{Share}(m_b)$.
- For each $i \in [n]$, generate $r_i \leftarrow\!\!\$\ \mathcal{R}$ and compute $c_i = \mathsf{Com}(pk, i||s_i; r_i)$.
- For each $i, j \in [n]$ such that $i \neq j$, query the challenger with $(c_j, (c_i, i||s_i, r_i))$, obtaining either $\pi_i^j \leftarrow\!\!\$\ \mathsf{Prove}(\omega, c_j, (c_i, i||s_i, r_i))$ or $\pi_i^j \leftarrow\!\!\$\ \mathsf{S}_1(\zeta, c_j, c_i)$ (depending on the experiment that D is currently playing).
- For each $i \in [n]$, write $s_i^* = (s_i, r_i, (c_j)_{j\neq i}, (\pi_j^i)_{j\neq i})$.
- Write $s^* = (s_1^*, \ldots, s_n^*)$.
- Run $b' \leftarrow\!\!\$\ \mathsf{A}_2(\alpha_1, (s_i^*)_{i\in\mathcal{U}})$.
- Output b'.

For the analysis, note that the simulation done by D is perfect. Thus, D distinguishes with noticeable advantage. The lemma follows. □

Lemma 2. $\forall t \in [n]$: $\{\mathbf{Hyb}^{t-1}_{\Sigma^*,\mathsf{A}}(\lambda, b)\}_{\lambda\in\mathbb{N}} \approx_s \{\mathbf{Hyb}^t_{\Sigma^*,\mathsf{A}}(\lambda, b)\}_{\lambda\in\mathbb{N}}$.

Proof. The proof is down to the statistical hiding property of the underlying commitment scheme. Fix $t \in [n]$. By contradiction, assume that there exists a computationally unbounded adversary A that can distinguish between the two experiments with noticeable advantage. Consider the following unbounded adversary $\hat{\mathsf{A}}$ attacking the statistical hiding property of $(\mathsf{Gen}, \mathsf{Com})$.

- Upon receiving $pk \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$, generate $(\omega, \zeta) \leftarrow\!\!\$\ \mathsf{S}_0(1^\lambda)$ and write $\omega^* = (\omega, pk)$.
- Run $(\alpha_1, m_0, m_1, \mathcal{U} \notin \mathcal{A}) \leftarrow\!\!\$\ \mathsf{A}_1(\omega^*)$.
- Generate $\hat{m} \leftarrow\!\!\$\ \mathcal{M}$ and compute $(\hat{s}_1, \ldots, \hat{s}_n) \leftarrow\!\!\$\ \mathsf{Share}(\hat{m})$.
- Send the pair $(t||s_t, t||\hat{s}_t)$ to the challenger, receiving a commitment c_t.
- For all $i \in [n]$ such that $i \neq t$, generate $r_i \leftarrow\!\!\$\ \mathcal{R}$ and compute c_i as follows. If $i \in \mathcal{U}$ or $i > t$, compute $c_i = \mathsf{Com}(pk, i||s_i; r_i)$; else, compute $c_i = \mathsf{Com}(pk, i||\hat{s}_i; r_i)$.
- For all $i, j \in [n]$ such that $i \neq j$, compute $\pi_i^j \leftarrow\!\!\$\ \mathsf{S}_1(\zeta, c_j, c_i)$.
- For all $i \in \mathcal{U}$, write $s_i^* = (s_i, r_i, (c_j)_{j \neq i}, (\pi_j^i)_{j \neq i})$.
- Run $b' \leftarrow\!\!\$\ \mathsf{A}_2(\alpha_1, (s_u^*)_{u \in \mathcal{U}})$.
- Output b'.

Note that the only difference between the two experiments is how the commitment c_t is computed. In particular, in case $t \in \mathcal{U}$ the two experiments are identical and thus we can assume wlog. that $t \notin \mathcal{U}$. Furthermore, the simulation performed by $\hat{\mathsf{A}}$ is perfect, so that if A tells apart the two experiments, $\hat{\mathsf{A}}$ distinguishes if the commitment c_t comes from $t||s_t$ or from $t||\hat{s}_t$, thus breaking the statistical hiding property of $(\mathsf{Gen}, \mathsf{Com})$. The lemma follows. □

Lemma 3. $\{\mathbf{Hyb}^n_{\Sigma^*,\mathsf{A}}(\lambda, 0)\}_{\lambda \in \mathbb{N}} \approx_c \{\mathbf{Hyb}^n_{\Sigma^*,\mathsf{A}}(\lambda, 1)\}_{\lambda \in \mathbb{N}}$.

Proof. The proof is down to the privacy of the underlying secret sharing scheme Σ. By contradiction, assume that there exists a PPT adversary A that can distinguish between the experiment with $b = 0$ and $b = 1$ with noticeable advantage. Consider the following PPT adversary $\hat{\mathsf{A}}$ attacking the privacy of Σ.

- Run $(\omega, \zeta) \leftarrow\!\!\$\ \mathsf{S}_0(1^\lambda)$ and $pk \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$ and write $\omega^* = (\omega, pk)$.
- Run $(\alpha_1, m_0, m_1, \mathcal{U} \notin \mathcal{A}) \leftarrow\!\!\$\ \mathsf{A}_1(\omega^*)$.
- Output $(m_0, m_1, \mathcal{U})$ to the challenger, receiving the shares $(s_i)_{i \in \mathcal{U}}$.
- Generate $\hat{m} \leftarrow\!\!\$\ \mathcal{M}$ and compute $(\hat{s}_1, \ldots, \hat{s}_n) \leftarrow\!\!\$\ \mathsf{Share}(\hat{m})$.
- For all $i \in [n]$, generate $r_i \leftarrow\!\!\$\ \mathcal{R}$ and compute c_i as follows. If $i \in \mathcal{U}$, compute $c_i = \mathsf{Com}(pk, i||s_i; r_i)$; otherwise, compute $c_i = \mathsf{Com}(pk, i||\hat{s}_i; r_i)$.
- For all $i, j \in [n]$ such that $i \neq j$, compute $\pi_i^j \leftarrow\!\!\$\ \mathsf{S}_1(\zeta, c_j, c_i)$.
- For all $i \in \mathcal{U}$, write $s_i^* = (s_i, r_i, (c_j)_{j \neq i}, (\pi_j^i)_{j \neq i})$.
- Run $b' \leftarrow\!\!\$\ \mathsf{A}_2(\alpha_1, (s_u^*)_{u \in \mathcal{U}})$.
- Output b'.

For the analysis, note that the reduction is perfect. Therefore, if A distinguishes between the two experiments with noticeable advantage, $\hat{\mathsf{A}}$ distinguishes the two messages using the unauthorized set of shares with noticeable advantage, thus breaking privacy for Σ. This concludes the proof. □

Continuous non-malleability. Recall the experiment $\mathbf{JSTamper}_{\Sigma^*,\mathsf{A}}(\lambda, b)$ defining continuous non-malleability for Σ^*. In particular, expanding the definitions of Init^* and Share^*:

$\underline{\mathbf{JSTamper}_{\Sigma^*,\mathsf{A}}(\lambda, b)}$:
$\omega \leftarrow\!\!\$\ \mathsf{CRSGen}(1^\lambda),\ pk \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda),\ \omega^* := (\omega, pk)$
$(\mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_k), m_0, m_1, \alpha_1) \leftarrow\!\!\$\ \mathsf{A}_1(\omega^*)$

Oracle $\mathcal{O}'_{\mathsf{nmss}}(s^*, \mathcal{T}, (f_1, \ldots, f_k))$:

If $\mathtt{stop} = \mathtt{true}$ return $\perp$
$\forall i \in [k] : \tilde{s}^*_{\mathcal{B}_i} := f_i(s^*_{\mathcal{B}_i})$
$\tilde{s}^* = (\tilde{s}^*_1, \ldots, \tilde{s}^*_n)$
If $\mathsf{Check}_{\mathcal{T}}(\omega^*, \tilde{s}^*) = 0$
  Return $\perp$ and $\mathtt{stop} \leftarrow \mathtt{true}$
$\forall t \in \mathcal{T}$:
  $s^*_t = (s_t, r_t, (c_j)_{j \neq t}, (\pi^t_j)_{j \neq t})$
  $\tilde{s}^*_t = (\tilde{s}_t, \tilde{r}_t, (\tilde{c}_j)_{j \neq t}, (\tilde{\pi}^t_j)_{j \neq t})$
If $\forall j \in [n], \tilde{c}_j = c_j$, return ♥
If $\exists j_1, j_2 \in [n] : \tilde{c}_{j_i} = c_{j_2}$, abort
$\tilde{\mathcal{I}} = \{i \in [n] : \tilde{c}_i \notin (c_j)_{j \in [n]}\}$
$\forall i \in [k] : \tilde{m}_i \leftarrow\!\!\$\ \mathsf{Extract}_{\mathcal{T}, \tilde{\mathcal{I}}}(\xi, \tilde{s}^*_{\mathcal{B}_i})$
If $\exists i, j \in [k] : \tilde{m}_i \neq \tilde{m}_j$ or $\tilde{m}_i = \perp$:
  Return $\perp$ and $\mathtt{stop} \leftarrow \mathtt{true}$
$\tilde{m} = \tilde{m}_i$
If $\tilde{m} \in \{m_0, m_1\}$
  Return ♥
Else return $\tilde{m}$

Algorithm $\mathsf{Check}_{\mathcal{T}}(\omega^*, (\tilde{s}^*_i)_{i \in \mathcal{I}})$:

$\forall i \in \mathcal{T} \cap \mathcal{I} : \tilde{s}^*_i = (\tilde{s}_i, \tilde{r}_i, (\tilde{c}^i_j)_{j \neq i}, (\tilde{\pi}^i_j)_{j \neq i})$
If $\exists i_1, i_2 \in \mathcal{T}, j \in [n] : \tilde{c}^{i_1}_j \neq \tilde{c}^{i_2}_j$
  Return 0
$\forall i \in \mathcal{T} \cap \mathcal{I} : \tilde{s}^*_i = (\tilde{s}_i, (\tilde{c}_j)_{j \neq i}, (\tilde{\pi}^i_j)_{j \neq i})$
If $\exists i \in \mathcal{T} \cap \mathcal{I} : \tilde{c}_i \neq \mathsf{Com}(pk, i || \tilde{s}_i; \tilde{r}_i)$
  Return 0
If $\exists j \in \mathcal{T}, i \in [n] : \mathsf{Ver}(\omega, \tilde{c}_j, (\tilde{c}_i, \tilde{\pi}^j_i)) = 0$
  Return 0
Return 1

Algorithm $\mathsf{Extract}_{\mathcal{T}, \tilde{\mathcal{I}}}(\xi, (\tilde{s}^*_i)_{i \in \mathcal{I}})$:

$\forall i \in \mathcal{T} \cap \mathcal{I} : \tilde{s}^*_i = (\tilde{s}_i, \tilde{r}_i (\tilde{c}_j)_{j \neq i}, (\tilde{\pi}^i_j)_{j \neq i})$
$\forall t \in \mathcal{T}$:
  If $t \in \tilde{\mathcal{I}}$:
    $\forall j \in [n], j \neq t$:
      $(\mathsf{pos}^j_t || \tilde{s}^j_t, r^j_t) \leftarrow\!\!\$\ \mathsf{K}(\xi, \tilde{c}_j, (\tilde{c}_t, \tilde{\pi}^j_t))$
  Else:
    $\forall j \in \tilde{\mathcal{I}}$:
      $(\mathsf{pos}^j_t || \tilde{s}^j_t, r^j_t) \leftarrow\!\!\$\ \mathsf{K}(\xi, \tilde{c}_j, (c_t, \tilde{\pi}^j_t))$
  If $\exists j, j_1, j_2 : \tilde{s}^{j_1}_t \neq \tilde{s}^{j_2}_t$ or $r^{j_1}_t \neq r^{j_2}_t$
      or $\mathsf{pos}^j_t \neq t$
    Return $\perp$
  Write $\tilde{s}_t = \tilde{s}^j_t$ for any j
Return $\mathsf{Rec}((\tilde{s}_t)_{t \in \mathcal{T}})$

Figure 3: Construction of the oracle $\mathcal{O}'_{\mathsf{nmss}}$ used in the experiment $\mathbf{Hyb}^2_{\Sigma^*, \mathsf{A}}(\lambda, b)$.

$(s_1, \ldots, s_n) \leftarrow\!\!\$\ \mathsf{Share}(m_b)$
$\forall i \in [n] : r_i \leftarrow\!\!\$\ \mathcal{R}, c_i = \mathsf{Com}(pk, i || s_i; r_i)$
$\forall i, j \in [n], i \neq j : \pi^j_i \leftarrow\!\!\$\ \mathsf{Prove}(\omega, c_j, (c_i, i || s_i, r_i))$
$\forall i \in [n] : s^*_i := (s_i, r_i, (c_j)_{j \neq i}, (\pi^i_j)_{j \neq i})$
$\mathtt{stop} \leftarrow \mathtt{false}$
$(\alpha_2, i^* \in [k]) \leftarrow\!\!\$\ \mathsf{A}_2^{\mathcal{O}_{\mathsf{nmss}}(s^*, \cdot, \cdot), \mathcal{O}_{\mathsf{leak}}(s^*, \cdot)}(\alpha_1)$
Return $\mathsf{A}_3(\alpha_2, s_{\mathcal{B}_{i^*}})$

Consider the following hybrid experiments:

- $\mathbf{Hyb}^1_{\Sigma^*, \mathsf{A}}(\lambda, b)$: consider the simulator $\mathsf{S} = (\mathsf{S}_0, \mathsf{S}_1)$ for the NIZK and replace the instructions $\omega \leftarrow\!\!\$\ \mathsf{CRSGen}(1^\lambda)$ and $\pi^j_i \leftarrow\!\!\$\ \mathsf{Prove}(\omega, c_j, (c_i, i || s_i))$ with $(\omega, \zeta, \xi) \leftarrow\!\!\$\ \mathsf{S}_0(1^\lambda)$ and $\pi^j_i \leftarrow\!\!\$\ \mathsf{S}_1(\zeta, c_j, c_i)$ respectively.
- $\mathbf{Hyb}^2_{\Sigma^*, \mathsf{A}}(\lambda, b)$: replace the oracle $\mathcal{O}_{\mathsf{nmss}}$ with $\mathcal{O}'_{\mathsf{nmss}}$ described in Fig. 3.

We first prove that the above experiments are computationally close, i.e. for all $b \in \{0, 1\}$,

$$\left\{\mathbf{JSTamper}_{\Sigma^*, \mathsf{A}}(\lambda, b)\right\}_{\lambda \in \mathbb{N}} \approx_c \left\{\mathbf{Hyb}^1_{\Sigma^*, \mathsf{A}}(\lambda, b)\right\}_{\lambda \in \mathbb{N}} \approx_c \left\{\mathbf{Hyb}^2_{\Sigma^*, \mathsf{A}}(\lambda, b)\right\}_{\lambda \in \mathbb{N}}.$$

Then, we prove that

$$\left\{\mathbf{Hyb}^2_{\Sigma^*, \mathsf{A}}(\lambda, 0)\right\}_{\lambda \in \mathbb{N}} \approx_c \left\{\mathbf{Hyb}^2_{\Sigma^*, \mathsf{A}}(\lambda, 1)\right\}_{\lambda \in \mathbb{N}}$$

by reduction to the underlying LRSS scheme Σ, thus proving continuous non-malleability:

$$\left\{\mathbf{JSTamper}_{\Sigma^*,\mathsf{A}}(\lambda,0)\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{\mathbf{JSTamper}_{\Sigma^*,\mathsf{A}}(\lambda,1)\right\}_{\lambda\in\mathbb{N}}.$$

Lemma 4. $\forall b \in \{0,1\}$: $\{\mathbf{JSTamper}_{\Sigma^*,\mathsf{A}}(\lambda,b)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{Hyb}^1_{\Sigma^*,\mathsf{A}}(\lambda,b)\}_{\lambda\in\mathbb{N}}$.

Proof. The proof is down to the adaptive multi-theorem zero-knowledge property of the underlying non-interactive zero-knowledge argument system $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Ver})$. By contradiction, assume that there exists a PPT distinguisher $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2, \mathsf{A}_3)$ that, for any fixed $b \in \{0,1\}$, can distinguish between $\{\mathbf{JSTamper}_{\Sigma^*,\mathsf{A}}(\lambda,b)\}_{\lambda\in\mathbb{N}}$ and $\{\mathbf{Hyb}^1_{\Sigma^*,\mathsf{A}}(\lambda,b)\}_{\lambda\in\mathbb{N}}$ with noticeable advantage. Consider the following PPT distinguisher D attacking the zero-knowledge property of Π.

- Upon receiving a CRS ω that is generated either from $\mathsf{CRSGen}(1^\lambda)$ or from the simulator $\mathsf{S}_0(1^\lambda)$ (depending on the experiment that D is currently playing), generate $pk \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$ and write $\omega^* = (\omega, pk)$.
- Run $(\mathcal{B}, m_0, m_1, \alpha_1) \leftarrow\!\!\$\ \mathsf{A}_1(\omega^*)$ and compute $s = (s_1, \ldots, s_n) \leftarrow\!\!\$\ \mathsf{Share}(m_b)$.
- For each $i \in [n]$, generate $r_i \leftarrow\!\!\$\ \mathcal{R}$ and compute $c_i = \mathsf{Com}(pk, i||s_i; r_i)$.
- For each $i, j \in [n]$ such that $i \neq j$, query the challenger with $(c_j, (c_i, i||s_i, r_i))$, obtaining either $\pi_i^j \leftarrow\!\!\$\ \mathsf{Prove}(\omega, c_j, (c_i, i||s_i, r_i))$ or $\pi_i^j \leftarrow\!\!\$\ \mathsf{S}_1(\zeta, c_j, c_i)$ (depending on the experiment that D is currently playing).
- For each $i \in [n]$, write $s_i^* = (s_i, r_i, (c_j)_{j\neq i}, (\pi_j^i)_{j\neq i})$.
- Write $s^* = (s_1^*, \ldots, s_n^*)$ and set $\mathtt{stop} \leftarrow \mathtt{false}$.
- Run $(\alpha_2, i^*) \leftarrow\!\!\$\ \mathsf{A}_2^{\mathcal{O}_{\mathsf{nmss}}(s^*,\cdot,\cdot),\mathcal{O}_{\mathsf{leak}}(s^*,\cdot)}(\alpha_1)$.
- Return the same as $\mathsf{A}_3(\alpha_2, s_{\mathcal{B}_{i^*}})$.

For the analysis, note that the simulation done by D is perfect. In particular, D can impersonate the oracles $\mathcal{O}_{\mathsf{nmss}}$ and $\mathcal{O}_{\mathsf{leak}}$ and can answer all the queries. Thus, D distinguishes with noticeable probability. The lemma follows. □

Lemma 5. $\forall b \in \{0,1\}$: $\{\mathbf{Hyb}^1_{\Sigma^*,\mathsf{A}}(\lambda,b)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{Hyb}^2_{\Sigma^*,\mathsf{A}}(\lambda,b)\}_{\lambda\in\mathbb{N}}$.

Proof. The only difference between the two experiments is how the oracle answers the tampering queries. In both experiments the following checks are performed:

- if there exist indices $i_1, i_2 \in \mathcal{T}, j \in [n]$ such that $c_j^{i_1} \neq c_j^{i_2}$, self-destruct;
- if there exist an index $i \in \mathcal{T}$ such that $\mathsf{Com}(pk, i||s_i; r_i) \neq c_i$, self-destruct;
- if there exist indices $i, j \in \mathcal{T}$ such that $\mathsf{Ver}(\omega, c_j, (c_i, \pi_i^j)) = 0$, self-destruct.

In particular, the check phase is perfectly simulated. Then, the oracle $\mathcal{O}'_{\mathsf{nmss}}$ finds the indices in which a mauled commitment appears.

- If $\forall j \in [n], \tilde{c}_j = c_j$, then $\mathsf{Com}(pk, j||\tilde{s}_j; \tilde{r}_j) = \mathsf{Com}(pk, j||s_j; r_j)$; call $\mathbf{Bad}_1$ the event that $\tilde{s}_j \neq s_j$. If $\mathbf{Bad}_1$ does not happen, the adversary didn't modify any of the shares, and thus the reconstructed message also remains the same $m_b \in \{m_0, m_1\}$.
- If there exist two distinct $j_1, j_2 \in [n]$ such that $\tilde{c}_{j_2} = c_{j_1}$, abort the experiment; call this event $\mathbf{Bad}_2$.
- If none of the above holds, then there exists at least one fresh commitment $\tilde{c}_j$, namely a commitment $\tilde{c}_j \notin (c_i)_{i\in[n]}$; call $\tilde{\mathcal{I}}$ the set of indices for which the commitment is fresh.

Next, the oracle $\mathcal{O}'_{\mathsf{nmss}}$ reconstructs a message for each subset $\mathcal{B}_i$ by extracting the needed shares from the proofs. In particular, for all $t \in \mathcal{T}$, the oracle extracts the pair $(\mathsf{pos}_t^j||\tilde{s}_t^j, r_t^j)$ from $(\tilde{c}_t, \tilde{\pi}_t^j)$ using $\tilde{c}_j$ as label, and this happens for all $j \in [n]$ if $\tilde{c}_t$ is fresh and for all $j \in \tilde{\mathcal{I}}$

otherwise. At this point of the execution, all proofs verify (otherwise, the oracle would have already provoked a self-destruct during the check phase) and all tuples $(\tilde{c}_j, (\tilde{c}_t, \tilde{\pi}_t^j))$ with the above condition on j are fresh, so that the extractor K can extract the pair $(\mathsf{pos}_t^j || \tilde{s}_t^j, \tilde{r}_t^j)$. Call $\mathbf{Bad}_3$ the event that $\mathsf{Com}(pk, \mathsf{pos}_t^j || \tilde{s}_t^j; \tilde{r}_t^j) \neq \tilde{c}_t = \mathsf{Com}(pk, t || \tilde{s}_t; \tilde{r}_t)$ and $\mathbf{Bad}_4$ the event that $\mathsf{pos}_t^j || \tilde{s}_t^j \neq t || \tilde{s}_t$ for any j. If neither $\mathbf{Bad}_3$ nor $\mathbf{Bad}_4$ happen, the algorithm $\mathsf{Extract}$ managed to extract all the shares $(\tilde{s}_t)_{t \in \mathcal{T}}$ and can reconstruct the message. In fact, if $\mathbf{Bad}_3$ does not happen, the extracted shares are consistent with the mauled commits and, if $\mathbf{Bad}_4$ does not happen, for all j_1, j_2, $\mathsf{pos}_t^{j_1} || \tilde{s}_t^{j_1} = \mathsf{pos}_t^{j_2} || \tilde{s}_t^{j_2} = t || \tilde{s}_t$. Finally, the oracle computes $\tilde{m}_i = \mathsf{Rec}((\tilde{s}_t)_{t \in \mathcal{T}})$ and, since $\mathsf{Rec}((\tilde{s}_t)_{t \in \mathcal{T}})$ does not depend on the set $\mathcal{B}_i$ chosen to extract the shares, we can simply write $\tilde{m} = \tilde{m}_i$.

The last part of the reconstruction is identical to the one performed in the original oracle: if $\tilde{m} = \perp$, self-destruct; if $\tilde{m} \in \{m_0, m_1\}$, output ♥; otherwise, output $\tilde{m}$. Call $\mathbf{Bad} = \mathbf{Bad}_1 \cup \mathbf{Bad}_2 \cup \mathbf{Bad}_3 \cup \mathbf{Bad}_4$. If $\mathbf{Bad}$ does not happen, the two oracles $\mathcal{O}_{\mathsf{nmss}}$ and $\mathcal{O}'_{\mathsf{nmss}}$ are equivalent. It remains to show that the event $\mathbf{Bad}$ happens with negligible probability.

- The event $\mathbf{Bad}_1$ happens when $\mathsf{Com}(pk, j || \tilde{s}_j; \tilde{r}_j) = \mathsf{Com}(pk, j || s_j; r_j)$ but $\tilde{s}_j \neq s_j$. A straightforward reduction to the computational binding property of $(\mathsf{Gen}, \mathsf{Com})$ shows that this event happens with negligible probability. By contradiction, suppose that there exists an adversary A that triggers $\mathbf{Bad}_1$ with noticeable probability. Then, we can construct an algorithm $\hat{\mathsf{A}}$ that, upon receiving a setup string $pk \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$, emulates the experiment $\mathbf{Hyb}^1_{\Sigma^*, \mathsf{A}}(\lambda, b)$ and, upon receiving a tampering query by A that triggers the event $\mathbf{Bad}_1$, halts the execution of A and outputs the values $(j || \tilde{s}_j, \tilde{r}_j)$ and $(j || s_j, r_j)$ that result in the same commitment c_j, thus breaking the computational binding property of $(\mathsf{Gen}, \mathsf{Com})$.
- The event $\mathbf{Bad}_2$ happens when $\mathsf{Com}(pk, j_1 || \tilde{s}_{j_1}; \tilde{r}_{j_1}) = \mathsf{Com}(pk, j_2 || s_{j_2}; r_{j_2})$ but $j_1 || \tilde{s}_{j_1} \neq j_2 || s_{j_2}$. Again, a straightforward reduction to the computational binding property of $(\mathsf{Gen}, \mathsf{Com})$ shows that this event happens with negligible probability. The proof is identical to the previous one and thus omitted.
- The event $\mathbf{Bad}_3$ happens when the pair of commitments $(\tilde{c}_j, \tilde{c}_i)$ is fresh and $\mathsf{Ver}(\omega, \tilde{c}_j, (\tilde{c}_i, \pi_i^j)) = 1$, but the relation over $\mathcal{L}^{pk}_{\mathsf{com}}$ does not hold for $(\tilde{c}_i, (i || \tilde{s}_i, \tilde{r}_i))$, i.e. $\mathsf{Com}(pk, i || \tilde{s}_i; \tilde{r}_i) \neq \tilde{c}_i$. A reduction to the true simulation extractability of $(\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Ver})$ shows that this event happens with negligible probability. By contradiction, suppose that there exists an adversary A that triggers $\mathbf{Bad}_3$ with noticeable probability. Then, we can construct the following algorithm $\hat{\mathsf{A}}$.
  - Upon receiving ω such that $(\omega, \zeta, \xi) \leftarrow\!\!\$\ \mathsf{S}_0(1^\lambda)$, generate $pk \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$ and run $(\mathcal{B}, m_0, m_1, \alpha_1) \leftarrow\!\!\$\ \mathsf{A}_1((\omega, pk))$.
  - Compute $(s_1, \ldots, s_n) \leftarrow\!\!\$\ \mathsf{Share}(m_b)$ and, for all $i \in [n]$, generate $r_i \leftarrow\!\!\$\ \mathcal{R}$ and compute $c_i = \mathsf{Com}(pk, i || s_i; r_i)$.
  - For all $i, j \in [n], i \neq j$, query the challenger with $(c_j, (c_i, i || s_i, r_i))$, receiving $\pi_i^j \leftarrow\!\!\$\ \mathsf{S}_1(\zeta, c_j, c_i)$.
  - For all $i \in [n]$, $s_i^* = (s_i, r_i, (c_j)_{j \neq i}, (\pi_j^i)_{j \neq i})$.
  - Run $\mathsf{A}_2(\alpha_1)$, answering to all its leakage queries as in the original experiment. Upon receiving a tampering query:
    - perform the same steps of $\mathcal{O}_{\mathsf{nmss}}$;
    - upon mauling s_t^*, for all $t \in \mathcal{T}, j \in [n], j \neq t$, if $(\tilde{c}_t, \tilde{c}_j)$ is fresh, the commitments are correct and the proof $\tilde{\pi}_j^t$ verifies, save for later the tuple $(\tilde{c}_t, \tilde{c}_j, \tilde{\pi}_j^t)$.
  - Randomly choose and output one of the tuples $(\tilde{c}_t, \tilde{c}_j, \tilde{\pi}_j^t)$.

  For the analysis, we next prove that the reduction breaks true simulation extractability with noticeable probability. Note first that all the proofs simulated by S_0 are relative to true statements, thus we can apply true simulation extractability. Call p the total number

of tampering queries asked by A and call q the query from which $\hat{\mathsf{A}}$ chooses the output. If $\mathbf{Bad}_3$ happens, the probability of $\hat{\mathsf{A}}$ guessing the right query is at least $\frac{1}{p}$; call p^* one of such queries. Furthermore, if $\mathbf{Bad}_3$ happens in query $p^* = q$, $\hat{\mathsf{A}}$ can choose among up to $n^2 - n$ tuples; call (j^*, t^*) the indices referring to one of the tuples in query p^* triggering the event $\mathbf{Bad}_3$ and (j, t) the indices referring to the tuple chosen by $\hat{\mathsf{A}}$. The probability of $\hat{\mathsf{A}}$ breaking true simulation extractability is

$$\begin{aligned}\mathbb{P}\left[\hat{\mathsf{A}} \text{ wins}\right] &\geq \mathbb{P}\left[\mathbf{Bad}_3 \wedge q = p^* \wedge (j,t) = (j^*,t^*)\right] \\ &\geq \mathbb{P}\left[\mathbf{Bad}_3\right]\mathbb{P}\left[q = p^* | \mathbf{Bad}_3\right]\mathbb{P}\left[(j,t) = (j^*,t^*) | \mathbf{Bad}_3 \wedge q = p^*\right].\end{aligned}$$

Since $\mathbb{P}\left[(j,t) = (j^*,t^*) | \mathbf{Bad}_3 \wedge q = p^*\right] = \frac{1}{n^2-n}$ and n is polynomial in λ, the third factor is $\frac{1}{\mathtt{poly}(\lambda)}$. Similarly, $\mathbb{P}\left[q = p^* | \mathbf{Bad}_3\right] = \frac{1}{p(\lambda)} = \frac{1}{\mathtt{poly}(\lambda)}$, being $p(\lambda)$ polynomial in λ. Finally, if $\mathbf{Bad}_3$ happens with noticeable probability, all the three factors are $\frac{1}{\mathtt{poly}(\lambda)}$ and their product is $\frac{1}{\mathtt{poly}(\lambda)}$. This concludes the reduction.

- The event $\mathbf{Bad}_4$ happens when the extractor K successfully extracts a pair $(\mathsf{pos}_t^j || \tilde{s}_t^j, \tilde{r}_t^j)$ that verifies the relation with $\tilde{c}_t$ (i.e. $\mathsf{Com}(pk, \mathsf{pos}_t^j || \tilde{s}_t^j; \tilde{r}_t^j) = \tilde{c}_t$), but $\mathsf{pos}_t^j || \tilde{s}_t^j \neq t || \tilde{s}_t$. Since $\tilde{c}_t = \mathsf{Com}(pk, t || \tilde{s}_t; \tilde{r}_t)$, a straightforward reduction to the computational binding property of $(\mathsf{Gen}, \mathsf{Com})$ shows that this event happens with negligible probability. The proof is similar to the ones about $\mathbf{Bad}_1$ and $\mathbf{Bad}_2$ and thus omitted.

Finally, since $\mathbf{Bad}$ is the union of the above events, its probability is less than or equal to the sum of the probabilities of the $\mathbf{Bad}_i$, thus negligible. The lemma follows. □

Lemma 6. $\{\mathbf{Hyb}^2_{\Sigma^*,\mathsf{A}}(\lambda, 0)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{Hyb}^2_{\Sigma^*,\mathsf{A}}(\lambda, 1)\}_{\lambda\in\mathbb{N}}$.

Proof. The proof is down to the ℓ-bounded leakage-resilience property of Σ. Here we consider an auxiliary family of weakly universal hash functions[10] Ψ. By contradiction, assume that there exists a PPT adversary A that can tell apart if the scheme has been applied on m_0 or on m_1. Consider the following PPT adversary $\hat{\mathsf{A}}$ attacking ℓ-bounded leakage-resilience of Σ.

1. Run $pk \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda)$ and $(\omega, \zeta, \xi) \leftarrow\!\!\$\ \mathsf{S}_0(1^\lambda)$ and write $\omega^* = (\omega, pk)$.
2. Generate $r^{\mathsf{A}} = (r_1^{\mathsf{A}}, r_2^{\mathsf{A}}, r_3^{\mathsf{A}}) \leftarrow\!\!\$\ \mathcal{R}^{\mathsf{A}}$ and run $(\mathcal{B}, m_0, m_1, \alpha_1) \leftarrow \mathsf{A}_1(\omega^*; r_1^{\mathsf{A}})$.
3. Output $(\mathcal{B}, m_0, m_1)$, obtaining access to the leakage oracle.
4. For all $i \in [n]$, generate $r_i \leftarrow\!\!\$\ \mathcal{R}$.
5. Query $(\hat{g}_i^{\mathsf{com}}(pk, r, \cdot))_{i\in[k]}$ to the leakage oracle, obtaining $(c_1, \ldots, c_n)$.
6. Simulate the proofs $(\pi_i^j)_{j\neq i} \leftarrow\!\!\$\ \mathsf{S}_1(\zeta, c_j, c_i)$.
7. Initialize the values $\mathsf{rcp} = ((r_i)_{i\in[n]}, (c_i)_{i\in[n]}, (\pi_i^j)_{j\neq i})$, $\mathsf{adv} = (\mathsf{A}_2, \alpha_1, r_2^{\mathsf{A}})$ and $\mathsf{state} = (\omega^*, \xi, \mathsf{adv}, \mathsf{rcp})$ and the empty string $\Lambda \leftarrow \varepsilon$.
8. Run the following loop:
   - In step q, query $(\hat{g}_i^{\mathsf{sd}}(\mathsf{state}, \bot, \Lambda, 0, \cdot))_{i\in[k]}$ to the leakage oracle, obtaining, for all $i \in [k]$, either $(\mathsf{leak}, \Lambda_q^i)$ or $(\mathsf{done}, q^i_{\mathsf{tamper}})$. Furthermore, for each $i \in [k]$, after receiving $(\mathsf{done}, q^i_{\mathsf{tamper}})$, we can replace $\hat{g}_i^{\mathsf{sd}}$ with a function returning ε in order to save leakage bits.
   - If the result of the query contains at least an element of the form $(\mathsf{leak}, \Lambda_q^i)$, replace any missing Λ_q^i with ε, write $\Lambda_q = (\Lambda_q^1, \ldots, \Lambda_q^k)$ and set $\Lambda \leftarrow \Lambda || \Lambda_q$;

[10] A family of hash functions $\Psi = \{\psi : \{0,1\}^* \to \{0,1\}^\lambda\}$ is *weakly universal* if for all $x, y \in \{0,1\}^*$ such that $x \neq y$, $\mathbb{P}\left[\psi(x) = \psi(y) | \psi \leftarrow\!\!\$\ \Psi\right] \leq \frac{1}{2^\lambda}$.

Function $\hat{g}_i^{\mathsf{com}}(pk, r, (s_j)_{j\in\mathcal{B}_i})$:
Parse $r = (r_1, \ldots, r_n)$
$\forall j \in \mathcal{B}_i : c_j = \mathsf{Com}(pk, j||s_j; r_j)$
Return $(c_j)_{j\in\mathcal{B}_i}$

Function $\hat{g}_i^{\mathsf{sd}}(\mathsf{state}, \psi, \Lambda, q^*, (s_j)_{j\in\mathcal{B}_i})$:
Parse $\mathsf{state} = (\omega^*, \xi, \mathsf{adv}, \mathsf{rcp})$
Parse $\omega^* = (\omega, pk)$ and $\mathsf{adv} = (\mathsf{A}, \alpha, r^{\mathsf{A}})$
Parse $\mathsf{rcp} = ((r_j)_{j\in[n]}, (c_j)_{j\in[n]}, (\pi_j^i)_{j\neq i})$
Parse $\Lambda = (\Lambda_1, \Lambda_2, \ldots)$
$\forall j \in \mathcal{B}_i : s_j^* = (s_j, r_j, (c_\beta)_{\beta\neq j}, (\pi_\beta^j)_{\beta\neq j})$
$q_{\mathsf{leak}} \leftarrow 0,\ q_{\mathsf{tamper}} \leftarrow 0$
$\mathtt{stop} \leftarrow \mathtt{false}$
$x_i \leftarrow \varepsilon$
Run $\mathsf{A}(\alpha, r)$ as follows:
  Upon leak query $(g_1, \ldots, g_k)$:
    $q_{\mathsf{leak}} \leftarrow q_{\mathsf{leak}} + 1$
    If Λ contains $\Lambda_{q_{\mathsf{leak}}}$, answer $\Lambda_{q_{\mathsf{leak}}}$
    Else, quit and return $(\mathsf{leak}, g_i(s^*_{\mathcal{B}_i}))$
  Upon tamper query $(\mathcal{T}, (f_1, \ldots, f_k))$:
    $q_{\mathsf{tamper}} \leftarrow q_{\mathsf{tamper}} + 1$
    $(\tilde{m}, \tilde{c}) \leftarrow\!\!\$\ \hat{f}_i(\mathsf{state}, \mathcal{T}, f_i, (s_j)_{j\in\mathcal{B}_i})$
    $x_i \leftarrow x_i||(\tilde{m}, \tilde{c})$
    If $q_{\mathsf{tamper}} = q^*$
      $y_i \leftarrow \psi(x_i)$
      Quit and return (hash, y_i)
    Else, answer $\tilde{m}$.
If $q^* = 0$ return $(\mathsf{done}, q_{\mathsf{tamper}})$
Else return (done)

Function $\hat{f}_i(\mathsf{state}, \mathcal{T}, f_i, (s_j)_{j\in\mathcal{B}_i})$:
If $\mathtt{stop} = \mathtt{true}$ return $\perp$
Parse $\mathsf{state}, \mathsf{rcp}$ as in $\hat{g}_i^{\mathsf{sd}}$
$\forall j \in \mathcal{B}_i : s_j^* := (s_j, r_j, (c_\beta)_{\beta\neq j}, (\pi_\beta^j)_{\beta\neq j})$
$\tilde{s}^*_{\mathcal{B}_i} := f_i(s^*_{\mathcal{B}_i})$
If $\mathsf{Check}_{\mathcal{T}}(\omega^*, \tilde{s}^*_{\mathcal{B}_i}) = 0$
  Return $\perp$ and $\mathtt{stop} \leftarrow \mathtt{true}$
$\forall t \in \mathcal{T} : \tilde{s}_t^* = (\tilde{s}_t, \tilde{r}_t, (\tilde{c}_j)_{j\neq t}, (\tilde{\pi}_j^t)_{j\neq t})$
$\tilde{c} := (\tilde{c}_j)_{j\in[n]}$
If $\forall j \in [n], \tilde{c}_j = c_j$, return $(\heartsuit, \tilde{c})$
If $\exists j_1, j_2 \in [n] : \tilde{c}_{j_i} = c_{j_2}$, abort
$\tilde{\mathcal{I}} = \{i \in [n] : \tilde{c}_i \notin (c_j)_{j\in[n]}\}$
$\tilde{m} \leftarrow\!\!\$\ \mathsf{Extract}_{\mathcal{T},\tilde{\mathcal{I}}}(\xi, \tilde{s}^*_{\mathcal{B}_i})$
If $\tilde{m} = \perp$ return $\perp$ and $\mathtt{stop} \leftarrow \mathtt{true}$
If $\tilde{m} \in \{m_0, m_1\}$, return $(\heartsuit, \tilde{c})$
Else return $(\tilde{m}, \tilde{c})$

Algorithm $\hat{g}_i^{\mathsf{leak}}(\mathsf{state}, \Lambda, p^*, (s_j)_{j\in\mathcal{B}_i})$:
Parse $\mathsf{state}, \mathsf{adv}, \mathsf{rcp}, \Lambda$ as in $\hat{g}_i^{\mathsf{sd}}$
$\forall j \in \mathcal{B}_i : s_j^* := (s_j, r_j, (c_\beta)_{\beta\neq j}, (\pi_\beta^j)_{\beta\neq j})$
$q_{\mathsf{leak}} \leftarrow 0,\ q_{\mathsf{tamper}} \leftarrow 0,\ \mathtt{stop} \leftarrow \mathtt{false}$
Run $(\alpha_2, i_{\mathsf{aug}}) \leftarrow \mathsf{A}(\alpha, r)$ as follows:
  Upon leak query $(g_1, \ldots, g_k)$:
    $q_{\mathsf{leak}} \leftarrow q_{\mathsf{leak}} + 1$
    If Λ contains $\Lambda_{q_{\mathsf{leak}}}$, answer $\Lambda_{q_{\mathsf{leak}}}$
    Else, quit and return $(\mathsf{leak}, g_i(s^*_{\mathcal{B}_i}))$
  Upon tamper query $(\mathcal{T}, (f_1, \ldots, f_k))$:
    $q_{\mathsf{tamper}} \leftarrow q_{\mathsf{tamper}} + 1$
    If $q_{\mathsf{tamper}} \geq p^*$, answer $\perp$
    Else:
      $(\tilde{m}, \tilde{c}) \leftarrow\!\!\$\ \hat{f}_i(\mathsf{state}, \mathcal{T}, f_i, (s_j)_{j\in\mathcal{B}_i})$
      Answer $\tilde{m}$
Return $(\mathsf{done}, i_{\mathsf{aug}})$

Figure 4: Construction of the functions used by the reduction to emulate tampering with leakage.

- else, break the loop, obtaining the string Λ and the (temporary) number of tamper queries $q_{\mathsf{tamper}} = \min_i\{q^i_{\mathsf{tamper}}\}$.

9. Set $\mathsf{range} = (q_0, q_1) = (0, q_{\mathsf{tamper}})$.
10. Run the following loop:
    - If $q_0 = q_1$, break the loop obtaining $p^* \leftarrow q_0 = q_1$; else, set $q_m = \lfloor \frac{q_0+q_1}{2} \rfloor$.
    - Sample $\psi \leftarrow\!\!\$\ \Psi$.
    - Query $(\hat{g}_i^{\mathsf{sd}}(\mathsf{state}, \psi, \Lambda, q_m, \cdot))_{i\in[k]}$ to the leakage oracle, obtaining, for all $i \in [k]$, either (hash, y_i) or (done).
    - If there exist some $i, i_1, i_2 \in [k]$ such that either $y_{i_1} \neq y_{i_2}$ or the i-th result of the query is (done), replace $(q_0, q_1) \leftarrow (q_0, q_m)$;

- else, replace $(q_0, q_1) \leftarrow (q_m + 1, q_1)$.

11. Set $\Lambda \leftarrow \varepsilon$ (i.e. discard all the leakage queries).
12. Run the following loop:
    - In step q, query $(\hat{g}_i^{\mathsf{leak}}(\mathsf{state}, \Lambda, p^*, \cdot))_{i\in[k]}$ to the leakage oracle, obtaining, for all $i \in [k]$, either $(\mathsf{leak}, \Lambda_q^i)$ or $(\mathsf{done}, i_{\mathsf{aug}})$.
    - If the result of the query contains $(\mathsf{leak}, \Lambda_q^i)_{i\in[k]}$, write $\Lambda_q = (\Lambda_q^1, \ldots, \Lambda_q^k)$ and set $\Lambda \leftarrow \Lambda || \Lambda_q$;
    - else, break the loop, obtaining the string Λ and the index i_{aug} of the chosen subset.
13. Tell the leakage oracle that there are no more queries and obtain the shares $s_{\mathcal{B}_{i_{\mathsf{aug}}}}$.
14. Write, for all $j \in \mathcal{B}_{i_{\mathsf{aug}}}$, $s_j^* = (s_j, r_j, (c_\beta)_{\beta\neq j}, (\pi_\beta^j)_{\beta\neq j})$
15. Run $(\alpha_2, i_{\mathsf{aug}}) \leftarrow \mathsf{A}_2^{\hat{\mathsf{A}}}(\alpha_1; r_2^{\mathsf{A}})$, answering as follows.
    - Upon receiving the q-th leakage query, write $\Lambda = (\Lambda_1, \ldots, \Lambda_q, \ldots)$ and return Λ_q.
    - Upon receiving the q-th tampering query $(\mathcal{T}, f_1, \ldots, f_k)$: if $q \geq p^*$, return $\perp$; else, compute $\tilde{s}^*_{\mathcal{B}_{i_{\mathsf{aug}}}} = f_{i_{\mathsf{aug}}}(s^*_{\mathcal{B}_{i_{\mathsf{aug}}}})$ and $(\tilde{m}, \tilde{c}) \leftarrow\!\!\$\ \hat{f}_{i_{\mathsf{aug}}}(\mathsf{state}, \mathcal{T}, f_{i_{\mathsf{aug}}}, s_{\mathcal{B}_{i_{\mathsf{aug}}}})$ and return $\tilde{m}$.
16. Run $b' \leftarrow\!\!\$\ \mathsf{A}_3(\alpha_2, s_{\mathcal{B}_{i_{\mathsf{aug}}}}; r_3^{\mathsf{A}})$.
17. Return b'.

For the analysis, we must show that $\hat{\mathsf{A}}$ is ℓ-admissible and that the reduction is correct. $\hat{\mathsf{A}}$ makes leakage queries in steps 5, 8, 10 and 12. The leakage amount in step 5 is $\gamma = O(\lambda)$ bits per share. The leakage amount in step 8 is bounded[11] by the ℓ^*-admissibility of A. In step 10, the reduction $\hat{\mathsf{A}}$ uses the binary search algorithm to find the first index p^* in which the mauled commitments or messages from two different subsets differ, thus making up to $\log(q_{\mathsf{tamper}}) = O(\log(\lambda))$ queries; in each query, the information obtained amounts to $\gamma = O(\lambda)$ bits, therefore the overall leakage amount in step 10 is $O(\lambda \log(\lambda))$. The leakage amount in step 12 is bounded again by the ℓ^*-admissibility of A.

Summing up the above, the overall leakage amount made by $\hat{\mathsf{A}}$ from the subset $\mathcal{B}_i$ is

$$|\mathcal{B}_i|\gamma + 2\ell^* + O(\lambda \log(\lambda)) \leq n\gamma + 2\ell^* + O(\lambda \log(\lambda)) = \ell,$$

therefore $\hat{\mathsf{A}}$ is ℓ-admissible.

Now it remains to show that the reduction $\hat{\mathsf{A}}$ behaves, with overwhelming probability, exactly like the challenger in the experiment $\mathbf{Hyb}^2_{\Sigma^*,\mathsf{A}}(\lambda, b)$. First of all, call **Abort** the event in which the computation of $\hat{f}_i$ aborts. This event is similar to the event $\mathbf{Bad}_2$ in proof to Lemma and happens when A manages to break the binding property of the underlying commitment scheme $(\mathsf{Gen}, \mathsf{Com})$, hence happening with negligible probability.

- In steps 1-6, the reduction $\hat{\mathsf{A}}$ determines the randomness of the algorithm A and perfectly simulates the challenger. On the other side, generates the needed randomness, obtains the commitments of the shares from the leakage oracle and simulates the proofs.
- In step 7, the reduction gradually acquires the results of the leakage queries and the number of tamper queries made by A. Since each $\hat{g}_i^{\mathsf{sd}}$ has only access to the set $\mathcal{B}_i$, at some point may exist $i, j \in [k]$ such that the result of a tampering query made by A in $\hat{g}_i^{\mathsf{sd}}$ differs from the result of the same tampering query in $\hat{g}_j^{\mathsf{sd}}$, thus the next leakage queries may also differ. In particular, A may query two different leakage functions for $\mathcal{B}_i$ in $\hat{g}_i^{\mathsf{sd}}$ and $\hat{g}_j^{\mathsf{sd}}$, resulting in an inconsistent answer. However, this is not a problem since the

[11] We should also consider $O(\log(\lambda))$ bits given by the indices q^i_{tamper}. Since $O(\log(\lambda)) + O(\lambda \log(\lambda)) = O(\lambda \log(\lambda))$, they are irrelevant and thus omitted. The same situation appears in step 12.

adversary receiving inconsistent answers is the one simulated in the leakage oracle and such inconsistent answers are discarded in the next steps; in particular, different answers from the same tampering query q should result in a self-destruct only, thus the tampering answers should remain the same before the query q and $\perp$ (thus, again, the same) from q onwards. Finally, note that, since $q^* = 0$ in $\hat{g}_i^{\mathsf{sd}}$, the condition $q_{\mathsf{tamper}} = q^*$ never verifies, thus there's no need to specify an hash function $\psi \in \Psi$.

- In steps 8-9, the reduction uses the binary search to find the first query p^* in which either the mauled commitments or the resulting reconstructions differ. In particular, the reduction samples $\psi \leftarrow\!\!\$\ \Psi$ and the leakage oracle runs up to q_m tampering queries, collecting, for each $i \in [k]$, all the relative commitments and reconstructed messages in a string x_i; then, the leakage oracle halts A, computes the hash y_i of x_i through ψ and outputs y_i. Now the reduction compares all these hashes and halves the search range accordingly. Call **Bad** the event in which there exist two different $x_i \neq x_j$ that generate the same $y_i = y_j$.
- In steps 10-11, the reduction discards all the Λ_i and queries them again. In particular, now $\hat{\mathsf{A}}$ knows when A receives two different answers to the same tampering query (unless **Bad** happens) and repeats the leakage acquisition similarly to the one performed in step 7; however, the new function $\hat{g}_i^{\mathsf{leak}}$ is programmed to answer normally to tamper queries until the p^*-th query, after which the answer is always $\perp$. If **Bad** does not happen, the algorithm A asks the same leakage and tampering queries for all $i \in [k]$ and the results remain the same. At the end of the run, $\hat{g}_i^{\mathsf{leak}}$ returns the index i_{aug} of the subset chosen by A (that, again, is the same for all $\hat{g}_i^{\mathsf{leak}}$ unless **Bad** happens).
- In steps 12-13, the reduction has no more leakage queries to do and can ask for the shares in the subset $\mathcal{B}_{i_{\mathsf{aug}}}$.
- In step 14, the reduction $\hat{\mathsf{A}}$ finally runs A, answering the leakage and tampering queries as in $\hat{g}_{i_{\mathsf{aug}}}^{\mathsf{leak}}$. Again, if **Bad** does not happen, this is a perfect simulation of the challenger: the answers for the leakage queries have been previously computed and the answers for the tampering queries remain the same for every subset $\mathcal{B}_i$ in which they are computed (so, in particular, in $\mathcal{B}_{i_{\mathsf{aug}}}$).
- Finally, in steps 15-16, the reduction $\hat{\mathsf{A}}$ obtains the distinguishing bit b' from A and outputs the same bit.

The simulation made by $\hat{\mathsf{A}}$ is perfect unless the event **Bad** $\cup$ **Abort** happens; in particular, if **Bad** does not happen, the commitments and the reconstructed messages are the same in all subsets until a self-destruct happens, therefore the check is equivalent to the one performed in the original experiment. The event **Bad** happens with negligible probability by weak universality of Ψ and the event **Abort** happens with negligible probability by computational binding of $(\mathsf{Gen}, \mathsf{Com})$. Therefore, if A exists, the reduction $\hat{\mathsf{A}}$ breaks the leakage resilience of Σ by querying up to ℓ bits of leakage and telling apart if the scheme has been applied on m_0 or on m_1 with noticeable probability. This concludes the proof. □

4.4 Concrete Instantiation

Finally, we show how to instantiate Thm. 4 from generic assumptions, thus yielding the statement of Thm. 2 as a corollary. It is well known that true-simulation extractable NIZKs can be obtained from (doubly-enhanced) trapdoor permutations [FLS90, SCO+01, DHLW10], whereas statistically hiding non-interactive commitments—with commitment size $O(\lambda)$ and $2^{-\Omega(\lambda)}$-statistical hiding—can be instantiated from collision-resistant hash functions [HM96].

As for the underlying leakage-resilient secret sharing, we can use the recent construction from [KMS18] which achieves information-theoretic security in the stronger setting where the

attacker can adaptively leak from subsets of shares of size at most $O(\log n)$, in a joint manner. The latter clearly implies leakage resilience under joint $O(n/\log n)$-selective partitioning.

5 Construction in the Plain Model

5.1 Description of the Scheme

We show how to obtain leakage-resilient continuously non-malleable secret sharing for arbitrary access structures in the plain model, with security against individual leakage and tampering attacks. Our construction combines a non-interactive commitment scheme Com with an auxiliary n-party secret sharing scheme $\Sigma = (\mathsf{Share}, \mathsf{Rec})$, as depicted in Fig. 5. The basic idea is to compute a commitment c to the message m being shared, using random coins r; hence, we secret share the string $m||r$ using the underlying sharing function Share, yielding shares $(s_1, \ldots, s_n)$. Hence, the final share of the i-th player is $s_i^* = (c, s_i)$.

We establish the following result. Note that when $n = 2$, we get as a special case the construction of split-state continuously non-malleable codes in the plain model that was originally proposed in [OPVV18], and later simplified in [FV19] by relying on noisy leakage. Our proof can be seen as a generalization of the proof strategy in [FV19] to the case $n > 2$.

Theorem 5. *Let $n \in \mathbb{N}$, and let $\mathcal{A}$ be an arbitrary access structure for n parties without singletons. Assume that:*

(i) Com *is a perfectly binding and computationally hiding non-interactive commitment;*
(ii) Σ *is an n-party ℓ-noisy leakage-resilient one-time non-malleable secret sharing scheme realizing access structure $\mathcal{A}$ against individual leakage and tampering in the plain model, with information-theoretic security and with message space $\mathcal{M}$ such that $|\mathcal{M}| \in \omega(\log(\lambda))$.*

Then, the secret sharing scheme Σ^* described in Fig. 5 is an n-party ℓ^*-noisy leakage-resilient continuously non-malleable secret sharing scheme realizing access structure $\mathcal{A}$ under individual leakage and tampering with computational security in the plain model, as long as $\ell = \ell^* + 1 + \gamma + O(\log \lambda)$ where $\gamma = \log |\mathcal{C}|$ is the size of a commitment.

Let Com be a non-interactive commitment scheme with message space $\mathcal{M}$, randomness space $\mathcal{R}$, and commitment space $\mathcal{C}$. Let $\Sigma = (\mathsf{Share}, \mathsf{Rec})$ be an auxiliary secret sharing scheme realizing access structure $\mathcal{A}$, with message space $\mathcal{M} \times \mathcal{R}$ and share space $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$. Define the following secret sharing scheme $\Sigma^* = (\mathsf{Share}^*, \mathsf{Rec}^*)$, with message space $\mathcal{M}$ and share space $\mathcal{S}^* = \mathcal{S}_1^* \times \cdots \times \mathcal{S}_n^*$ where for each $i \in [n]$ we have $\mathcal{S}_i^* = \mathcal{C} \times \mathcal{S}_i$.

Sharing algorithm Share^*: Upon input a value $m \in \mathcal{M}$, sample random coins $r \leftarrow\!\!\$\, \mathcal{R}$ and compute $c = \mathsf{Com}(m; r)$ and $(s_1, \ldots, s_n) \leftarrow\!\!\$\, \mathsf{Share}(m||r)$. Return the shares $s^* = (s_1^*, \ldots, s_n^*)$, where for each $i \in [n]$ we set $s_i^* = (c, s_i)$.

Reconstruction algorithm Rec^*: Upon input shares $(s_i^*)_{i \in \mathcal{I}}$ parse $s_i^* = (s_i, c_i)$ for each $i \in \mathcal{I}$. Hence, proceed as follows:

(a) If $\exists i_1, i_2 \in \mathcal{I}$ for which $c_{i_1} \neq c_{i_2}$, return $\perp$; else, let the input shares be $s_i^* = (s_i, c)$.
(b) Run $m||r = \mathsf{Rec}((s_i)_{i \in \mathcal{I}})$; if the outcome equals $\perp$ return $\perp$.
(c) If $c = \mathsf{Com}(m; r)$ return m, else return $\perp$.

Figure 5: Leakage-resilient continuously non-malleable secret sharing for arbitrary access structures against individual leakage/tampering in the plain model.

5.2 Proof Overview

Before coming to the proof, we discuss the main intuition behind privacy and continuous non-malleability.

Privacy. In order to show privacy, we need to prove that no PPT attacker can distinguish the distribution of $\mathsf{Share}^*(1^\lambda, m_0)_{\mathcal{U}}$ from $\mathsf{Share}^*(1^\lambda, m_1)_{\mathcal{U}}$, for any choice of messages $m_0, m_1 \in \mathcal{M}$, as well as unauthorized subset $\mathcal{U} \notin \mathcal{A}$. Towards this, consider a modified sharing algorithm $\mathsf{Share}'(1^\lambda, m)$ that is identical to $\mathsf{Share}(1^\lambda, m)$, except that the shares $(s_1, \ldots, s_n)$ are now obtained by secret sharing (via Share) an independent (uniformly random) string $\hat{m}||\hat{r}$.

The proof of privacy then proceeds in two steps. In the first step, we show that for any message $m \in \mathcal{M}$ and unauthorized subset $\mathcal{U} \notin \mathcal{A}$ the distributions $\mathsf{Share}^*(1^\lambda, m)_{\mathcal{U}}$ and $\mathsf{Share}'(1^\lambda, m)_{\mathcal{U}}$ are computationally close; this follows by privacy of Σ, as we can easily turn a distinguisher D telling apart $\mathsf{Share}^*(1^\lambda, m)_{\mathcal{U}}$ and $\mathsf{Share}'(1^\lambda, m)_{\mathcal{U}}$ into a distinguisher $\hat{\mathsf{D}}$ telling apart $\mathsf{Share}(1^\lambda, m||r)_{\mathcal{U}}$ and $\mathsf{Share}(1^\lambda, \hat{m}||r)_{\mathcal{U}}$, for random r.[12]

In the second step, we show that for all messages $m_0, m_1 \in \mathcal{M}$, and for all unauthorized subset $\mathcal{U} \notin \mathcal{A}$, the distributions $\mathsf{Share}'(1^\lambda, m_0)_{\mathcal{U}}$ and $\mathsf{Share}'(1^\lambda, m_1)_{\mathcal{U}}$ are computationally close; this is because in Share' the input to the sharing algorithm Share is decoupled from the input to the commitment Com, which immediately yields a reduction from a distinguisher telling apart $\mathsf{Share}'(1^\lambda, m_0)_{\mathcal{U}}$ and $\mathsf{Share}'(1^\lambda, m_1)_{\mathcal{U}}$ to a distinguisher telling apart $\mathsf{Com}(1^\lambda, m_0)$ and $\mathsf{Com}(1^\lambda, m_1)$. The latter contradicts computational hiding of the commitment.

Non-malleability. The proof of non-malleability follows along the same lines of the proof in [OPVV18, FV19]. The goal is to show that no PPT attacker A can distinguish the experiments $\mathbf{Tamper}_{\Sigma^*,\mathsf{A}}(\lambda, 0)$ and $\mathbf{Tamper}_{\Sigma^*,\mathsf{A}}(\lambda, 1)$ (cf. Def. 5). Towards this, we consider a hybrid experiment $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda, b)$ which is obtained from $\mathbf{Tamper}_{\Sigma^*,\mathsf{A}}(\lambda, b)$ by making two changes: (i) The target secret sharing s^* is computed using the modified algorithm Share' considered already in the proof of privacy; (ii) Whenever the message $\tilde{m}$ reconstructed inside the tampering oracle $\mathcal{O}_{\mathsf{nmss}}$ equals the dummy message $\hat{m}$, we return ♥ as long as the mauled commitment $\tilde{c}_1 = \ldots = \tilde{c}_n$ equals to the original commitment c, and otherwise we return $\perp$ leading to self-destruct.[13]

The first step is to show that, for any b, the distributions $\mathbf{Tamper}_{\Sigma^*,\mathsf{A}}(\lambda, b)$ and $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda, b)$ are statistically close. The proof is by induction on the number of tampering queries p asked by the attacker A to the target $\mathcal{O}_{\mathsf{nmss}}$ oracle.

Induction basis. Assume we have an unbounded attacker A telling apart the experiments $\mathbf{Tamper}_{\Sigma^*,\mathsf{A}}(\lambda, b)$ and $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda, b)$ using a single tampering query $(\mathcal{T}, (f_1, \ldots, f_n))$ to the target $\mathcal{O}_{\mathsf{nmss}}(s^*, \cdot, \cdot)$ oracle. We build an unbounded attacker $\hat{\mathsf{A}}$ whose goal is to distinguish $\mathbf{Tamper}_{\Sigma,\hat{\mathsf{A}}}(\lambda, 0)$ and $\mathbf{Tamper}_{\Sigma,\hat{\mathsf{A}}}(\lambda, 1)$. Roughly, the reduction works as follows. When A outputs $m_0, m_1 \in \mathcal{M}$, we pick uniformly $\hat{m}, r, \hat{r}$ and forward $\hat{m}_0 = m_b||r$ and $\hat{m}_1 = \hat{m}||\hat{r}$ to the challenger. Denote by s the target secret sharing. Note that the above choice of the messages $\hat{m}_0, \hat{m}_1$ yields a perfectly distributed copy of s^* inside the target leakage and tampering oracles $\mathcal{O}_{\mathsf{leak}}(s, \cdot)$ and $\mathcal{O}_{\mathsf{nmss}}(s, \cdot, \cdot)$, by hard-wiring to each query the value $c = \mathsf{Com}(m_b; r)$. This allows to trivially simulate all the leakage queries asked by A.

[12] In particular, upon receiving $(\hat{s}_u)_{u \in \mathcal{U}}$, distinguisher $\hat{\mathsf{D}}$ simply appends $c = \mathsf{Com}(m; r)$ to each share s_u, and finally runs D upon input the resulting set of shares.

[13] The second change is needed as otherwise an attacker could easily distinguish between the original experiment and the hybrid by tampering with the identity function.

In order to simulate A's tampering query $(\mathcal{T}, (f_1, \ldots, f_n))$, we proceed as follows. We first leak the mauled commitments $(\tilde{c}_t)_{t\in\mathcal{T}}$ via the target leakage oracle. Hence, we use the target tampering oracle to obtain the mauled reconstructed value $\tilde{m}||\tilde{r}$ corresponding to $\tilde{s}^*_{\mathcal{T}}$. Finally, we can use knowledge of $\tilde{m}||\tilde{r}$ and $(\tilde{c}_t)_{t\in\mathcal{T}}$ to simulate the output of A's tampering query correctly, with all but a negligible[14] probability.

Inductive step. Assume now that the real experiment and the hybrid are indistinguishable for any unbounded attacker asking at most p tampering queries, but there exists an unbounded adversary A that can tell them apart using $p+1$ tampering queries. We construct an unbounded attacker $\hat{\mathsf{A}}$ attacking leakage-resilient one-time non-malleability of Σ as in the proof for the induction basis. The main idea is to simulate the answer to A's first p tampering queries via limited leakage; assuming this can be done, we can then use A's last tampering query in order to define $\hat{\mathsf{A}}$'s unique tampering query exactly as in the previous case.

In particular, for each tampering query $(\mathcal{T}^{(q)}, (f_1^q), \ldots, f_n^{(q)})$ asked by A, the strategy of the reduction is to leak the mauled commitments $(\tilde{c}_t)_{t\in\mathcal{T}^{(q)}}$. Hence, if the commitments are all equal to a single value $\tilde{c}$, attacker $\hat{\mathsf{A}}$ finds by brute force the corresponding[15] opening $\tilde{m}$ and uses this value to answer A's query, and otherwise it sets $\tilde{m} = \perp$ with consequent self-destruct. To show that this strategy is sound, we need to overcome two challenges:

1. **(Adjusting the final guess.)** Note that by answering the first p tampering queries by inverting the mauled commitment $\tilde{c}$ (when this is possible and such value is uniquely determined), the reduction neglects the possibility that the answer to one of tampering queries could be $\perp$ (with consequent self-destruct) due to the fact that the inner mauled secret sharing $\tilde{s}$ encodes an invalid message.
   This might cause an inconsistency in the simulation. However, as originally shown in [OPVV18], the latter problem can be overcome (in the information-theoretic setting) by having the reduction asking an additional leakage query indicating whether the simulation of the first p tampering queries was correct or not. Since by the induction hypothesis the event that A does not provoke a self-destruct within the first p tampering queries must be noticeable, the extra leakage allows to maintain simulation and in particular preserve the advantage of A in the reduction.
2. **(Bounding the entropy loss.)** In order to conclude the proof, we still have to argue that the reduction did not leak too much information. In fact, note that $\hat{\mathsf{A}}$ for each tampering query leaks from the i-th share the mauled commitment $\tilde{c}_i$. Since the number of tampering queries p is an arbitrary polynomial, we cannot simply bound the entropy loss on each share by the length of a commitment, as this would result in too much leakage. Instead, we borrow a trick from [FNSV18] which allows to bound the entropy loss on each share by the size of a single commitment plus $O(\log \lambda)$. Intuitively, this holds because as long as the commitments $\tilde{c}_i$ leaked from each share are all equal, we can interpret the leakage on the s_i as a function of the other shares $(s_j)_{j\neq i}$. On the other hand, if the mauled commitments are not all equal, the leakage query does reveal information, but that accounts to the size of a single commitment and moreover happens at most once, since the reduction self-destructs afterwards.

[14] When $\tilde{m} = \hat{m}$, the reduction returns ♥ as long as $\tilde{c} = (\tilde{c}_t)_{t\in\mathcal{T}} = c$; this is a perfect simulation as long as the real experiment never outputs $\hat{m}$, which however happens with overwhelming probability if the message length is super-logarithmic in the security parameter.

[15] Note that this value is unique, by perfect binding.

Concluding the proof. Finally, we show that $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,0)$ and $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,1)$ are computationally close. Here, we rely on the computational hiding property of the commitment.

In particular, given a PPT attacker A telling apart the experiments $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,0)$ and $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,1)$ we construct a PPT distinguisher $\hat{\mathsf{D}}$ distinguishing $\mathsf{Com}(1^\lambda,m_0)$ from $\mathsf{Com}(1^\lambda,m_1)$. This is straightforward because in the hybrid experiment the input to the inner secret sharing algorithm Share is detached from the input to the commitment. Thus, the reduction can compute $s=(s_1,\ldots,s_n)$ as a secret sharing of a random and independent string $\hat{m}||\hat{r}$, and use the challenge commitment $\widehat{c}$ to answer all of A's leakage and tampering queries locally.

5.3 Security Analysis

Correctness w.r.t. access structure $\mathcal{A}$ is immediate. We thus need to prove that Σ^* satisfies both privacy and continuous non-malleability.

Privacy. We need to prove that for all pairs of messages $m_0,m_1\in\mathcal{M}$, and for all unqualified subsets $\mathcal{U}\notin\mathcal{A}$, we have that

$$\{(\mathsf{Share}(1^\lambda,m_0))_\mathcal{U}\}_{\lambda\in\mathbb{N}}\approx_c\{(\mathsf{Share}(1^\lambda,m_1))_\mathcal{U}\}_{\lambda\in\mathbb{N}}.$$

Consider a modified sharing algorithm Share' where we let $c=\mathsf{Com}(m,r)$, for $r\leftarrow_\$\mathcal{R}$, as in the original scheme, but $(s_1,\ldots,s_n)$ are now defined by secret sharing an independent random string $\hat{m}||\hat{r}\leftarrow_\$\mathcal{M}\times\mathcal{R}$; the final shares are set to $s'_i=(c,s_i)$ for all $i\in[n]$.

Lemma 7. $\forall m\in\mathcal{M},\forall\mathcal{U}\notin\mathcal{A}:\ \{(\mathsf{Share}^*(1^\lambda,m))_\mathcal{U}\}_{\lambda\in\mathbb{N}}\approx_c\{(\mathsf{Share}'(1^\lambda,m))_\mathcal{U}\}_{\lambda\in\mathbb{N}}$.

Proof. The proof is down to the privacy property of the underlying secret sharing scheme Σ. By contradiction, assume that there exists a PPT distinguisher D, a message $m\in\mathcal{M}$, and an unauthorized subset $\mathcal{U}\notin\mathcal{A}$, such that

$$\left|\mathbb{P}\left[\mathsf{D}((\mathsf{Share}^*(1^\lambda,m))_\mathcal{U})=1\right]-\mathbb{P}\left[\mathsf{D}((\mathsf{Share}'(1^\lambda,m))_\mathcal{U})=1\right]\right|\geq\frac{1}{\mathtt{poly}(\lambda)}.$$

Consider the following PPT distinguisher $\hat{\mathsf{D}}$ attacking privacy of Σ.

- Output messages $\hat{m}_0=m||r$ and $\hat{m}_1=\hat{m}||\hat{r}$, where $\hat{m}\leftarrow_\$\mathcal{M}$ and $r,\hat{r}\leftarrow_\$\mathcal{R}$.
- Upon receiving a set of shares $(\hat{s}_u)_{u\in\mathcal{U}}$, compute $c=\mathsf{Com}(m;r)$ and set $\hat{s}^*_u=(c,\hat{s}_u)$ for each $u\in\mathcal{U}$.
- Return the same as $\mathsf{D}((\hat{s}^*_u)_{u\in\mathcal{U}})$.

For the analysis, note that in case the values $(\hat{s}_u)_{u\in\mathcal{U}}$ are distributed like a secret sharing of $\hat{m}_0$, the distribution of $(\hat{s}^*_1,\ldots,\hat{s}^*_n)$ is identical to that of $\mathsf{Share}^*(1^\lambda,m)$, whereas in case the values $(\hat{s}_u)_{u\in\mathcal{U}}$ are distributed like a secret sharing of $\hat{m}_1$, the distribution of $(\hat{s}^*_1,\ldots,\hat{s}^*_n)$ is identical to that of $\mathsf{Share}'(1^\lambda,m)$. The lemma follows. □

Lemma 8. $\forall m_0,m_1\in\mathcal{M},\forall\mathcal{U}\notin\mathcal{A}:\ \{(\mathsf{Share}'(1^\lambda,m_0))_\mathcal{U}\}_{\lambda\in\mathbb{N}}\approx_c\{(\mathsf{Share}'(1^\lambda,m_1))_\mathcal{U}\}_{\lambda\in\mathbb{N}}$.

Proof. The proof is down to the computational hiding property of the non-interactive commitment scheme Com. By contradiction, assume that there exists a PPT distinguisher D, a pair of messages $m_0,m_1\in\mathcal{M}$, and an unauthorized subset $\mathcal{U}\notin\mathcal{A}$, such that

$$\left|\mathbb{P}\left[\mathsf{D}((\mathsf{Share}'(1^\lambda,m_0))_\mathcal{U})=1\right]-\mathbb{P}\left[\mathsf{D}((\mathsf{Share}'(1^\lambda,m_1))_\mathcal{U})=1\right]\right|\geq\frac{1}{\mathtt{poly}(\lambda)}.$$

Consider the following PPT distinguisher $\hat{\mathsf{D}}$, attacking the hiding property of Com.

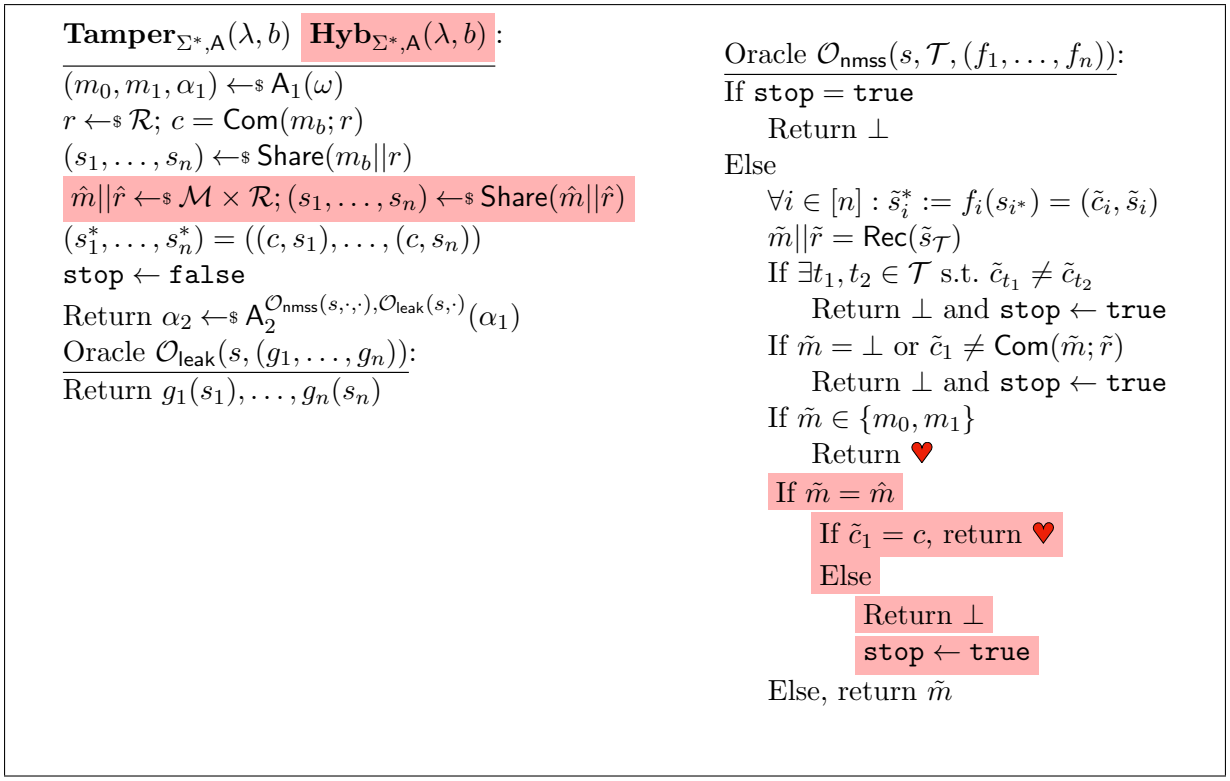

Figure 6: Hybrid experiments in the proof of Theorem 5.

- Upon receiving a target commitment $\hat{c}$ that is either a commitment to m_0 or a commitment to m_1, sample $\hat{m} \leftarrow\$ \mathcal{M}$ and $\hat{r} \leftarrow\$ \mathcal{R}$, and let $(s_1,\ldots,s_n) \leftarrow\$ \mathsf{Share}(\hat{m}||\hat{r})$.
- Define $\hat{s}_i^* = (\hat{c},s_i)$ for each $i\in[n]$.
- Return the same as $\mathsf{D}((\hat{s}_u^*)_{u\in\mathcal{U}})$.

For the analysis, note that in case the value $\hat{c}$ is distributed like a commitment to m_0, the distribution of $(\hat{s}_1^*,\ldots,\hat{s}_n^*)$ is identical to that of $\mathsf{Share}'(1^\lambda,m_0)$, whereas in case the value $\hat{c}$ is distributed like a commitment to m_1, the distribution of $(\hat{s}_1^*,\ldots,\hat{s}_n^*)$ is identical to that of $\mathsf{Share}'(1^\lambda,m_1)$. The lemma follows. □

Fix arbitrary messages $m_0,m_1\in\mathcal{M}$ and $\mathcal{U}\notin\mathcal{A}$. Combining the two lemmas above, we have obtained:

$$\left\{(\mathsf{Share}^*(1^\lambda,m_0))_{\mathcal{U}}\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{(\mathsf{Share}'(1^\lambda,m_0))_{\mathcal{U}}\right\}_{\lambda\in\mathbb{N}}$$
$$\approx_c \left\{(\mathsf{Share}'(1^\lambda,m_1))_{\mathcal{U}}\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{(\mathsf{Share}^*(1^\lambda,m_1))_{\mathcal{U}}\right\}_{\lambda\in\mathbb{N}}.$$

Continuous non-malleability. Let $\mathbf{Tamper}_{\Sigma^*,\mathsf{A}}(\lambda,b)$ be the original experiment defining continuous non-malleability of Σ^*. Consider a modified experiment $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,b)$, where we replace $(s_1,\ldots,s_n)$ with a secret sharing of a random and independent value $\hat{m}||\hat{r}\leftarrow\$\,\mathcal{M}\times\mathcal{R}$; when answering tampering queries, the experiment returns ♥ in case the mauled message $\tilde{m}$ happens to be equal to $\hat{m}$. See Fig. 6 for a more formal description.

We first prove that the above experiments are computationally close by induction over the number of tampering queries $p\in\mathtt{poly}(\lambda)$ asked by adversary A. Towards this, let us denote by $\mathbf{Tamper}_{\Sigma^*,\mathsf{A}}(\lambda,p,b)$ (resp. $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,p,b)$) the original (resp. hybrid) experiment where

adversary A is limited to ask exactly p queries to oracle $\mathcal{O}_{\mathsf{nmss}}$. The lemma below constitutes the basis of the induction.

Lemma 9. $\forall b \in \{0,1\}: \left\{\mathbf{Tamper}_{\Sigma^*,\mathsf{A}}(\lambda,1,b)\right\}_{\lambda\in\mathbb{N}} \approx_s \left\{\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,1,b)\right\}_{\lambda\in\mathbb{N}}$.

Proof. The proof is down to the statistical leakage-resilient one-time non-malleability of Σ. Fix $b = 0$ (the proof for the other case being identical). Assume that there exists an unbounded adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ which can distinguish $\mathbf{Tamper}_{\Sigma^*,\mathsf{A}}(\lambda,1,0)$ and $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,1,0)$ with non-negligible probability. By an averaging argument, this means that there must exist values $r \in \mathcal{R}$ and $\hat{m}||\hat{r} \in \mathcal{M} \times \mathcal{R}$ such that A distinguishes the two experiments when we fix these particular values of r and $\hat{m}||\hat{r}$. Let $s = (s_1, \ldots, s_n)$ be the target encoding in the tampering experiment relative to $(\mathsf{Share}, \mathsf{Rec})$. Consider the following adversary $\hat{\mathsf{A}}$ attacking Σ, given as input the above fixed values $r, \hat{m}, \hat{r}$.

1. Run $\mathsf{A}_1(1^\lambda)$, obtaining a pair of messages (m_0, m_1) and auxiliary state α_1; send $\hat{m}_0 = m_0||r$ and $\hat{m}_1 = \hat{m}||\hat{r}$ to the challenger, and compute $c := \mathsf{Com}(m_0; r)$. Run $\mathsf{A}_2(\alpha_1)$.
2. For each leakage query $(g_1, \ldots, g_n)$ asked by A_2, define the leakage function $\hat{g}_i$ that hard-wires (a description of) g_i, c, and returns the same as $g_i(c, s_i)$. Forward $(\hat{g}_1, \ldots, \hat{g}_n)$ to the target leakage oracle.
3. Upon input the only tampering query $(\mathcal{T}, (f_1, \ldots, f_n))$ from A_2, proceed as follows.
   (a) For each $t \in \mathcal{T}$, define the leakage function $\hat{h}_t$ that hard-wires (a description of) f_t, and returns the commitment $\tilde{c}_t$ such that $f_t(c, s_t) = (\tilde{c}_t, \tilde{s}_t)$.
   (b) Forward $(\hat{h}_t)_{t\in\mathcal{T}}$ to the target leakage oracle,[16] obtaining values $(\tilde{c}_t)_{t\in\mathcal{T}}$.
   (c) Define the tampering function $\hat{f}_i$ that hard-wires c and (a description of) f_i, and, upon input s_i, returns the value $\tilde{s}_i$ specified by $f_i(c, s_i) = (\tilde{c}_i, \tilde{s}_i)$.
   (d) Forward $(\mathcal{T}, (\hat{f}_1, \ldots, \hat{f}_n))$ to the target tampering oracle, obtaining $\tilde{m}||\tilde{r} \in \mathcal{M} \times \mathcal{R} \cup \{\bot, \heartsuit\}$. Hence:
      - If there exist $t_1, t_2 \in \mathcal{T}$ such that $\tilde{c}_{t_1} \neq \tilde{c}_{t_2}$, return $\bot$ to A; else, let $\tilde{c}$ be the unique commitment returned by the leakage oracle.
      - If $\tilde{m}||\tilde{r} = \bot$ or is not a valid opening of $\tilde{c}$, return $\bot$ to A.
      - If $\tilde{m}||\tilde{r} = \heartsuit$, in case $\tilde{c} = c$ return $\heartsuit$ to A (and otherwise $\bot$).
      - If $\tilde{m} \in \{m_0, m_1\}$, return $\heartsuit$ to A. Else, if $\tilde{m} = \hat{m}$ return $\heartsuit$ to A in case $\tilde{c} = c$, and otherwise return $\bot$ to A.
      - Else, return $\tilde{m}$ to A.
4. Output the same guess as that of A_2.

For the analysis, we next prove that the simulation performed by the above reduction is perfect with overwhelming probability. First, depending on the target $(s_1, \ldots, s_n)$ being either a secret sharing of $\hat{m}_0$ or of $\hat{m}_1$, for every $i \in [n]$ the input to the tampering function f_i (resp. leakage function g_i) emulated inside $\hat{f}_i$ (resp. $\hat{g}_i$) is identically distributed to the i-th share of the target secret sharing in either experiment $\mathbf{Tamper}_{\Sigma^*,\mathsf{A}}(\lambda,0,1)$ or $\mathbf{Hyb}(\lambda,0,1)$, with our fixed choice of $r, \hat{m}, \hat{r}$. Second, the answer to A_2's tampering query is simulated correctly, with all but a negligible probability. Indeed:

- If $\mathsf{Rec}(\tilde{s}_\mathcal{T})$ yields $\bot$, both the real and the hybrid experiment would return $\bot$, which is perfectly emulated by the reduction.
- If $\mathsf{Rec}(\tilde{s}_\mathcal{T})$ yields $\heartsuit$, it means that the inner secret sharing reconstructs to either $\hat{m}_0 = m_0||r$ or to $\hat{m}_1 = \hat{m}||\hat{r}$. Without loss of generality, assume further that the commitments

[16] More precisely, the leakage oracle takes as input n functions (one for each of the shares), but we can simply assume that $\hat{h}_i$ equals the empty string ε for each $i \notin \mathcal{T}$.

in the tampered shares are all equal to a single value $\tilde{c}$.[17] There are 4 possible cases: either both experiments output $\hat{m}_0$, or both experiments output $\hat{m}_1$, or one experiment outputs $\hat{m}_0$ while the other outputs $\hat{m}_1$. However, since the view in the real experiment is independent of the value $\hat{m}$, except with negligible probability $2^{-\omega(\log(\lambda))}$, we can condition on the event that the real experiment does not output this value. Thus, there are only two cases to consider:

(i) Both the real and the hybrid experiment return $\hat{m}_0 = m_0||r$.
(ii) The real experiment returns $\hat{m}_0 = m_0||r$, whereas the hybrid returns $\hat{m}_1 = \hat{m}||\hat{r}$.

In both cases, the output of the two experiments is equal to ♥ in case $\tilde{c} = c$, and otherwise both experiments return $\perp$. This is exactly what the reduction does. Hence, the simulation is perfect except with negligible probability.

- If $\mathsf{Rec}(\tilde{s}_{\mathcal{T}})$ yields some value $\tilde{m}||\tilde{r} \notin \{♥, \perp\}$, it means in particular that $\tilde{m}||\tilde{r} \notin \{\hat{m}_0, \hat{m}_1\}$. In such a case both experiments return $\perp$ in case the modified commitment $\tilde{c}$ does not match the opening $\tilde{m}, \tilde{r}$. Otherwise, it means that the modified shares produced by A lead to a valid message $\tilde{m} \in \mathcal{M}$. Thus, the output of both experiments would either be ♥ or $\tilde{m}$ (depending on $\tilde{m}$ being equal to one of the two messages m_0, m_1 or not).

Finally, note that as long as A is ℓ^*-admissible, $\hat{\mathsf{A}}$ is $\hat{\ell}$-admissible for $\hat{\ell} = \ell^* + \gamma$, and since $\hat{\ell} < \ell$, attacker $\hat{\mathsf{A}}$ is ℓ-admissible. Thus, we can conclude that the distinguishing advantage of $\hat{\mathsf{A}}$ is the same as that of A which concludes the proof of the lemma. □

The lemma below constitutes the inductive step.

Lemma 10. *Fix any $p \in \mathtt{poly}(\lambda)$. Assume that for all $b \in \{0,1\}$,*

$$\left\{\mathbf{Tamper}_{\Sigma^*,\mathsf{A}}(\lambda, p, b)\right\}_{\lambda\in\mathbb{N}} \approx_s \left\{\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda, p, b)\right\}_{\lambda\in\mathbb{N}}.$$

Then: $\left\{\mathbf{Tamper}_{\Sigma^*,\mathsf{A}}(\lambda, p+1, b)\right\}_{\lambda\in\mathbb{N}} \approx_s \left\{\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda, p+1, b)\right\}_{\lambda\in\mathbb{N}}$, for all $b \in \{0,1\}$.

Proof. The proof is down to the statistical leakage-resilient one-time non-malleability of Σ. Fix $b = 0$ (the proof for the other case being identical). Assume that there exists an unbounded adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ which can distinguish $\mathbf{Tamper}_{\Sigma^*,\mathsf{A}}(\lambda, p+1, 0)$ and $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda, p+1, 0)$ with non-negligible probability. By an averaging argument, this means that there must exist values $r \in \mathcal{R}$ and $\hat{m}||\hat{r} \in \mathcal{M}\times\mathcal{R}$ such that A distinguishes the two experiments when we fix these particular values of r and $\hat{m}||\hat{r}$. Let $s = (s_1, \ldots, s_n)$ be the target encoding in the tampering experiment relative to $(\mathsf{Share}, \mathsf{Rec})$. Consider the following adversary $\hat{\mathsf{A}}$ attacking Σ, given as input the above fixed values $r, \hat{m}, \hat{r}$.

1. Run $\mathsf{A}_1(1^\lambda)$, obtaining a pair of messages (m_0, m_1) and auxiliary state α_1; send $\hat{m}_0 = m_0||r$ and $\hat{m}_1 = \hat{m}||\hat{r}$ to the challenger, and compute $c := \mathsf{Com}(m_0; r)$. Run $\mathsf{A}_2(\alpha_1)$.
2. The reduction keeps a list of all possible secret sharing of the messages $\hat{m}_0, \hat{m}_1$; let $\hat{\mathcal{S}}_1^{(1)}, \ldots, \hat{\mathcal{S}}_n^{(1)}$ be the initial sets.
3. For each leakage query $(g_1, \ldots, g_n)$ asked by A_2, define the leakage function $\hat{g}_i$ that hard-wires (a description of) g_i, c, and returns the same as $g_i(c, s_i)$. Forward $(\hat{g}_1, \ldots, \hat{g}_n)$ to the target leakage oracle.
4. For each $q \in [p]$, upon input the q-th tampering query $(\mathcal{T}^{(q)}, (f_1^{(q)}, \ldots, f_n^{(q)}))$, proceed as follows.

[17] In fact, if this is not the case, both experiments return $\perp$, which is once again perfectly emulated by the reduction.

(a) For each $t \in \mathcal{T}^{(q)}$, define the leakage function $\hat{h}_t^{(q)}$ that hard-wires (a description of) $f_t^{(q)}$, and returns the commitment $\tilde{c}_t^{(q)}$ such that $f_t^{(q)}(c, s_t) = (\tilde{c}_t^{(q)}, \tilde{s}_t)$.

(b) Forward $(\hat{h}_t^{(q)})_{t \in \mathcal{T}^{(q)}}$ to the target leakage oracle,[18] obtaining values $(\tilde{c}_t^{(q)})_{t \in \mathcal{T}^{(q)}}$. Hence:

- If there exist $t_1, t_2 \in \mathcal{T}^{(q)}$ such that $\tilde{c}_{t_1}^{(q)} \neq \tilde{c}_{t_2}^{(q)}$, set $\tilde{m}^{(q)} = \perp$ and self-destruct; else, let $\tilde{c}^{(q)}$ be the unique commitment returned by the leakage oracle.
- Find by brute force the opening $\tilde{m}^{(q)}$ (i.e., $\tilde{c}^{(q)} = \mathsf{Com}(\tilde{m}^{(q)}; \tilde{r}^{(q)})$ for some $\tilde{r}^{(q)} \in \mathcal{R}$); if no such value is found, set $\tilde{m}^{(q)} = \perp$ and self-destruct.
- If $\tilde{m}^{(q)} \in \{m_0, m_1\}$, re-define $\tilde{m}^{(q)} := \heartsuit$. Else, if $\tilde{m}^{(q)} = \hat{m}$ re-define $\tilde{m}^{(q)} := \heartsuit$ in case $\tilde{c}^{(q)} = c$, and otherwise $\tilde{m}^{(q)} := \perp$ and self-destruct.
- Return $\tilde{m}^{(q)}$ to A.
- For each $i \in [n]$, define $\hat{\mathcal{S}}_i^{(q+1)} \subseteq \hat{\mathcal{S}}_i^{(q)}$ the set of shares for the i-th player that are compatible with the answer to the q-th tampering query being $\tilde{m}^{(q)}$, i.e. $(\hat{s}_1, \ldots, \hat{s}_n) \in \hat{\mathcal{S}}_1^{(q+1)} \times \ldots \times \hat{\mathcal{S}}_n^{(q+1)}$ if and only if $\tilde{m}^{(q)}$ is the output corresponding to $(\mathcal{T}^{(q)}, (f_1^{(q)}, \ldots, f_n^{(q)}))$ in the hybrid experiment.

5. Upon input the last tampering query $(\mathcal{T}^{(p+1)}, (f_1^{(p+1)}, \ldots, f_n^{(p+1)}))$, proceed as follows.

(a) Define the same functions $\hat{h}_t^{(p+1)}$ considered in step 4a, and forward $(\hat{h}_t^{(p+1)})_{t \in \mathcal{T}^{(p+1)}}$ to the target leakage oracle, obtaining values $(\tilde{c}_t^{(p+1)})_{t \in \mathcal{T}^{(p+1)}}$.

(b) Define the tampering function $\hat{f}_i$ that hard-wires c and (a description of) $f_i^{(p+1)}$, and, upon input s_i, returns the value $\tilde{s}_i^{(p+1)}$ specified by $f_i^{(p+1)}(c, s_i) = (\tilde{c}_i^{(p+1)}, \tilde{s}_i^{(p+1)})$.

(c) Forward $(\mathcal{T}^{(p+1)}, (\hat{f}_1, \ldots, \hat{f}_n))$ to the target tampering oracle, obtaining $\tilde{m}^{(p+1)} || \tilde{r}^{(p+1)} \in \mathcal{M} \times \mathcal{R} \cup \{\perp, \heartsuit\}$. Hence:

- If there exist $t_1, t_2 \in \mathcal{T}^{(p+1)}$ such that $\tilde{c}_{t_1}^{(p+1)} \neq \tilde{c}_{t_2}^{(p+1)}$, return $\perp$ to A; else, let $\tilde{c}^{(p+1)}$ be the unique commitment returned by the leakage oracle.
- If $\tilde{m}^{(p+1)} || \tilde{r}^{(p+1)} = \perp$ or is not a valid opening of $\tilde{c}^{(p+1)}$, return $\perp$ to A.
- If $\tilde{m}^{(p+1)} || \tilde{r}^{(p+1)} = \heartsuit$, in case $\tilde{c}^{(p+1)} = c$ return $\heartsuit$ to A (and otherwise $\perp$).
- If $\tilde{m}^{(p+1)} \in \{m_0, m_1\}$, return $\heartsuit$ to A. Else, if $\tilde{m}^{(p+1)} = \hat{m}$ return $\heartsuit$ to A in case $\tilde{c}^{(p+1)} = c$, and otherwise return $\perp$ to A.
- Else, return $\tilde{m}^{(p+1)}$ to A.

6. Check that the simulation up to the first p queries did not cause any inconsistency, due to the fact that the outcome of the q-th tampering queries should have been $\perp$ because $(\tilde{s}_t)_{t \in \mathcal{T}^{(q)}}$ was not a valid secret sharing.

(a) Without loss of generality, assume that the output of A_2 is equal to 0 whenever A believes that the target secret sharing is distributed as in the real experiment.

(b) Define the following special leakage function $\hat{h}_{\mathsf{check}} : \mathcal{S}_1 \to \{0, 1\}$.

- The function hard-wires a description of A_2, the values (c, m_0, m_1), a description of the final tampering query $(\mathcal{T}^{(p+1)}, (f_1^{(p+1)}, \ldots, f_n^{(p+1)}))$, the answer to previous tampering queries $(\tilde{m}^{(1)}, \ldots, \tilde{m}^{(p)})$, the answer to all leakage queries on each of the shares $\Lambda := (\Lambda_1, \ldots, \Lambda_n)$, and the sets $\hat{\mathcal{S}}_2^{(p+1)}, \ldots, \hat{\mathcal{S}}_n^{(p+1)}$.
- Let $\hat{s}^* = ((c, s_1), (c, \hat{s}_2), \ldots, (c, \hat{s}_n))$ be the target secret sharing for each possible remaining set of compatible shares $(\hat{s}_2, \ldots, \hat{s}_n) \in \hat{\mathcal{S}}_2^{(p+1)} \times \cdots \times \hat{\mathcal{S}}_n^{(p+1)}$.
- The output of the function is a bit $\tilde{b}$ such that $\tilde{b} = 1$ if and only if $\mathsf{A}_2(\tilde{m}^{(1)}, \ldots, \tilde{m}^{(p)}, \tilde{m}^*, \Lambda) = 0$ more often when $\hat{s}^*$ is a valid secret sharing of message m_0,

[18] More precisely, the leakage oracle takes as input n functions (one for each of the shares), but we can simply assume that $\hat{h}_i^{(q)}$ equals the empty string ε for each $i \notin \mathcal{T}^{(q)}$.

where $\tilde{m}^*$ is the output of the $\mathcal{O}_{\mathsf{nmss}}$ oracle in the hybrid experiment upon input $(\mathcal{T}^{(p+1)}, (f_1^{(p+1)}, \ldots, f_n^{(p+1)}))$ with target secret sharing $\hat{s}^*$.

(c) Forward $(h_{\mathsf{check}}, \varepsilon, \ldots, \varepsilon)$ to the target leakage oracle, obtaining a bit $\tilde{b}$.

7. Upon receiving a bit b' from A_2, in case $\tilde{b} = 1$ output b', and else return a random guess.

Attacker $\hat{\mathsf{A}}$ runs in exponential time. We now show that its distinguishing advantage is negligibly close to that of A. Indeed:

$$\begin{aligned}
&\left|\mathbb{P}\left[\mathbf{Tamper}_{\Sigma,\hat{\mathsf{A}}}(\lambda,1,0)=1\right]-\mathbb{P}\left[\mathbf{Tamper}_{\Sigma,\hat{\mathsf{A}}}(\lambda,1,1)=1\right]\right| \\
&=\left|\mathbb{P}\left[\mathbf{Tamper}_{\Sigma,\hat{\mathsf{A}}}(\lambda,1,0)=1\wedge\tilde{b}=1\right]-\mathbb{P}\left[\mathbf{Tamper}_{\Sigma,\hat{\mathsf{A}}}(\lambda,1,1)=1\wedge\tilde{b}=1\right]\right| && (1)\\
&\geq \frac{1}{\mathtt{poly}(\lambda)}\cdot\left|\mathbb{P}\left[\mathbf{Tamper}_{\Sigma,\hat{\mathsf{A}}}(\lambda,1,0)=1|\tilde{b}=1\right]-\mathbb{P}\left[\mathbf{Hyb}_{\Sigma,\hat{\mathsf{A}}}(\lambda,1,1)=1|\tilde{b}=1\right]\right| && (2)\\
&\geq \frac{1}{\mathtt{poly}(\lambda)}\cdot\left(\frac{1}{\mathtt{poly}(\lambda)}-\mathtt{negl}(\lambda)\right), && (3)
\end{aligned}$$

where Eq. (1) follows because when $\tilde{b}=0$, the reduction $\hat{\mathsf{A}}$ returns a random guess, and thus its distinguishing advantage is zero; Eq. (2) holds as the induction hypothesis implies that $\tilde{b}=1$ with non-negligible probability, otherwise A generates an invalid secret sharing $(\tilde{s}_1^*,\ldots,\tilde{s}_n^*)$ within the first p tampering queries with overwhelming probability, which in turn means that A can distinguish using less than $p+1$ outputs from the decoding. Finally, Eq. (2) holds because an analysis identical to that of Lemma 9 shows that the view of A is perfectly simulated (except with negligible probability) conditioned on $\tilde{b}=1$, and thus in this case $\hat{\mathsf{A}}$ retains essentially the same advantage as that of A.

In order to conclude the proof, it remains to show that $\hat{\mathsf{A}}$ is ℓ-admissible, for ℓ as in the statement of the theorem. Note that adversary $\hat{\mathsf{A}}$ makes leakage queries at steps 3, 4b, 5a, and 6c, but the leakage queries of step 6c is executed only once, and for a total of at most 1 bit of leakage. Let $q^*\in\mathbb{N}$ be the index of the tampering query, if any, where the commitments retrieved in step 4b happen to be different, and set $q^*=p+1$ in case that never happens; note that q^* is a random variable, which we denote by $\mathbf{q}^*$. Clearly, the leakage queries of step 4b are executed exactly $\min\{q^*,p\}$ times. For each $i\in[n]$, denote by $\mathbf{\Lambda}_i$ the random variable corresponding to the leakage performed by the reduction on the i-th share of the target secret sharing $(\mathbf{S}_1,\ldots,\mathbf{S}_n)$. We can write:

$$\begin{aligned}
\widetilde{\mathbb{H}}_\infty(\mathbf{S}_i|(\mathbf{S}_j)_{j\neq i},\mathbf{\Lambda}_i) &\geq \widetilde{\mathbb{H}}_\infty(\mathbf{S}_i|(\mathbf{S}_j)_{j\neq i},\hat{h}_i^{(1)}(\mathbf{S}_i),\ldots,\hat{h}_i^{(\mathbf{q}^*)}(\mathbf{S}_i),\hat{h}_i^{(p+1)}(\mathbf{S}_i))-\ell^*-1 && (4)\\
&=\widetilde{\mathbb{H}}_\infty(\mathbf{S}_i|(\mathbf{S}_j)_{j\neq i},\mathbf{q}^*,\hat{h}_i^{(\mathbf{q}^*)}(\mathbf{S}_i),\hat{h}_i^{(p+1)}(\mathbf{S}_i))-\ell^*-1 && (5)\\
&\geq \widetilde{\mathbb{H}}_\infty(\mathbf{S}_i|(\mathbf{S}_j)_{j\neq i})-\ell^*-1-\gamma-O(\log\lambda) && (6)
\end{aligned}$$

In the above derivation, Eq. (4) follows by definition of $\mathbf{\Lambda}_i$, and because the adversary A is ℓ^*-admissible and furthermore the leakage performed in step 6c consists of at most 1 bit; Eq. (5) follows by the fact that, for each $q<q^*$, the commitments leaked in step 5a are all identical and thus can be computed as a function of $(\mathbf{S}_j)_{j\neq i}$ and $\mathbf{q}^*$; Eq. (6) follows because either $\mathbf{q}^*=p+1$, and then the min-entropy drop due to the leakage $\hat{h}_i^{(\mathbf{q}^*)}(\mathbf{S}_i),\hat{h}_i^{(p+1)}(\mathbf{S}_i)$ is bounded by the size γ of a commitment, or $\mathbf{q}^*<p+1$, in which case the adversary $\hat{\mathsf{A}}$ self-destructs and only the value $\hat{h}_i^{(\mathbf{q}^*)}(\mathbf{S}_i)$ is leaked (causing a drop of at most γ in the min-entropy bound). The lemma follows. □

Combining Lemma 9 and Lemma 10, we get that for all $b\in\{0,1\}$:

$$\left\{\mathbf{Tamper}_{\Sigma^*,\mathsf{A}}(\lambda,b)\right\}_{\lambda\in\mathbb{N}}\approx_s\left\{\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,b)\right\}_{\lambda\in\mathbb{N}.}$$

The lemma below concludes the proof.

Lemma 11. $\left\{\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,0)\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,1)\right\}_{\lambda\in\mathbb{N}}$.

Proof. Assume that there exists a PPT adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ telling apart $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,0)$ and $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,1)$ with non-negligible probability. We build a PPT distinguisher D such that

$$\Big|\mathbb{P}\left[\mathsf{D}(1^\lambda,\hat{c})) = 1 :\ \hat{c} := \mathsf{Com}(m_0;r); r \leftarrow\!\!\$\ \mathcal{R}\right] - \mathbb{P}\left[\mathsf{D}(1^\lambda,\hat{c}) = 1 :\ \hat{c} := \mathsf{Com}(m_1;r); r \leftarrow\!\!\$\ \mathcal{R}\right]\Big| \geq 1/\mathtt{poly}(\lambda),$$

where (m_0, m_1) are the messages output by $\mathsf{A}_1(1^\lambda)$. This contradicts the computational hiding property of the non-interactive commitment scheme, and thus concludes the proof of the lemma.

Distinguisher D, with black-box access to A_2, and with input $\hat{c} \in \mathcal{C}$, is described below:

- Sample $(s_1, \ldots, s_n) \leftarrow\!\!\$\ \mathsf{Share}(\hat{m}||\hat{r})$, for $\hat{m}||\hat{r} \leftarrow\!\!\$\ \mathcal{M} \times \mathcal{R}$.
- Upon input a leakage query $(g_1, \ldots, g_n)$ from A_2, return $(g_1(\hat{c}, s_1), \ldots, g_n(\hat{c}, s_n))$.
- Upon input the q-th tampering query $(\mathcal{T}^{(q)}, (f_1^{(q)}, \ldots, f_n^{(q)}))$ from A_2, proceed as follows:
  - For each $i \in [n]$, compute
  $$\tilde{s}_i^* := f_i^{(q)}(\hat{c}, s_i) = (\tilde{c}_i, \tilde{s}_i)$$
  and let $\tilde{m}^{(q)}||\tilde{r}^{(q)} = \mathsf{Rec}(\tilde{s}_{\mathcal{T}^{(q)}})$ where $\tilde{s} = (\tilde{s}_1, \ldots, \tilde{s}_n)$.
  - If $\exists t_1, t_2 \in \mathcal{T}^{(q)}$ s.t. $\tilde{c}_{t_1} \neq \tilde{c}_{t_2}$, return $\perp$ to A_2 and self-destruct.
  - If $\tilde{m}^{(q)} = \perp$ or $\tilde{c}_1 \neq \mathsf{Com}(\tilde{m}^{(q)}; \tilde{r}^{(q)})$, return $\perp$ to A_2 and self-destruct.
  - If $\tilde{m}^{(q)} \in \{m_0, m_1\}$, return ♥ to A_2.
  - If $\tilde{m}^{(q)} = \hat{m}$, return ♥ to A_2 in case $\tilde{c}_1 = \hat{c}$ and otherwise return $\perp$ to A_2 and self-destruct.
  - Else, return $\tilde{m}^{(q)}$ to A_2.
- Return the same guess as A_2.

For the analysis, note that the simulation done by D is perfect. In particular, depending on the value $\hat{c}$ being either a commitment to m_0 or a commitment to m_1, the view of A_2 is identical to the view in either experiment $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,0)$ or $\mathbf{Hyb}_{\Sigma^*,\mathsf{A}}(\lambda,1)$. Thus, D distinguishes with non-negligible probability. This finishes the proof. □

6 Statistical One-Time Non-Malleability with Noisy Leakage

Since non-interactive, perfectly binding, commitments can be obtained in the plain model assuming one-to-one one-way functions [GMW87], all that remains in order to derive Thm. 1 as a corollary of Thm. 5 is an unconditional construction of noisy-leakage resilient one-time non-malleable secret sharing for arbitrary access structures. The only known scheme achieving all these properties unconditionally is the one in [KMS18], but unfortunately that scheme only tolerates bounded leakage, and it is unclear how to generalize the proof to the setting of noisy leakage.[19] Hence, we take a different approach and we instead show how to generalize a recent transformation from [BGW19], which is tailored to the case $n = 2$.

[19] This is because [KMS18] relies on lower bounds in communication complexity.

6.1 Asymmetric Noisy-Leakage-Resilient Secret Sharing

Our construction exploits so-called leakage-resilient encryption, as recently introduced by Ball, Guo, and Wichs [BGW19]. To keep the exposition more uniform, we cast their definition in terms of a special 2-out-of-2 leakage-resilient secret sharing satisfying three additional properties: (i) One of the shares is uniformly random, and can be sampled independently from the message; (ii) The shares are almost uncorrelated, namely the distribution of one share in isolation and conditioned on the other share have very similar min-entropy; (iii) The size of the shares are asymmetric, namely one share is substantially larger than the other share. Given a 2-out-of-2 secret sharing scheme $\Sigma = (\mathsf{Share}, \mathsf{Rec})$, abusing notation, for any fixed $s_1 \in \mathcal{S}_1$ and $m \in \mathcal{M}$, we write $s_2 \leftarrow\!\!\$\ \mathsf{Share}(m, s_1)$ for the sharing algorithm that computes share s_2 subject to (s_1, s_2) being a valid sharingof m.

Definition 7 (Asymmetric secret sharing)**.** Let $\Sigma = (\mathsf{Share}, \mathsf{Rec})$ be a 2-out-of-2 secret sharing scheme. We call Σ $(\alpha, \sigma_1, \sigma_2)$-*asymmetric*, if it satisfies the following properties:

(i) For any $s_1 \in \mathcal{S}_1$, and any $m \in \mathcal{M}$, it holds that $\mathsf{Rec}(s_1, \mathsf{Share}(m, s_1)) = m$;
(ii) For any message $m \in \mathcal{M}$, and for all $i \in \{1, 2\}$, it holds that $\widetilde{\mathbb{H}}_\infty(\mathbf{S}_i|\mathbf{S}_{3-i}) \geq \log|\mathcal{S}_i| - \alpha$, where $\mathbf{S}_1, \mathbf{S}_2$ are the random variables corresponding to sampling $s_1 \leftarrow\!\!\$\ \mathcal{S}_1$ and $s_2 \leftarrow\!\!\$\ \mathsf{Share}(m, s_1)$;
(iii) It holds that $\log|\mathcal{S}_1| = \sigma_1$ and $\log|\mathcal{S}_2| = \sigma_2$.

As for security we consider the same security experiment of a leakage-resilient secret sharing, however, we consider a more general class of admissible adversaries:

Definition 8 (Noisy admissibility for asymmetric secret sharing)**.** Let $\Sigma = (\mathsf{Share}, \mathsf{Rec})$ be a 2-out-of-2 secret sharing scheme. We say that an unbounded adversary $\mathsf{A} = (\mathsf{A}_1, \mathsf{A}_2)$ is (ℓ_1, ℓ_2)-asymmetric noisy-leakage admissible ((ℓ_1, ℓ_2)-NLA for short) if it satisfies Def. 6 without property (iii), and using the following variant of property (ii):

(ii) A_2 outputs a sequence of leakage queries (chosen adaptively) $(g^{(q)})_{q\in[p]}$, with $p(\lambda) \in \mathtt{poly}(\lambda)$, such that for all $i \in \{1, 2\}$, and for all $m \in \mathcal{M}$:

$$\widetilde{\mathbb{H}}_\infty\left(\mathbf{S}_i|\mathbf{S}_{3-i}, g_i^{(1)}(\mathbf{S}_i), \cdots, g_i^{(p)}(\mathbf{S}_i)\right) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{S}_i|\mathbf{S}_{3-i}) - \ell_i,$$

where $\mathbf{S}_1$ is uniformly random over $\mathcal{S}_1$ and $\mathbf{S}_2$ is the random variable corresponding to $\mathsf{Share}(m, \mathbf{S}_1)$.

Finally, we say that a 2-out-of-2 secret sharing is augmented (ℓ_1, ℓ_2)-noisy-leakage resilient if it is secure as per Def. 5, against the class of all unbounded adversaries that are (ℓ_1, ℓ_2)-NLA. The theorem below says that there is an unconditional construction of such a leakage-resilient secret sharing that is also asymmetric as per Def. 7. The proof appears in §B.

Theorem 6. *For any $\alpha \in \mathbb{N}$, and for any large enough $\ell_1, \ell_2 \in \mathtt{poly}(\lambda, \alpha)$, there exists $\sigma_1, \sigma_2 \in \mathtt{poly}(\lambda, \alpha)$ and an $(\alpha, \sigma_1, \sigma_2)$-asymmetric secret sharing scheme Σ with message space $\{0, 1\}^\alpha$.*

6.2 Construction

Before presenting our scheme, we establish some notation. Given a reconstruction set $\mathcal{I} = \{i_1, \ldots, i_k\}$, we always assume that $i_j \leq i_{j+1}$ for $j \in [k]$. further, we define the function $\mathtt{nxt}_\mathcal{I} : \mathcal{I} \to \mathcal{I}$ as:

$$\mathtt{nxt}_\mathcal{I}(i_j) := \begin{cases} i_{j+1} & j < k \\ i_1 & \text{otherwise} \end{cases}$$

Let $\Sigma' = (\mathsf{Share}', \mathsf{Rec}')$ be a secret sharing scheme realizing access structure $\mathcal{A}$, with message space $\mathcal{M}$ and share space $\mathcal{S}' = \mathcal{S}'_1 \times \cdots \times \mathcal{S}'_n$ where $\mathcal{S}'_i \subseteq \mathcal{M}''$. Let $\Sigma'' = (\mathsf{Share}'', \mathsf{Rec}'')$ be a 2-out-of-2 *asymmetric* secret sharing scheme with message space $\mathcal{M}''$ and share space $\mathcal{S}'' = \mathcal{S}''_1 \times \mathcal{S}''_2$. Define the following secret sharing scheme $\Sigma = (\mathsf{Share}, \mathsf{Rec})$, with message space $\mathcal{M}$ and share space $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$, where for each $i \in [n]$ we have $\mathcal{S}_i \subseteq (\mathcal{S}''_1)^{n-1} \times (\mathcal{S}''_2)^{n-1}$.

Sharing algorithm Share**:** Upon input a value $m \in \mathcal{M}$, compute $(s'_1, \ldots, s'_n) \leftarrow\!\!\$\ \mathsf{Share}'(m)$. For each $i \in [n]$ and $j \in [n] \setminus \{i\}$, sample a random share $s''_{i,j,1} \leftarrow\!\!\$\ \mathcal{S}''_1$ and compute $s''_{i,j,2} \leftarrow\!\!\$\ \mathsf{Share}''(s'_i, s''_{i,j,1})$. Return the shares $s = (s_1, \ldots, s_n)$, where for each $i \in [n]$ we set $s_i = ((s''_{j,i,1})_{j \neq i}, (s''_{i,j,2})_{j \neq i})$.

Reconstruction algorithm Rec**:** Upon input shares $(s_i)_{i \in \mathcal{I}}$ with $\mathcal{I} \in \mathcal{A}$, parse $s_i = ((s''_{j,i,1})_{j \neq i}, (s''_{i,j,2})_{j \neq i})$. Hence, proceed as follows:

(a) Compute $s'_i = \mathsf{Rec}''(s''_{i,\mathtt{nxt}(i),1}, s''_{i,\mathtt{nxt}(i),2})$ for $i \in \mathcal{I}$;

(b) Return $\mathsf{Rec}'((s'_i)_{i \in \mathcal{I}})$.

Figure 7: Noisy-leakage-resilient one-time statistically non-malleable secret sharing for arbitrary access structures against individual leakage/tampering in the plain model.

and the function $\mathtt{prv}_{\mathcal{I}}$ to be the inverse of $\mathtt{nxt}_{\mathcal{I}}$. Whenever it is clear from the context we omit the reconstruction set $\mathcal{I}$ and simply write $\mathtt{nxt}$ and $\mathtt{prv}$.

Intuitively, our construction (cf. Fig. 7) relies on a one-time non-malleable (but not leakage resilient) secret sharing Σ', and on an asymmetric leakage-resilient secret sharing Σ'. The sharing of a message m is obtained by first sharing m under Σ', obtaining n shares $(s'_1, \ldots, s'_n)$, and then sharing each s_i independently $n-1$ times under Σ'', obtaining pairs of shares $(s''_{i,j,1}, s''_{i,j,2})_{j \neq i}$; the final share of party i is then set to be the collection of right shares corresponding to i and all the left shares corresponding to the parties $j \neq i$. We can now state the main theorem of this section.

Theorem 7. *Let $n \in \mathbb{N}$, and let $\mathcal{A}$ be an arbitrary access structure for n parties without singletons. Assume that:*

(i) Σ' is an n-party one-time non-malleable secret sharing scheme realizing access structure $\mathcal{A}$ against individual tampering in the plain model, with information-theoretic security;

(ii) Σ'' is an $(\alpha, \sigma_1, \sigma_2)$-asymmetric augmented (ℓ_1, ℓ_2)-noisy-leakage-resilient secret sharing scheme.

Then, the secret sharing scheme Σ described in Fig. 7 is an n-party ℓ-noisy leakage-resilient one-time non-malleable secret sharing scheme realizing access structure $\mathcal{A}$ under individual leakage and tampering with statistical security in the plain model, as long as $\ell_1 = \ell + (2n-3)\alpha$ and $\ell_2 = \ell + (2n-3)\alpha + \sigma_1$.

6.3 Proof Overview

Privacy of Σ follows in a fairly straightforward manner from privacy of Σ'. In fact, recall that the shares $s''_{i,j,1}$, with $i, j \in [n]$ and $i \neq j$, are sampled uniformly at random and independently of s'. Thus, in the reduction we can sample these values locally and then define the shares $(s_u)_{u \in \mathcal{U}}$ as a function of the shares $(s'_u)_{u \in \mathcal{U}}$. As for the proof of leakage-resilient one-time non-malleability, the idea is to reduce to the one-time non-malleability of Σ' and simulate the leakage by sampling dummy values for the shares $s''_{i,j,1}, s''_{i,j,2}$.

The main challenge is to make sure that the answer to tampering query $f = (f_1, \ldots, f_n)$ is consistent with the simulated leakage. To this end, in the reduction we define the tampering

function $f' = (f'_1, \ldots, f'_n)$, acting on the shares $s' = (s'_1, \ldots, s'_n)$, as follows. Each function f'_i, upon input s'_i and given the values $(s''_{i,j,1})_{j \neq i}$, samples $(\hat{s}_{i,j,2})_{j \neq i}$ in such a way that for any j the reconstruction $\mathsf{Rec}''(s_{i,j,1}, \hat{s}_{i,j,2})$ yields a share s'_i that is consistent with the simulated leakage using the dummy values. Noisy-leakage resilience of Σ'' guarantees that the function f'_i samples from a valid distribution (namely, a non-empty one). Note that the function f'_i might not be efficiently computable; however, as we are reducing to statistical non-malleability, this is not a problem.

An additional difficulty is that the functions $(f'_t)_{t \in \mathcal{T}}$ need to communicate in order to produce their outputs. In fact, for any $t \in \mathcal{T}$, the function f'_t returns a tampered share for Σ' that depends on the mauled share $\tilde{s}_{\mathtt{prv}(t),t,1}$ (generated by $f_{\mathtt{prv}(t)}$). To overcome this problem, we let the reduction perform an additional leakage query on the dummy values before tampering. Thanks to this extra leakage, the reduction learns the values $\tilde{s}_{\mathtt{prv}(t),t,1}$ for all $t \in \mathcal{T}$, which can be hard-coded in the description of $(f'_t)_{t \in \mathcal{T}}$. Here is where we rely on the asymmetric property of Σ'', which allows us to leak σ_1 bits from the second share.

At this point, a reader familiar with [BGW19] might notice that the two proofs proceed very similarly. However, our proof requires extra care when bounding the amount of leakage performed by the reduction. The key ideas are that: (i) Each of the shares under Σ' is shared using $n-1$ independent invocations of Σ''; and (ii) our reconstruction procedure depends only on one of those (chosen as function of the reconstruction set). Property (i) allows to reduce independent leakage on n shares under Σ to independent leakage on 2 shares under Σ'' by sampling locally the missing $n-2$ shares when reducing to noisy-leakage resilience of Σ''. Property (ii) allows to bound the amount of information the reduction needs to simulate the tampering query to a single short leakage from each of the shares (i.e., the value $\tilde{s}_{\mathtt{prv}(t),t,1}$ for $t \in \mathcal{T}$).

6.4 Security Analysis

The proof relies on the following technical lemma.

Lemma 12. *Let Σ be a 2-out-of-2 secret sharing scheme with message space $\mathcal{M}$. If there exists $\tilde{\sigma} \in \mathbb{R}$ such that for all $m \in \mathcal{M}$ we have $\widetilde{\mathbb{H}}_\infty(\mathbf{S}_a | \mathbf{S}_{3-a}) \geq \tilde{\sigma}$, where $\mathbf{S}_1, \mathbf{S}_2$ are the random variables corresponding to $\mathsf{Share}(m)$, then for all $k \in \mathbb{N}$, any $a_1, \ldots, a_k \in \{1,2\}^k$ and any distribution $\mathbf{D}$ over $\mathcal{M}^k$, we have:*

$$\widetilde{\mathbb{H}}_\infty((\mathbf{S}_{a_j,j})_{j \in [k]} | (\mathbf{S}_{3-a_j,j})_{j \in [k]}) \geq \widetilde{\mathbb{H}}_\infty((\mathbf{S}_{a_j,j})_{j \in [k-1]} | (\mathbf{S}_{3-a_j,j})_{j \in [k-1]}) + \tilde{\sigma},$$

where $\mathbf{S}_{1,j}, \mathbf{S}_{2,j}$ are the random variables corresponding to $\mathsf{Share}(\mathbf{M}_j)$ and $\mathbf{M} = \mathbf{M}_1, \ldots, \mathbf{M}_k \leftarrow\!\!\$\ \mathbf{D}$.

Proof. Let $\bar{\mathbf{S}}_a = (\mathbf{S}_{a_j,j})_{j \in [k-1]}$, $\bar{\mathbf{S}}_{3-a} = (\mathbf{S}_{3-a_j,j})_{j \in [k-1]}$, and similarly let $\bar{\mathbf{S}}'_a = (\mathbf{S}_{a_j,j})_{j \in [k]}$ and $\bar{\mathbf{S}}'_{3-a} = (\mathbf{S}_{3-a_j,j})_{j \in [k]}$. By definition of conditional average min-entropy, we need to show that:

$$\max_{\mathsf{P}} \mathbb{P}\left[\mathsf{P}(\bar{\mathbf{S}}_{3-a}, \mathbf{S}_{3-a_k,k}) = \bar{\mathbf{S}}_a, \mathbf{S}_{a_k,k}\right] \leq 2^{-\tilde{\sigma}} \max_{\mathsf{P}} \mathbb{P}\left[\mathsf{P}(\bar{\mathbf{S}}_{3-a}) = \bar{\mathbf{S}}_a\right].$$

Given a predictor P for the left-side of the equation above, let P' be the predictor that runs as P but outputs only the first $k-1$ values, and let P'' be the predictor that runs as P but outputs only the last value. As we can wlog. assume that P is deterministic, we have that

$\mathsf{P}(\cdot) = \mathsf{P}'(\cdot)||\mathsf{P}''(\cdot)$. We can write:

$$\begin{aligned}
&\max_{\mathsf{P}} \mathbb{P}\left[\mathsf{P}(\bar{\mathbf{S}}'_{3-a}) = \bar{\mathbf{S}}_a, \mathbf{S}_{a_k,k}\right] = \\
&\max_{\mathsf{P}} \mathbb{E}_m\Big[\mathbb{P}\left[\mathsf{P}(\bar{\mathbf{S}}'_{3-a}) = \bar{\mathbf{S}}_a, \mathbf{S}_{a_k,k} \mid \mathbf{M} = m\right]\Big] = \\
&\max_{\mathsf{P}} \mathbb{E}_m\Big[\mathbb{P}\left[\mathsf{P}'(\bar{\mathbf{S}}'_{3-a}) = \bar{\mathbf{S}}_a \mid \mathbf{M} = m,\ \mathsf{P}''(\bar{\mathbf{S}}'_{3-a}) = \mathbf{S}_{a_k,k}\right] \\
&\qquad\qquad \cdot \mathbb{P}\left[\mathsf{P}''(\bar{\mathbf{S}}'_{3-a}) = \mathbf{S}_{a_k,k} | \mathbf{M} = m\right]\Big] = \\
&\max_{\mathsf{P}',\mathsf{P}''} \mathbb{E}_m\Big[\mathbb{P}\left[\mathsf{P}'(\bar{\mathbf{S}}_{3-a}) = \bar{\mathbf{S}}_a \mid \mathbf{M} = m,\ \mathsf{P}''(\mathbf{S}_{3-a_k,k}) = \mathbf{S}_{a_k,k}\right] \\
&\qquad\qquad \cdot \mathbb{P}\left[\mathsf{P}''(\mathbf{S}_{3-a_k,k}) = \mathbf{S}_{a_k,k} | \mathbf{M} = m\right]\Big] = \\
&\max_{\mathsf{P}'} \mathbb{E}_m\Big[\mathbb{P}\left[\mathsf{P}'(\bar{\mathbf{S}}_{3-a}) = \bar{\mathbf{S}}_a \mid \mathbf{M} = m,\ \mathsf{P}''(\mathbf{S}_{3-a_k,k}) = \mathbf{S}_{a_k,k}\right]\Big] \cdot 2^{-\tilde{\sigma}},
\end{aligned}$$

where in the third equality we used the fact that, once we fix $\mathbf{M} = m$, the random variables $(\mathbf{S}_{1,j}, \mathbf{S}_{2,j})$ for any j are independently distributed, and in the last equality we used the hypothesis of the lemma. □

of Thm. 7. We use a hybrid argument. Recall that the experiment $\mathbf{Tamper}_{\Sigma,\mathsf{A}}(\lambda, b)$ computes the target secret sharing by first sampling $s'_1, \ldots, s'_n \leftarrow\!\!\$\ \mathsf{Share}'(m_b)$, and $s''_{i,j,1}, s''_{i,j,2}$ for $i, j \in [n]$ and $j \neq i$, and finally defines $s_1, \ldots, s_n$ by arranging the shares as described in Fig 7.

First hybrid. Let $\mathbf{Hyb}^1_{\Sigma,\mathsf{A}}(\lambda, b)$ be the same as $\mathbf{Tamper}_{\Sigma,\mathsf{A}}(\lambda, b)$, with the following differences.

1. The hybrid records all answers to A's leakage queries. In particular, let τ be the transcript containing all such answers; we can parse τ as $\tau_1, \ldots, \tau_n$, where τ_i is the leakage obtained from the share s_i.
2. When A outputs its tampering query $(\mathcal{T}, (f_1, \ldots, f_n))$:
   (a) For $t \in \mathcal{T}$, compute the value $\tilde{s}_t = ((\tilde{s}''_{i,j,1})_{j\neq i}, (\tilde{s}''_{j,i,2})_{j\neq i}) = f_t(s_t)$, and set $\tau_t \leftarrow \tau_t \| \tilde{s}''_{\mathtt{prv}(t),t,1}$.
   (b) For $t \in \mathcal{T}$, sample $(\hat{s}_{t,j,2})_{j\neq t}$ conditioned on:
      i. $\mathsf{Rec}(s''_{t,j,1}, \hat{s}_{t,j,2}) = s'_t$ for any j, where s'_t is the value computed by the sharing algorithm;
      ii. The values $(\hat{s}_{t,j,2})_{j\neq i}$ being consistent with the leakage transcript τ_t, namely, let $\hat{s}_t := ((s''_{j,i,1})_{j\neq i}, (\hat{s}_{i,j,2})_{j\neq i})$, by applying all the leakage functions to $\hat{s}_t = ((s''_{j,i,1})_{j\neq i}, (\hat{s}_{i,j,2})_{j\neq i})$ we obtain the value τ_t.
   (c) Output $\underline{\mathsf{Rec}((f_t(\hat{s}_t))_{t\in\mathcal{T}})}$ (instead of $\mathsf{Rec}((f_t(s_t))_{t\in\mathcal{T}})$).

Lemma 13. *$\forall b \in \{0,1\}$: $\{\mathbf{Tamper}_{\Sigma,\mathsf{A}}(\lambda, b)\}_{\lambda\in\mathbb{N}} \equiv \{\mathbf{Hyb}^1_{\Sigma,\mathsf{A}}(\lambda, b)\}_{\lambda\in\mathbb{N}}$.*

Proof. Notice that the hybrid samples $(\hat{s}_t)_{t\in\mathcal{T}}$ from exactly the same conditional distribution where the shares $(s_t)_{t\in\mathcal{T}}$ are picked. Thus the output of the tampering oracle is distributed identically in the two experiments.

Formally, let $\mathbf{Z}$ be the random variable describing the tuple $(s'_i, (s''_{i,j,1})_{j\neq i}, \tau_i)_{i\in[n]}$, let $\mathbf{S}$ be the random variable $(s''_{t,j,2})_{t\in\mathcal{T}, j\neq t}$, and let $\hat{\mathbf{S}}$ be the random variable $(\hat{s}_{t,j,2})_{t\in\mathcal{T}, j\neq t}$. We can think of $\mathbf{Z}$ being the output of a randomized function of $\mathbf{S}$: let F be such function, then $\mathbf{Z} = F(\mathbf{S})$. Notice that the two hybrids are equivalent if the distributions $(\mathbf{S}, F(\mathbf{S}))$ and $(\hat{\mathbf{S}}, F(\mathbf{S}))$ are

equivalent. To prove this, we observe that for any assignments z and $s = ((s_{t,j})_{j\neq t})_{t\in\mathcal{T}}$ the following holds:

$$\Pr[\mathbf{S} = s|\mathbf{Z} = z] = \prod_t \mathbb{P}\left[(\mathbf{S}''_{t,j,2})_{j\neq t} = (s_{t,j})_{j\neq t}|\mathbf{Z} = z\right] \tag{7}$$

$$= \prod_t \mathbb{P}\left[(\hat{\mathbf{S}}_{t,j,2})_{j\neq t} = (s_{t,j})_{j\neq t}|\mathbf{Z} = z\right] \tag{8}$$

$$= \Pr[\hat{\mathbf{S}} = s|\mathbf{Z} = z]. \tag{9}$$

Eq. (7) follows because for any $i \in [n]$, and for any assignments of $s' = s'_1, \ldots, s'_n$ and $(s''_{i,j,1})_{j\neq i}$, the random variables $((\mathbf{S}''_{i,j,2})_{j\neq i}|\mathbf{S}' = s', (\mathbf{S}''_{i,j,1})_{j\neq i} = (s''_{i,j,1})_{j\neq i})$ are independent, and thus by Lemma 16 the random variables $((\mathbf{S}''_{i,j,2})_{j\neq i}|\mathbf{Z} = z)$ for any $i \in [n]$ are independent too. Eq. (8) uses the fact that, once we fix $\mathbf{Z}$, for any t the random variables $(\mathbf{S}''_{t,j,2})_{j\neq t}$ and $(\hat{\mathbf{S}}_{t,j,2})_{j\neq i}$ are chosen from the same distribution. Finally, Eq. (9) follows because for any t the random variables $(\hat{\mathbf{S}}_{t,j,2})_{j\neq t}$ are independently sampled. □

Second hybrid. Let $\mathbf{Hyb}^2_{\Sigma,\mathsf{A}}(\lambda, b)$ be the same as $\mathbf{Hyb}^1_{\Sigma,\mathsf{A}}(\lambda, b)$, except that:

1. The hybrid additionally samples for any $i \in [n]$ and $j \in [n]\backslash\{i\}$ the share $s^*_{i,j,2} \leftarrow\!\!\$\ \mathsf{Share}(0^\mu, s''_{i,j,1})$, and sets $s^*_i = ((s''_{j,i,1})_{j\neq i}, (s^*_{i,j,2})_{j\neq i})$.
2. The answers to A's leakage queries are answered using the shares $s^*_1, \ldots, s^*_n$.

Lemma 14. $\forall b \in \{0,1\}$: $\{\mathbf{Hyb}^1_{\Sigma,\mathsf{A}}(\lambda, b)\}_{\lambda\in\mathbb{N}} \approx_s \{\mathbf{Hyb}^2_{\Sigma,\mathsf{A}}(\lambda, b)\}_{\lambda\in\mathbb{N}}$.

Proof. We use a hybrid argument over all indices i, j, where $j \neq i$ and $i, j \in \mathbb{N}$. To simplify notation, let $N = \binom{n}{2}$ and define $\binom{[n]}{2} := \{\mathcal{H}_1, \ldots, \mathcal{H}_N\}$. Set $\mathbf{Hyb}^{(0)}_{\Sigma,\mathsf{A}} := \mathbf{Hyb}^2_{\Sigma,\mathsf{A}}$, and for any $l > 0$ consider the hybrid $\mathbf{Hyb}^{(l)}_{\Sigma,\mathsf{A}}$ to be the same as $\mathbf{Hyb}^{(l-1)}_{\Sigma,\mathsf{A}}$ but where $s^*_{h_1,h_2,2} \leftarrow\!\!\$\ \mathsf{Share}(s'_{h_1}, s''_{h_1,h_2,1})$ for $\mathcal{H}_l = \{h_1, h_2\}$. It is clear that $\mathbf{Hyb}^{(N)}_{\Sigma,\mathsf{A}}$ is distributed exactly as $\mathbf{Hyb}^1_{\Sigma,\mathsf{A}}$. We prove that $\mathbf{Hyb}^{(l)}_{\Sigma,\mathsf{A}}$ and $\mathbf{Hyb}^{(l-1)}_{\Sigma,\mathsf{A}}$ are statistically close via a reduction to leakage resilience of Σ''. Consider the adversary $\mathsf{B} = (\mathsf{B}_1, \mathsf{B}_2, \mathsf{B}_3)$ defined below.

Adversary $\mathsf{B}_1(1^\lambda)$:

1. Sample $(s'_1, \ldots, s'_n) \leftarrow\!\!\$\ \mathsf{Share}'(m_b)$;
2. Return state $\alpha_1 = (s'_1, \ldots, s'_n)$, and challenge messages $(s'_{h_1}, 0^\mu)$.

Adversary $\mathsf{B}_2^{\mathcal{O}_{\mathsf{leak}}((s''_{h_1,h_2,1}, s^*_{h_1,h_2,2}),\cdot)}(\alpha_1)$:

3. Sample $s''_{h'_1,h'_2,1}$ and $s^*_{h'_1,h'_2,2}$ for any $\{h'_1, h'_2\} \neq \mathcal{H}_l$, as described in $\mathbf{Hyb}^{(l)}_{\Sigma,\mathsf{A}}$.
4. Run A and answer to its leakage queries. In particular, upon input $(g_1, \ldots, g_n)$, for any index $k \notin \mathcal{H}_l$ the output of leakage function g_k can be computed without the help of the leakage oracle. Let z_k be this output. Hence:
   - Let $\bar{g}_1(s''_{h_1,h_2,1})$ be the function that first defines s_{h_1} by plugging $s''_{h_1,h_2,1}$ and hard-coding the remaining values, and then outputs $g_{h_1}(s_{h_1})$.
   - Let $\bar{g}_2(s''_{h_1,h_2,2})$ be the function that first defines s_{h_2} by plugging $s''_{h_1,h_2,2}$ and hard-coding the remaining values, and then outputs $g_{h_2}(s_{h_2})$.

   Forward the leakage query $(\bar{g}_1, \bar{g}_2)$ to the challenger, and denote by z_{h_1}, z_{h_2} the corresponding answer. Forward the values $z_1, \ldots, z_n$ to A, and update the transcript $\tau \leftarrow \tau\|(z_1, \ldots, z_n)$.

5. Eventually the adversary A outputs a tampering query $(\mathcal{T}, (f_1, \ldots, f_n))$
6. Consider the leakage function $g_2^*(s''_{h_1,h_2,2})$ that first defines s_{h_2} by plugging $s''_{h_1,h_2,2}$ and hard-coding the remaining values, then lets $\tilde{s}_{h_2} = f_{h_2}(s_{h_2})$, and finally outputs $\tilde{s}''_{\mathtt{prv}(h_2),h_2,1}$. Forward the leakage query (ε, g_2^*) to the challenger, and let $(\varepsilon, \tilde{s}''_{\mathtt{prv}(h_2),h_2,1})$ be the corresponding answer. Set $\tau_{h_2} \leftarrow \tau_{h_2} \| \tilde{s}''_{\mathtt{prv}(h_2),h_2,1}$.
7. Let α_2 be the full state of B up to this point, including the transcript τ, and the state α_{A} of attacker A.

Adversary $\mathsf{B}_3(\alpha_2, s''_{h_1,h_2,1})$:

8. Compute $\tilde{s}''_{\mathtt{prv}(t),t,1}$ for $t \in \mathcal{T} \setminus \{h_2\}$, as described in $\mathbf{Hyb}^1_{\Sigma,\mathsf{A}}$; in particular, $\tau_t \leftarrow \tau_t \| \tilde{s}''_{\mathtt{prv}(t),t,1}$.
9. Sample $(\hat{s}_{t,j,2})_{j \neq t}$ for $t \in \mathcal{T}$, conditioned on the knowledge of $s'_t, (s''_{t,j,1})_{j \neq t}$ and τ, as described in $\mathbf{Hyb}^1_{\Sigma,\mathsf{A}}$.
10. Set $\hat{s}_t$ for $t \in \mathcal{T}$, as described in step 2b of $\mathbf{Hyb}^1_{\Sigma,\mathsf{A}}$, and compute the tampered message $\tilde{m}$ using these shares.
11. Resume A using α_{A}, forward to A the tampered message $\tilde{m}$, and output whatever A does.

Claim 1. *The following holds $\forall b \in \{0,1\}$:*

$$\{\mathbf{Leak}_{\Sigma'',\mathsf{B}}(\lambda, 0)\}_{\lambda \in \mathbb{N}} \equiv \{\mathbf{Hyb}^{(l-1)}_{\Sigma,\mathsf{A}}(\lambda, b)\}_{\lambda \in \mathbb{N}}$$
$$\{\mathbf{Leak}_{\Sigma'',\mathsf{B}}(\lambda, 1)\}_{\lambda \in \mathbb{N}} \equiv \{\mathbf{Hyb}^{(l)}_{\Sigma,\mathsf{A}}(\lambda, b)\}_{\lambda \in \mathbb{N}}.$$

Proof. The proof of the claim holds by inspection of adversary B. □

Claim 2. *If Σ'' has α-correlated shares and adversary A is ℓ-NLA, then adversary B is $(\ell + (2n-3)\alpha, \ell + (2n-3)\alpha + \sigma_1)$-NLA.*

Proof. We analyze the leakage performed by B_2 on $s''_{h_1,h_2,1}$ and $s^*_{h_1,h_2,2}$. Let $\mathbf{S}''_1$ be the random variable describing the value $s''_{h_1,h_2,1}$, let $\mathbf{S}''_2$ be the random variable describing the value $s''_{h_1,h_2,2}$, let $g^{(q)} := (g_1^{(q)}, \ldots, \bar{g}_n^{(q)})$ be the q-th leakage leakage query output by A, and $(\bar{g}_1^{(q)}, \bar{g}_2^{(q)})$ be the q-th leakage query output by B in the reduction. For each $a \in \{1,2\}$, let $\mathbf{\Lambda}_a := (\bar{g}_a^{(1)}(\mathbf{S}''_a), \ldots, \bar{g}_a^{(p)}(\mathbf{S}''_a))$ be the complete transcript containing all answers to leakage queries, and let $\mathbf{\Lambda}^*_2 := (\mathbf{\Lambda}_2, g_2^*(\mathbf{S}''_2))$. Finally, for $k \in [n]$, let $\mathbf{S}_k$ be the random variable describing the share s^*_k as sampled by B, and let $\mathbf{S}^*_{h_1}$ (resp. $\mathbf{S}^*_{h_2}$) be the part of share $s^*_{h_1}$ (resp. $s^*_{h_2}$) which does not include $s^*_{h_1,h_2,2}$ (resp. $s''_{h_1,h_2,1}$). First, observe that:

$$\widetilde{\mathbb{H}}_\infty(\mathbf{S}''_2 \mid \mathbf{S}''_1,\ \mathbf{\Lambda}^*_1) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{S}''_2 \mid \mathbf{S}''_1,\ \mathbf{\Lambda}_1) - \sigma_1, \tag{10}$$

which holds by Lemma 17. Moreover, for $a \in \{1,2\}$, the following holds:

$$\begin{aligned}
&\widetilde{\mathbb{H}}_\infty(\mathbf{S}''_a \mid \mathbf{S}''_{3-a},\ \mathbf{\Lambda}_a) \\
&\geq \widetilde{\mathbb{H}}_\infty(\mathbf{S}''_a \mid \mathbf{S}''_{3-a},\ \mathbf{S}^*_{h_{3-a}}, (\mathbf{S}_k)_{k \in [n] \setminus \mathcal{H}_l},\ \mathbf{\Lambda}_a) \\
&= \widetilde{\mathbb{H}}_\infty(\mathbf{S}''_a \mid (\mathbf{S}_k)_{k \neq h_a},\ \mathbf{\Lambda}_a) \\
&\geq \widetilde{\mathbb{H}}_\infty(\mathbf{S}''_a, \mathbf{S}^*_{h_a} \mid (\mathbf{S}_k)_{k \neq h_a},\ \mathbf{\Lambda}_a) - ((n-1)\sigma_{3-a} + (n-2)\sigma_a) \qquad (11) \\
&\geq \widetilde{\mathbb{H}}_\infty(\mathbf{S}''_a, \mathbf{S}^*_{h_a} \mid (\mathbf{S}_k)_{k \neq h_a}) - ((n-1)\sigma_{3-a} + (n-2)\sigma_a) - \ell. \qquad (12)
\end{aligned}$$

Eq. (11) follows by Lemma 17, whereas Eq. (12) follows by admissibility of A. For any $k \neq h_a$, we can split $\mathbf{S}_k$ into two pieces: the random variable $\mathbf{S}^{(1)}_k$ corresponding $(s''_{k,h_a,1}, s''_{h_a,k,2})$, and

the random variable $\mathbf{S}_k^{(2)}$ corresponding to $(s''_{k,j,1})_{j\neq k, j\neq h_a}, (s^*_{j,k,2})_{j\neq k, j\neq h_a}$. Recall that Σ has α-correlated shares; for convenience, we define $\tilde{\sigma}_i = \sigma_i - \alpha$. Let $\mathbf{S}'$ be the random variable associated to the output of $\mathsf{Share}'(m)$. For any assignments of $s' = (s'_1, \ldots, s'_n) \leftarrow\!\!\$\ \mathsf{Share}(m)$, we can write:

$$\begin{aligned}
&\widetilde{\mathbb{H}}_\infty(\mathbf{S}''_a, \mathbf{S}^*_{h_a} \mid (\mathbf{S}_k^{(1)}, \mathbf{S}_k^{(2)})_{k\neq h_a},\ \mathbf{S}' = s') \\
&\qquad = \widetilde{\mathbb{H}}_\infty(\mathbf{S}''_a, \mathbf{S}^*_{h_a} \mid (\mathbf{S}_k^{(1)})_{k\neq h_a},\ \mathbf{S}' = s') &(13)\\
&\qquad \geq \widetilde{\mathbb{H}}_\infty(\mathbf{S}''_a \mid \mathbf{S}''_{3-a},\ \mathbf{S}' = s') + (n-1)\tilde{\sigma}_{3-a} + (n-2)\tilde{\sigma}_a. &(14)
\end{aligned}$$

Eq. (13) follows because, once the value s' is fixed, then the random variables $\mathbf{S}_k^{(2)}$ are independent, whereas Eq. (14) follows by Lemma 12. Finally, by averaging over all possible assignments of s', we get:

$$\widetilde{\mathbb{H}}_\infty(\mathbf{S}''_a, \mathbf{S}^*_{h_a} \mid (\mathbf{S}_k^{(1)}, \mathbf{S}_k^{(2)})_{k\neq h_a}) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{S}''_a \mid \mathbf{S}''_{3-a}) + (n-1)\tilde{\sigma}_{3-a} + (n-2)\tilde{\sigma}_a,$$

which together with Eq. (12) and Eq. (10) proves the claim. □

□

Lemma 15. $\{\mathbf{Hyb}^2_{\Sigma,\mathsf{A}}(\lambda, 0)\}_{\lambda\in\mathbb{N}} \approx_s \{\mathbf{Hyb}^2_{\Sigma,\mathsf{A}}(\lambda, 1)\}_{\lambda\in\mathbb{N}}$.

Proof. Consider the following adversary B against the non-malleability of Σ' given black-box access to an attacker A playing in $\mathbf{Hyb}^2_{\Sigma,\mathsf{A}}(\lambda, b)$.

1. At the beginning B runs $\mathsf{A}_1(1^\lambda)$, and outputs the same values $(m_0, m_1, \alpha_\mathsf{A})$ returned by A_1.
2. Sample $s''_{i,j,1} \leftarrow\!\!\$\ \mathcal{S}''_1$ and $s^*_{i,j,2} \leftarrow\!\!\$\ \mathsf{Share}(0^\mu, s''_{i,j,1})$, for $i, j \in [n]$ and $i \neq j$, and set $s^*_i \leftarrow ((s''_{j,i,1})_{j\neq i}, (s^*_{i,j,2})_{j\neq i})$.
3. Resume the adversary A by running $\mathsf{A}_2(\alpha_\mathsf{A})$, and reply to its leakage queries using the shares $s^*_1, \ldots, s^*_n$.
4. Eventually, A_2 outputs $(\mathcal{T}, (f_1, \ldots, f_n))$. Sample the values $\tilde{s}''_{\mathtt{prv}(t),t,1}$ for $t \in \mathcal{T}$, as described in $\mathbf{Hyb}^1_{\Sigma,\mathsf{A}}$, and consider the tampering function $f'_t(s'_t)$ that samples $(\hat{s}_{t,j,2})_{j\neq t}$ conditioned on: (i) $\mathsf{Rec}(s_{i,j,1}, \hat{s}_{i,j,2}) = s'_i$ for all $j \neq i$; and (ii) the leakage transcript $\tau_t \| \tilde{s}''_{\mathtt{prv}(t),t,1}$ being consistent with $(\hat{s}_{i,j,2})_{j\neq i}$. The function sets $\hat{s}_i := ((s''_{j,i,1})_{j\neq i}, (\hat{s}_{i,j,2})_{j\neq i})$, computes $\tilde{s}_i \leftarrow f(\hat{s}_i)$, and outputs $\mathsf{Rec}(\tilde{s}''_{i,\mathtt{nxt}(i),1}, \tilde{s}''_{i,\mathtt{nxt}(i),2})$.
5. Forward $(\mathcal{T}, (f'_1, \ldots, f'_n))$ to the challenger, and send the answer back to A.
6. Output whatever A outputs.

We claim that for any $b \in \{0, 1\}$, we have $\mathbf{Tamper}_{\Sigma',\mathsf{B}}(\lambda, b) \equiv \mathbf{Hyb}^2_{\Sigma,\mathsf{A}}(\lambda, b)$. In fact, the adversary B runs exactly as in $\mathbf{Hyb}^2_{\Sigma,\mathsf{A}}(\lambda, b)$. The only (syntactical) difference is that B samples the values $(\hat{s}_{i,j,2})_{j\neq i}$ and uses the target tampering oracle to answer A's tampering query. The lemma then follows by statistical non-malleability of Σ'. □

The statement of the theorem follows by putting together the above lemmas, and applying the triangular inequality. □

7 Conclusions and Open Problems

We have shown new constructions of leakage-resilient continuously non-malleable secret sharing schemes, for general access structures. Our first scheme is in the plain model, and guarantees security against noisy leakage and independent tampering with all of the shares. Our second scheme is in the CRS model, and guarantees security against bounded leakage and joint tampering with a fixed partition of the n shares into non-overlapping subsets of size $O(\log n)$.

The two major questions left over by our work are whether continuous non-malleability against joint tampering is achievable in the plain model, or against adaptive (rather than selective) joint tampering with the shares. Interestingly, our proof strategy breaks down in the case of adaptive tampering, and this holds true even assuming that the inner leakage-resilient secret sharing is secure in the presence of adaptive joint leakage. Intuitively, the reason is that in the reduction we must run different copies of the adversary inside the leakage oracle; in particular, we use a fixed subset of the shares in order to simulate the answer to all tampering queries asked by each copy of the attacker, and this is clearly possible only if the adversary does not change the partition within each query.

It would also be interesting to achieve continuous non-malleability under joint selective partitioning for better values of the parameter k (namely, the attacker can tamper jointly with subsets of size super-logarithmic in n). Note that this would follow immediately by our result if we plug in our construction a leakage-resilient secret sharing scheme tolerating joint leakage from subsets of shares with size more than $O(\log n)$. Unfortunately, the only known such scheme is the one in [KMS18], and as the authors explain improving the parameters in their construction would lead to progress on longstanding open problems in complexity theory. We leave it open to establish whether this holds true even in the case of selective partitioning (recall that the scheme of [KMS18] achieves adaptive leakage resilience), or whether the current state of affairs can be improved in the computational setting (with or without trusted setup).

A further open question is to improve the rate of our constructions. Note that by applying the rate compiler of [FV19], we do get *rate-one* continuously non-malleable secret sharing for general access structures, against independent tampering in the plain model. However, this is well-known to be sub-optimal in the computational setting, where the optimal share size would be $O(\mu/n)$, where μ is the size of the message [Kra93]. Note that it is unclear whether the same rate compiler works also for our construction against joint tampering under selective partitioning. This is because the analysis in [FV19] crucially relies on the resilience of the initial rate-zero non-malleable secret sharing against noisy leakage, whereas our construction only achieves security in the bounded-leakage model.

References

[AAG+16] Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In *TCC*, pages 393–417, 2016.

[ADKO15a] Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In *STOC*, pages 459–468, 2015.

[ADKO15b] Divesh Aggarwal, Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Leakage-resilient non-malleable codes. In *TCC*, pages 398–426, 2015.

[ADL14] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In *STOC*, pages 774–783, 2014.

[ADN+18] Divesh Aggarwal, Ivan Damgaard, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, Joao Ribeiro, and Mark Simkin. Stronger leakage-resilient and non-malleable secret-sharing schemes for general access structures. Cryptology ePrint Archive, Report 2018/1147, 2018. https://ia.cr/2018/1147.

[BGW19] Marshall Ball, Siyao Guo, and Daniel Wichs. Non-malleable codes for decision trees. Cryptology ePrint Archive, Report 2019/379, 2019. https://eprint.iacr.org/2019/379.

[BS18] Saikrishna Badrinarayanan and Akshayaram Srinivasan. Revisiting non-malleable secret sharing. Cryptology ePrint Archive, Report 2018/1144, 2018. https://ia.cr/2018/1144.

[CFV19] Sandro Coretti, Antonio Faonio, and Daniele Venturi. Rate-optimizing compilers for continuously non-malleable codes. Cryptology ePrint Archive, Report 2019/055, 2019. https://ia.cr/2019/055.

[CG88] Benny Chor and Oded Goldreich. Unbiased bits from sources of weak randomness and probabilistic communication complexity. *SIAM J. Comput.*, 17(2):230–261, 1988.

[CG14] Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In *TCC*, pages 440–464, 2014.

[CL18] Eshan Chattopadhyay and Xin Li. Non-malleable codes, extractors and secret sharing for interleaved tampering and composition of tampering. Cryptology ePrint Archive, Report 2018/1069, 2018. https://eprint.iacr.org/2018/1069.

[DDV10] Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In *SCN*, pages 121–137, 2010.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Efficient public-key cryptography in the presence of key leakage. In *ASIACRYPT*, pages 613–631, 2010.

[DKO13] Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In *CRYPTO*, pages 239–257, 2013.

[DORS08] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam D. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM J. Comput.*, 38(1):97–139, 2008.

[DP07] Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *FOCS*, pages 227–237, 2007.

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *Innovations in Computer Science*, pages 434–452, 2010.

[FLS90] Uriel Feige, Dror Lapidot, and Adi Shamir. Multiple non-interactive zero knowledge proofs based on a single random string (extended abstract). In *FOCS*, pages 308–317, 1990.

[FMNV14] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In *TCC*, pages 465–488, 2014.

[FNSV18] Antonio Faonio, Jesper Buus Nielsen, Mark Simkin, and Daniele Venturi. Continuously non-malleable codes with split-state refresh. In *ACNS*, pages 1–19, 2018.

[FV19] Antonio Faonio and Daniele Venturi. Non-malleable secret sharing in the computational setting: Adaptive tampering, noisy-leakage resilience, and improved rate. *IACR Cryptology ePrint Archive*, 2019:105, 2019.

[GK18a] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In *STOC*, pages 685–698, 2018.

[GK18b] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing for general access structures. In *CRYPTO*, pages 501–530, 2018.

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In *STOC*, pages 218–229, 1987.

[GPR16] Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In *STOC*, pages 1128–1141, 2016.

[Gro06] Jens Groth. Simulation-sound NIZK proofs for a practical language and constant size group signatures. In *ASIACRYPT*, pages 444–459, 2006.

[HM96] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In *CRYPTO*, pages 201–215, 1996.

[KMS18] Ashutosh Kumar, Raghu Meka, and Amit Sahai. Leakage-resilient secret sharing. Cryptology ePrint Archive, Report 2018/1138, 2018. https://ia.cr/2018/1138.

[Kra93] Hugo Krawczyk. Secret sharing made short. In *CRYPTO*, pages 136–146, 1993.

[LCG+19] Fuchun Lin, Mahdi Cheraghchi, Venkatesan Guruswami, Reihaneh Safavi-Naini, and Huaxiong Wang. Non-malleable secret sharing against affine tampering. *CoRR*, abs/1902.06195, 2019.

[Li17] Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In *STOC*, pages 1144–1156, 2017.

[LL12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In *CRYPTO*, pages 517–532, 2012.

[NS19] Jesper Buus Nielsen and Mark Simkin. Lower bounds for leakage-resilient secret sharing. Cryptology ePrint Archive, Report 2019/181, 2019. https://eprint.iacr.org/2019/181.

[OPVV18] Rafail Ostrovsky, Giuseppe Persiano, Daniele Venturi, and Ivan Visconti. Continuously non-malleable codes in the split-state model from minimal assumptions. In *CRYPTO*, pages 608–639, 2018.

[SCO+01] Alfredo De Santis, Giovanni Di Crescenzo, Rafail Ostrovsky, Giuseppe Persiano, and Amit Sahai. Robust non-interactive zero knowledge. In *CRYPTO*, pages 566–598, 2001.

[SV18] Akshayaram Srinivasan and Prashant Nalini Vasudevan. Leakage resilient secret sharing and applications. Cryptology ePrint Archive, Report 2018/1154, 2018. https://ia.cr/2018/1154.

A Standard Definitions

Basic notation. For a string x, we denote its length by $|x|$; if $\mathcal{X}$ is a set, $|\mathcal{X}|$ represents the number of elements in $\mathcal{X}$. When x is chosen randomly in $\mathcal{X}$, we write $x \leftarrow\!\!\$\, \mathcal{X}$. When A is a randomized algorithm, we write $y \leftarrow\!\!\$\, \mathsf{A}(x)$ to denote a run of A on input x (and implicit random coins r) and output y; the value y is a random variable, and $\mathsf{A}(x;r)$ denotes a run of A on input x and randomness r. An algorithm A is *probabilistic polynomial-time* (PPT) if A is randomized and for any input $x, r \in \{0,1\}^*$ the computation of $\mathsf{A}(x;r)$ terminates in a polynomial number of steps (in the size of the input).

Negligible functions. We denote with $\lambda \in \mathbb{N}$ the security parameter. A function p is a polynomial, denoted $p(\lambda) \in \mathtt{poly}(\lambda)$, if $p(\lambda) \in O(\lambda^c)$ for some constant $c > 0$. A function $\nu : \mathbb{N} \to [0,1]$ is negligible in the security parameter (or simply negligible) if it vanishes faster than the inverse of any polynomial in λ, i.e. $\nu(\lambda) \in O(1/p(\lambda))$ for all positive polynomials $p(\lambda)$. We often write $\nu(\lambda) \in \mathtt{negl}(\lambda)$ to denote that $\nu(\lambda)$ is negligible.

Unless stated otherwise, throughout the paper, we implicitly assume that the security parameter is given as input (in unary) to all algorithms.

Random variables. For a random variable $\mathbf{X}$, we write $\mathbb{P}[\mathbf{X} = x]$ for the probability that $\mathbf{X}$ takes on a particular value $x \in \mathcal{X}$ (with $\mathcal{X}$ being the set where $\mathbf{X}$ is defined). The statistical distance between two random variables $\mathbf{X}$ and $\mathbf{X}'$ defined over the same set $\mathcal{X}$ is defined as $\mathbb{SD}(\mathbf{X};\mathbf{X}') = \frac{1}{2}\sum_{x\in\mathcal{X}} |\mathbb{P}[\mathbf{X} = x] - \mathbb{P}[\mathbf{X}' = x]|$.

Given two ensembles $\mathbf{X} = \{\mathbf{X}_\lambda\}_{\lambda\in\mathbb{N}}$ and $\mathbf{Y} = \{\mathbf{Y}_\lambda\}_{\lambda\in\mathbb{N}}$, we write $\mathbf{X} \equiv \mathbf{Y}$ to denote that they are identically distributed, $\mathbf{X} \approx_s \mathbf{Y}$ to denote that they are statistically close, i.e. $\mathbb{SD}(\mathbf{X}_\lambda;\mathbf{X}'_\lambda) \in \mathtt{negl}(\lambda)$, and $\mathbf{X} \approx_c \mathbf{Y}$ to denote that they are computationally indistinguishable, i.e., for all PPT distinguishers D:

$$|\mathbb{P}[\mathsf{D}(\mathbf{X}_\lambda) = 1] - \mathbb{P}[\mathsf{D}(\mathbf{Y}_\lambda) = 1]| \in \mathtt{negl}(\lambda).$$

We extend the notion of computational indistinguishability to the case of interactive experiments (a.k.a. games) featuring an adversary A. In particular, let $\mathbf{G}_\mathsf{A}(\lambda)$ be the random variable corresponding to the output of A at the end of the experiment, where wlog. we may assume A outputs a decision bit. Given two experiments $\mathbf{G}_\mathsf{A}(\lambda,0)$ and $\mathbf{G}_\mathsf{A}(\lambda,1)$, we write $\{\mathbf{G}_\mathsf{A}(\lambda,0)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{G}_\mathsf{A}(\lambda,1)\}_{\lambda\in\mathbb{N}}$ as a shorthand for

$$|\mathbb{P}[\mathbf{G}_\mathsf{A}(\lambda,0) = 1] - \mathbb{P}[\mathbf{G}_\mathsf{A}(\lambda,1) = 1]| \in \mathtt{negl}(\lambda).$$

The above naturally generalizes to statistical distance (in case of unbounded adversaries). We recall a useful lemma from [DP07, DDV10].

Lemma 16 ([DDV10], Lemma 4)**.** *Let $\mathcal{O}_{\mathsf{leak}}(x,g)$ be an oracle that upon input a value x and a function g outputs $g(x)$, and let $\mathbf{X}$ and $\mathbf{Y}$ be two independently distributed random variables. For any adversary A, and for any value z, the distributions $\left(\mathbf{X}|z = \mathsf{A}^{\mathcal{O}_{\mathsf{leak}}(\mathbf{X},\cdot),\mathcal{O}_{\mathsf{leak}}(\mathbf{Y},\cdot)}\right)$ and $\left(\mathbf{Y}|z = \mathsf{A}^{\mathcal{O}_{\mathsf{leak}}(\mathbf{X},\cdot),\mathcal{O}_{\mathsf{leak}}(\mathbf{Y},\cdot)}\right)$ are independently distributed.*

Average min-entropy. The min-entropy of a random variable $\mathbf{X}$ with domain $\mathcal{X}$ is $\mathbb{H}_\infty(\mathbf{X}) := -\log \max_{x\in\mathcal{X}} \mathbb{P}[\mathbf{X} = x]$, and intuitively it measures the best chance to predict $\mathbf{X}$ (by a computationally unbounded algorithm). For conditional distributions, unpredictability is measured by the conditional average min-entropy [DORS08]: $\widetilde{\mathbb{H}}_\infty(\mathbf{X}|\mathbf{Y}) := -\log \mathbb{E}_y[2^{-\mathbb{H}_\infty(\mathbf{X}|\mathbf{Y}=y)}]$. The lemma below is sometimes known as the "chain rule" for conditional average min-entropy.

Lemma 17 ([DORS08], Lemma 2.2)**.** *Let* $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ *be random variables. If* $\mathbf{Y}$ *has at most* 2^ℓ *possible values, then* $\widetilde{\mathbb{H}}_\infty(\mathbf{X}|\mathbf{Y}, \mathbf{Z}) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{X}, \mathbf{Y}|\mathbf{Z}) - \ell \geq \widetilde{\mathbb{H}}_\infty(\mathbf{X}|\mathbf{Z}) - \ell$*. In particular,* $\widetilde{\mathbb{H}}_\infty(\mathbf{X}|\mathbf{Y}) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{X}, \mathbf{Y}) - \ell \geq \widetilde{\mathbb{H}}_\infty(\mathbf{X}) - \ell$*.*

A.1 Randomness Extractors

An (average-case, strong) seeded extractor is a polynomial-time deterministic algorithm $\mathsf{Ext} : \{0,1\}^d \times \{0,1\}^\rho \to \{0,1\}^\alpha$ taking as input a source x and a seed r, and outputting an α-bit string.

Definition 9 (Seeded extractors)**.** Let $\kappa \in \mathbb{N}$ and $\epsilon \in [0,1]$. We say that Ext is (κ, ϵ)-secure, if for any random variable $\mathbf{\Lambda}$ such that $\widetilde{\mathbb{H}}_\infty(\mathbf{X}|\mathbf{\Lambda}) \geq \kappa$ the statistical distance between the distribution $(\mathbf{R}, \mathsf{Ext}(\mathbf{X}, \mathbf{R}), \mathbf{\Lambda})$ and $(\mathbf{R}, \mathbf{A}, \mathbf{\Lambda})$ is at most ϵ, where $\mathbf{R} \equiv \mathbf{U}_\rho$ and $\mathbf{A} \equiv \mathbf{U}_\alpha$.

An (average-case) two-source extractor is a deterministic polynomial-time algorithm $\mathsf{2Ext} : \{0,1\}^d \times \{0,1\}^d \to \{0,1\}^\alpha$ taking as input two sources x and y and outputting an α-bit string.

Definition 10 (Two-source extractors)**.** Let $\kappa_1, \kappa_2 \in \mathbb{N}$ and $\epsilon \in [0,1]$. We say that $\mathsf{2Ext}$ is $(\kappa_1, \kappa_2, \epsilon)$-secure if for any random variable $\mathbf{\Lambda}$ such that $\widetilde{\mathbb{H}}_\infty(\mathbf{X}|\mathbf{\Lambda}) \geq \kappa_1$, $\widetilde{\mathbb{H}}_\infty(\mathbf{Y}|\mathbf{\Lambda}) \geq \kappa_2$, and $(\mathbf{X}|\mathbf{\Lambda}), (\mathbf{Y}|\mathbf{\Lambda})$ are independent, the statistical distance between the distribution $(\mathbf{X}, \mathsf{2Ext}(\mathbf{X}, \mathbf{Y}), \mathbf{\Lambda})$ and $(\mathbf{X}, \mathbf{A}, \mathbf{\Lambda})$ is at most ϵ, where $\mathbf{A} \equiv \mathbf{U}_\alpha$.

A.2 Non-Interactive Commitments

A non-interactive commitment scheme $\Pi = (\mathsf{Gen}, \mathsf{Com})$ is a pair of polynomial-time algorithms specified as follows: (i) The randomized algorithm Gen takes as input 1^λ and outputs a public key $pk \in \mathcal{K}$; (ii) The randomized algorithm Com takes as input the public key pk and a message $m \in \mathcal{M}$, and outputs a commitment $c = \mathsf{Com}(pk, m; r) \in \mathcal{C}$ using random coins $r \in \mathcal{R}$. The pair (m, r) is called the opening. In the plain model, we omit the algorithm Gen and simply set $pk = 1^\lambda$.

Intuitively, a secure commitment satisfies two properties called binding and hiding. The first property says that it is hard to open a commitment in two different ways. The second property says that a commitment hides the underlying message. The formal definitions follow.

Definition 11 (Binding)**.** We say that a non-interactive commitment scheme $\Pi = (\mathsf{Gen}, \mathsf{Com})$ is *computationally binding* if the following probability is negligible for all PPT adversaries A:

$$\mathbb{P}\left[m_0 \neq m_1 \wedge \mathsf{Com}(pk, m_0; r_0) = \mathsf{Com}(pk, m_1; r_1) : \begin{array}{c} pk \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda) \\ (m_0, r_0, m_1, r_1) \leftarrow\!\!\$\ \mathsf{A}(pk) \end{array}\right].$$

In case the above definition holds for all unbounded adversaries, we say that Π is *statistically binding*. Finally, in case the above probability is exactly 0 (i.e., each commitment can be opened to at most a single message), then we say that Π is *perfectly binding*.

Definition 12 (Hiding)**.** We say that a non-interactive commitment scheme $\Pi = (\mathsf{Gen}, \mathsf{Com})$ is *computationally hiding* if the following holds for all PPT adversaries A:

$$\left\{\begin{array}{c} pk \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda); (m_0, m_1, \alpha_1) \leftarrow\!\!\$\ \mathsf{A}_1(pk) \\ c \leftarrow\!\!\$\ \mathsf{Com}(pk, m_0); b' \leftarrow\!\!\$\ \mathsf{A}_2(\alpha_1, c) \end{array}\right\} \approx_c \left\{\begin{array}{c} pk \leftarrow\!\!\$\ \mathsf{Gen}(1^\lambda); (m_0, m_1, \alpha_1) \leftarrow\!\!\$\ \mathsf{A}_1(pk) \\ c \leftarrow\!\!\$\ \mathsf{Com}(pk, m_1); b' \leftarrow\!\!\$\ \mathsf{A}_2(\alpha_1, c) \end{array}\right\}.$$

In case the above ensembles are statistically close (resp. identically distributed), we speak of *statistical* (resp. *perfect*) hiding.

Note that in the plain model the above definition of hiding is equivalent to saying that for all pairs of messages $m_0, m_1 \in \mathcal{M}$ the following holds:

$$\left\{c : \ c \leftarrow\!\!\$\ \mathsf{Com}(1^\lambda, m_0)\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{c : \ c \leftarrow\!\!\$\ \mathsf{Com}(1^\lambda, m_1)\right\}_{\lambda\in\mathbb{N}}.$$

A.3 Non-Interactive Zero Knowledge

Let R be a relation, corresponding to an NP language $\mathcal{L}$. A non-interactive zero-knowledge (NIZK) proof system for R is a tuple of efficient algorithms $\Pi = (\mathsf{CRSGen}, \mathsf{Prove}, \mathsf{Ver})$ specified as follows. (i) The randomized algorithm CRSGen takes as input the security parameter and outputs a common reference string ω; (ii) The randomized algorithm $\mathsf{Prove}(\omega, \phi, (x, w))$, given $(x, w) \in \mathsf{R}$ and a label $\phi \in \{0,1\}^*$, outputs a proof π; (iii) The deterministic algorithm $\mathsf{Ver}(\omega, \phi, (x, \pi))$, given an instance x, a proof π, and a label $\phi \in \{0,1\}^*$, outputs either 0 (for "reject") or 1 (for "accept"). We say that a NIZK for relation R is *correct* if for every $\lambda \in \mathbb{N}$, all ω as output by $\mathsf{Init}(1^\lambda)$, any label $\phi \in \{0,1\}^*$, and any $(x, w) \in \mathsf{R}$, we have that $\mathsf{Ver}(\omega, \phi, (x, \mathsf{Prove}(\omega, \phi, (x, w)))) = 1$.

We define two properties of a NIZK proof system. The first property says that honest proofs do not reveal anything beyond the fact that $x \in \mathcal{L}$.

Definition 13 (Adaptive multi-theorem zero-knowledge)**.** A NIZK with labels Π for a relation R satisfies adaptive multi-theorem zero-knowledge if there exists a PPT simulator $\mathsf{S} := (\mathsf{S}_0, \mathsf{S}_1)$ such that the following holds:

(i) S_0 outputs ω, a simulation trapdoor ζ and an extraction trapdoor ξ.
(ii) For all PPT distinguishers D, we have that

$$\Big|\mathbb{P}\left[\mathsf{D}^{\mathsf{Prove}(\omega,\cdot,(\cdot,\cdot))}(\omega) = 1 : \ \omega \leftarrow\!\!\$\ \mathsf{Init}(1^\lambda)\right] - \mathbb{P}\left[\mathsf{D}^{\mathcal{O}_{\mathsf{sim}}(\zeta,\cdot,\cdot,\cdot)}(\omega) = 1 : \ (\omega,\zeta) \leftarrow\!\!\$\ \mathsf{S}_0(1^\lambda)\right]\Big|$$

is negligible in λ, where the oracle $\mathcal{O}_{\mathsf{sim}}(\zeta, \cdot, \cdot, \cdot)$ takes as input a tuple (ϕ, x, w) and returns $\mathsf{S}_1(\zeta, \phi, x)$ iff $\mathsf{R}(x, w) = 1$ (and otherwise it returns $\perp$).

Groth [Gro06] introduced the concept of simulation-extractable NIZK, which informally states that knowledge soundness should hold even if the adversary can see simulated proofs for possibly false statements of its choice. For our purpose, it will suffice to consider the weaker notion of true simulation extractability, as defined by Dodis *et al.* [DHLW10].

Definition 14 (True simulation extractability)**.** Let Π be a NIZK proof systems for a relation R, that satisfies adaptive multi-theorem zero-knowledge w.r.t. a simulator $\mathsf{S} := (\mathsf{S}_0, \mathsf{S}_1)$. We say that Π is *true simulation extractable* if there exists a PPT algorithm K such that every PPT adversary A has a negligible probability of winning in the following game:

- The challenger runs $(\omega, \zeta, \xi) \leftarrow\!\!\$\ \mathsf{S}_0(1^\lambda)$, and gives ω to A.
- Adversary A can ask polynomially many queries of the form (ϕ, x, w), upon which the challenger returns $\mathsf{S}_1(\zeta, \phi, x)$ if $(x, w) \in \mathsf{R}$ and $\perp$ otherwise.
- Adversary A outputs a tuple (ϕ^*, x^*, π^*).
- The challenger runs $w \leftarrow\!\!\$\ \mathsf{K}(\xi, \phi^*, (x^*, \pi^*))$.

We say that A wins iff: (a) (ϕ^*, x^*) was not queried in the second step; (b) $\mathsf{Ver}(\omega, \phi^*, (x^*, \pi^*)) = 1$; (c) $(x^*, w) \notin \mathsf{R}$.

B Construction of Asymmetric Secret Sharing

We show that the construction of [BGW19] is secure in the noisy-leakage model. Let Ext be a seeded extractor with d-bit source, ρ-bit seed, and μ-bit output (cf. Def. 9 in §A.1). Let $\mathsf{2Ext}$ be a two-source extractor with σ_1-bit source and ρ-bit output (cf. Def. 10 in §A.1). Consider the following 2-out-of-2 secret sharing scheme $\Sigma = (\mathsf{Share}, \mathsf{Rec})$, with $\mathcal{S}_1 = \{0,1\}^{\sigma_1}$, $\mathcal{S}_2 = \{0,1\}^{d+\sigma_1+\mu}$, and $\mathcal{M} = \{0,1\}^{\mu}$.

- Algorithm $\mathsf{Share}(m)$ samples $s_1 \leftarrow\!\!\$\ \{0,1\}^{\sigma_1}$, $x \leftarrow\!\!\$\ \{0,1\}^{d}$, $y \leftarrow\!\!\$\ \{0,1\}^{\sigma_1}$, computes $r = \mathsf{2Ext}(s_1, y)$ and $z = \mathsf{Ext}(x, r) \oplus m$, and finally outputs s_1 and $s_2 = (x, y, z)$;
- Algorithm $\mathsf{Rec}(s_1, s_2)$ parses $s_2 = (x, y, z)$, and returns $z \oplus \mathsf{Ext}(x, \mathsf{2Ext}(s_1, y))$.

Claim 3. *The above secret sharing scheme Σ is $(\alpha, \sigma_1, \sigma_2)$-asymmetric, for $\alpha = \mu$, and $\sigma_2 = d + \sigma_1 + \mu$.*

Proof. The size of the shares follows by inspection. Moreover, the share s_1 is sampled uniformly at random and used as the first source for the two-source extractor. As for the conditional min-entropy, we have:

$$\widetilde{\mathbb{H}}_\infty(\mathbf{S}_1|\mathbf{S}_2) = \widetilde{\mathbb{H}}_\infty((\mathbf{X},\mathbf{Y},\mathbf{Z})|\mathbf{S}_1) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{X},\mathbf{Y}|\mathbf{S}_1) \geq |\mathbf{X}| + |\mathbf{Y}| = \sigma_2 - \mu$$
$$\widetilde{\mathbb{H}}_\infty(\mathbf{S}_2|\mathbf{S}_1) = \widetilde{\mathbb{H}}_\infty(\mathbf{S}_1|(\mathbf{X},\mathbf{Y},\mathbf{Z})) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{S}_1|\mathbf{X},\mathbf{Y}) - |\mathbf{Z}| \geq \sigma_1 - \mu.$$

□

Following [BGW19, DDV10], we define a slightly different game $\mathbf{Leak}'_{\mathsf{A},\Sigma}(\lambda, b)$ which is the same as $\mathbf{Leak}_{\mathsf{A},\Sigma}$ but where the leakage oracle is instantiated with $s_1, (x, y)$ instead of s_1, s_2, and at the end of the experiment the adversary A additionally receives z in full.

Claim 4. *Assume that for all (ℓ_1, ℓ_2)-NLA adversaries A', we have*

$$\left|\mathbb{P}\left[\mathbf{Leak}'_{\mathsf{A}',\Sigma}(\lambda, 0) = 1\right] - \mathbb{P}\left[\mathbf{Leak}'_{\mathsf{A}',\Sigma}(\lambda, 1) = 1\right]\right| \leq \epsilon,$$

for some $\epsilon' \in [0,1]$. Then, for all $(\ell_1, \ell_2 - \mu)$-NLA adversaries A we have:

$$\left|\mathbb{P}\left[\mathbf{Leak}_{\mathsf{A},\Sigma}(\lambda, 0) = 1\right] - \mathbb{P}\left[\mathbf{Leak}_{\mathsf{A},\Sigma}(\lambda, 1) = 1\right]\right| \leq \epsilon \cdot 2^{\mu}.$$

Proof. By contradiction, let A be an $(\ell_1, \ell_2 - \mu)$-NLA adversary for the original game with advantage $\epsilon \cdot 2^{\mu}$. We build an attacker A' in the modified game $\mathbf{Leak}'_{\mathsf{A},\Sigma}(\lambda, b)$. The reduction A' samples z' uniformly at random, and simply runs A by forwarding its leakage queries to its own leakage oracle with hard-wired value z'. Eventually, A' receives the real value z. If $z = z'$, then A' continues running A, and else it aborts outputting a random bit. It is easy to see that the winning probability of A' equals the probability that $z = z'$ times the winning probability of A, which yields the bound in the claim.

Note that conditioned on $z = z'$, the adversary A' is $(\ell_1, \ell_2 - \mu)$-NLA (and thus (ℓ_1, ℓ_2)-NLA) since z is a deterministic function of the target shares. On the other hand, when $z \neq z'$, we can write:

$$\widetilde{\mathbb{H}}_\infty((\mathbf{X},\mathbf{Y})|\mathbf{S}_1, \mathbf{\Lambda}) \geq \widetilde{\mathbb{H}}_\infty((\mathbf{X},\mathbf{Y},\mathbf{Z})|\mathbf{S}_1, \mathbf{\Lambda}) - |\mathbf{Z}| \geq \widetilde{\mathbb{H}}_\infty((\mathbf{X},\mathbf{Y})|\mathbf{S}_1) - \ell_2.$$

□

Claim 5. *If $\mathsf{2Ext}$ is a $(\sigma_1 - \ell_1, \sigma_1 - \ell_2, \epsilon_2)$-secure two-source extractor, and Ext is a $(d - \ell_2, \epsilon_1)$-secure seeded extractor, then for all (ℓ_1, ℓ_2)-NLA adversaries A:*

$$\left|\mathbb{P}\left[\mathbf{Leak}'_{\mathsf{A},\Sigma}(\lambda, 0) = 1\right] - \mathbb{P}\left[\mathbf{Leak}'_{\mathsf{A},\Sigma}(\lambda, 1) = 1\right]\right| \leq 2(\epsilon_1 + \epsilon_2).$$

Proof. We proceed with a hybrid argument.

- Let $\mathbf{Hyb}^1_{\mathsf{A},\Sigma}(\lambda, b)$ be the same as $\mathbf{Leak}'_{\mathsf{A},\Sigma}(\lambda, b)$, except that we pick $r \leftarrow\!\!\$\ \{0,1\}^\rho$ instead of computing $r = \mathsf{2Ext}(s_1, y)$. The statistical distance between $\mathbf{Leak}'_{\mathsf{A},\Sigma}(\lambda, b)$ and $\mathbf{Hyb}^1_{\mathsf{A},\Sigma}(\lambda, b)$ is at most ϵ_2. To see this, let $\mathbf{\Lambda}$ be the leakage performed by A. By Lemma 16, the random variables $\mathbf{Y}|\mathbf{\Lambda}$ and $\mathbf{S}_1|\mathbf{\Lambda}$ are independently distributed. Moreover, since A is (ℓ_1, ℓ_2)-NLA, we have:

$$\begin{aligned}\widetilde{\mathbb{H}}_\infty(\mathbf{S}_1|\mathbf{\Lambda}) &\geq \sigma_1 - \ell_1 \\ \widetilde{\mathbb{H}}_\infty(\mathbf{Y}|\mathbf{\Lambda}) &\geq \widetilde{\mathbb{H}}_\infty(\mathbf{X}, \mathbf{Y}|\mathbf{S}_1, \mathbf{\Lambda}) - |\mathbf{X}| \\ &\geq \widetilde{\mathbb{H}}_\infty(\mathbf{X}, \mathbf{Y}|\mathbf{S}_1) - \ell_2 - d = \sigma_1 - \ell_2,\end{aligned}$$

  so that we can apply the security of the two-source extractor $\mathsf{2Ext}$.
- Let $\mathbf{Hyb}^2_{\mathsf{A},\Sigma}(\lambda, b)$ be the same as $\mathbf{Hyb}^1_{\mathsf{A},\Sigma}(\lambda, b)$, except that we now pick $z \leftarrow\!\!\$\ \{0,1\}^\mu$. The statistical distance between $\mathbf{Hyb}^1_{\mathsf{A},\Sigma}(\lambda, b)$ and $\mathbf{Hyb}^2_{\mathsf{A},\Sigma}(\lambda, b)$ is at most ϵ_1. In fact, neither the seed r nor the output z of Ext are in the view of the adversary during the leakage phase. Furthermore, a derivation similar to the one above shows that the random variable $\mathbf{X}|\mathbf{\Lambda}$ has conditional average min-entropy at least $d - \ell_2$, so that we can apply the security of the seeded extractor Ext.

The claim now follows by noting that for all (ℓ_1, ℓ_2)-NLA attackers A we must have $\mathbf{Hyb}^2_{\mathsf{A},\Sigma}(\lambda, 0) \equiv \mathbf{Hyb}^2_{\mathsf{A},\Sigma}(\lambda, 1)$, as the view of the adversary in the last hybrid is independent of b. □

Invoking the result of [CG88, DORS08], for any ℓ_1, ℓ_2, any $\alpha = \mu$, and any ϵ_1, ϵ_2, we can always find ρ, d, σ_1 polynomial in ℓ_1, ℓ_2, α and $\log(1/\epsilon_1), \log(1/\epsilon_2)$ such that Ext is a $(d - \ell_2, \epsilon_1)$-secure seeded extractor and $\mathsf{2Ext}$ is a $(\sigma_1 - \ell_1, \sigma_1 - \ell_2, \epsilon_2)$-secure two-source extractor By combining the above claims and setting $\epsilon_1 = \epsilon_2 = 2^{-\alpha-\lambda-2}$, we obtain that Σ is a noisy-leakage-resilient secret sharing scheme.