Non-Malleable Secret Sharing in the Computational Setting: Adaptive Tampering, Noisy-Leakage Resilience, and Improved Rate

Antonio Faonio[1] and Daniele Venturi[2]

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

February 3, 2019

Abstract

We revisit the concept of *non-malleable* secret sharing (Goyal and Kumar, STOC 2018) in the computational setting. In particular, under the assumption of one-to-one one-way functions, we exhibit a *computationally* private, *threshold* secret sharing scheme satisfying all of the following properties.

- **Continuous non-malleability:** No computationally-bounded adversary tampering independently with all the shares can produce mauled shares that reconstruct to a value related to the original secret. This holds even in case the adversary can tamper *continuously*, for an *unbounded* polynomial number of times, with the same target secret sharing, where the next sequence of tampering functions, as well as the subset of shares used for reconstruction, can be chosen *adaptively* based on the outcome of previous reconstructions.
- **Resilience to noisy leakage:** Non-malleability holds even if the adversary can additionally leak information independently from all the shares. There is no bound on the length of leaked information, as long as the overall leakage does not decrease the min-entropy of each share by too much.
- **Improved rate:** The information rate of our final scheme, defined as the ratio between the size of the message and the maximal size of a share, asymptotically approaches 1 when the message length goes to infinity.

Previous constructions achieved information-theoretic security, sometimes even for arbitrary access structures, at the price of *at least one* of the following limitations: (i) Non-malleability only holds against one-time tampering attacks; (ii) Non-malleability holds against a bounded number of tampering attacks, but both the choice of the tampering functions and of the sets used for reconstruction is non-adaptive; (iii) Information rate asymptotically approaching zero; (iv) No security guarantee in the presence of leakage.

Keywords: Secret sharing, non-malleability, leakage resilience, computational security.

Contents

1 Introduction

In a secret sharing (SS) scheme, a trusted dealer divides a secret message m into shares that are distributed to n parties, in such a way that any authorized subset of parties can efficiently determine the secret, whereas unauthorized subsets of parties have (statistically) no information about the message. In this paper, we focus on *threshold* secret sharing (TSS), where the unauthorized subsets are those with at most $\tau - 1$ players, for a parameter $\tau \leq n$.

The above type of SS is also known as τ-out-of-n TSS, and was originally introduced by Shamir [Sha79] and Blakey [Bla79]. SS has found many applications to cryptography, ranging from data storage [Has] and threshold cryptography [DF91], to secure message transmission [DDWY93], multi-party computation [GMW87, CCD88, BGW88], and private circuits [ISW03, FRR+10, AIS18].

An important parameter of an SS scheme is its *information rate*, defined as the ratio between the size of the message and the maximal size of a share. It is well-known that the best possible information rate for TSS satisfying statistical privacy is 1, meaning that the size of each share must at least be equal to that of the message being shared.

1.1 Non-Malleable Secret Sharing

Classical SS offers no guarantee in the presence of a *tampering adversary* modifying (possibly all!) the shares. Motivated by this shortcoming, Goyal and Kumar [GK18a] introduced *one-time non-malleable* secret sharing (NMSS), which intuitively guarantees that even if all of the shares are tampered once, the reconstructed message is either equal to the original shared value or independent of it. The only limitation is that the adversary is restricted to change the shares *independently*, a model sometimes known under the name of *individual tampering*. As usual, in order to reconstruct the secret, only $\varrho \leq n$ shares are required, and typically the reconstruction threshold ϱ equals the privacy threshold τ.

Recently, the topic of NMSS has received a lot of attention. We summarize the state of the art below, and in Tab. 1.

- In their original paper, Goyal and Kumar [GK18a] gave a construction of NMSS with 1-time non-malleability against individual tampering. The rate of this construction is $O(\frac{1}{n \log \mu})$, where μ is the size of the message. They also propose a more complicated construction, with even worse rate, that satisfies 1-time non-malleability in a stronger model where the adversary is allowed to jointly tamper subsets of up to $\tau - 1$ shares.
  In a follow-up work [GK18b], the same authors construct NMSS satisfying 1-time non-malleability against both individual and joint tampering, and further supporting arbitrary monotone access structures. The rate of these constructions asymptotically approaches zero when the length of the message goes to infinity.
- Badrinarayanan and Srinivasan [BS18] construct NMSS with improved rate. In particular, they put forward a stronger security model called p-time non-malleability, in which the adversary can tamper with the same target secret sharing $s = (s_1, \ldots, s_n)$ for $p \geq 1$ times, by *non-adaptively* specifying sequences of tampering functions

$$(f_1^{(1)}, \ldots, f_n^{(1)}), \ldots, (f_1^{(p)}, \ldots, f_n^{(p)}) \tag{1}$$

  yielding mauled shares $\tilde{s}^{(q)} = (\tilde{s}_1^{(q)}, \ldots, \tilde{s}_n^{(q)})$, for each $q \in [p]$. Non-malleability here means that for every reconstruction set $\mathcal{T}$ with size at least τ, fixed *before* tampering takes place, the secrets reconstructed out of $\tilde{s}_{\mathcal{T}}^{(1)}, \ldots, \tilde{s}_{\mathcal{T}}^{(p)}$ are independent of the original message.
  The main result of [BS18] are NMSS schemes with p-time non-malleability, both for threshold access structures (with $\varrho = \tau \geq 4$), and for arbitrary 4-monotone access structures,

Reference	Access Structure	Non-Malleability	Leakage Resilience	Rate	Assumption	Notes
[GK18a]	Threshold ($\tau \geq 2$)	1-time	✗	$O\left(\frac{1}{n \log \mu}\right)$	—	IT
	Threshold ($\tau \geq 2$)	1-time	✗	$O\left(\mu^{-9}\right)$	—	JT
[GK18b]	Arbitrary (monotone)	1-time	✗	$O\left(\frac{1}{n \log \mu}\right)$	—	IT
	Threshold ($\tau = n$)	1-time	✗	$O(\mu^{-6})$	—	JT
[BS18]	Threshold ($\tau \geq 4$)	p-time	✗	$\Theta\left(\frac{1}{p^3 \cdot \tau \cdot \log^2 n}\right)$	—	IT, NAT
	Arbitrary (4-monotone)	p-time	✗	$\Theta\left(\frac{1}{p^3 \cdot \tau_{\mathsf{max}} \cdot \log^2 n}\right)$	—	IT, NAT
[ADN+18]	Arbitrary (3-monotone)	p-time	✗	$\Theta\left(\frac{1}{n \log \mu}\right)$	–	IT, NAT, NACR
[SV18]	Arbitrary (4-monotone)	1-time	✗	$O(1)$	–	IT
[KMS18]	Arbitrary (monotone)	1-time	ℓ-Bounded	$O\left(\frac{1}{\ell n \log n \log \mu}\right)$	–	IT
This paper	Threshold ($\tau \leq \varrho - 1$)	poly-time	Noisy	1	1-to-1 OWFs	IT, ACR

Table 1: Comparison of state-of-the-art NMSS schemes. The value n denotes the number of parties, μ denotes the size of the message, ℓ denotes the leakage parameter, and τ (resp. ϱ) is the privacy (resp. reconstruction) threshold in case of TSS, where $\varrho = \tau$ unless stated otherwise. In case of general access structures, τ_{max} is the maximum size of a minimal authorized subset. IT stands for "individual tampering", JT for "joint tampering", NAT for "non-adaptive tampering", NACR for "non-adaptive concurrent reconstruction", and ACR for "adaptive concurrent reconstruction".

with rates, respectively, $\Theta\left(\frac{1}{p^3 \cdot \tau \cdot \log^2 n}\right)$ and $\Theta\left(\frac{1}{p^3 \cdot \tau_{\mathsf{max}} \cdot \log^2 n}\right)$ (where τ_{max} is the maximum size of a minimal authorized subset). Importantly, the maximal value of p is a priori fixed and, in fact, the shares' size can depend on it. Moreover, they proved that, in the information-theoretic setting, it is impossible to have a NMSS that achieves non-malleability against an unbounded polynomial number of tampering attempts.

- Aggarwal *et al.* [ADN+18] consider a strengthening of p-time non-malleability, in which the adversary tampers non-adaptively p times, as in Eq. (1), but additionally specifies p different sets $\mathcal{T}_1, \ldots, \mathcal{T}_p$ for the reconstruction of each mauled shares $\tilde{s}^{(1)}, \ldots, \tilde{s}^{(p)}$. In other words, the requirement is now that $\tilde{s}^{(1)}_{\mathcal{T}_1}, \ldots, \tilde{s}^{(p)}_{\mathcal{T}_p}$ are independent of the original message. They dub their model p-time non-malleability under *non-adaptive concurrent reconstruction*, since the sets $\mathcal{T}_1, \ldots, \mathcal{T}_p$ are specified in a non-adaptive fashion.
  The main result of [ADN+18] is a construction of NMSS satisfying p-time non-malleability under non-adaptive concurrent reconstruction, with rate $\Theta(\frac{1}{n \log \mu})$, which again asymptotically approaches zero.
- Srinivasan and Vasudevan [SV18] improve [BS18] by constructing the first NMSS for 4-monotone access structures, and satisfying 1-time non-malleability with rate $O(1)$.
- Finally, Kumar, Meka, and Sahai [KMS18] construct NMSS with 1-time non-malleability, but where the adversary is additionally allowed to adaptively leak information on the shares independently, i.e. they considered for the first time *leakage-resilient* NMSS (LR-NMSS). Note that here, the choice of the tampering functions can adaptively depend on the leakage. The rate of this scheme asymptotically approaches zero.

1.2 Our Contributions

All the above mentioned works construct NMSS, with different characteristics, in the *information-theoretic* setting, where both the privacy and the non-malleability of the scheme holds even against unbounded adversaries. A natural question is whether one can improve the state of the art in the *computational* setting, where the adversary for privacy and non-malleability is computationally bounded. Note that this is particularly appealing, in view of the fact that

fully-fledged *continuous* non-malleability is impossible to achieve in the information-theoretic setting [BS18]. Hence, the following question is open:

> *Can we construct NMSS where a* computationally-bounded *adversary can tamper* adaptively*, with the* same *target shares, for an* unbounded polynomial *number of times, and under* adaptive concurrent reconstruction*?*

In this work, we answer the above question affirmatively for the case of threshold access structures and individual tampering, assuming 1-to-1 one-way functions (OWFs). Our final scheme has rate asymptotically approaching 1, and furthermore satisfies leakage resilience.

Theorem 1 (Main Theorem, Informal)**.** *Let $\tau, \varrho, n \in \mathbb{N}$ be such that $\tau, \varrho \leq n$ and $\tau \leq \varrho - 1$. Assuming* 1*-to-*1 *OWFs, there exists* noisy-leakage-resilient, continuously non-malleable *τ-out-of-n secret sharing (LR-CNMSS) under* adaptive concurrent reconstruction *(where at least ϱ parties are needed to reconstruct the secret), with information rate (asymptotically) one.*

We observe that leakage resilience holds in the so-called *noisy*-leakage model, where the actual amount of information that can be leaked independently from each share is unbounded, as long as the uncertainty of each share does not decrease by too much. Also, notice that there is a minimal gap between the reconstruction threshold ϱ and the privacy threshold τ (i.e., $\tau \leq \varrho - 1$). Interestingly, as we explain in §4.2, CNMSS cannot exist unconditionally only for the optimal parameters $\tau = \varrho$, and thus our work leaves open the question of constructing TSS where both privacy and non-malleability hold statistically, as long as $\tau < \varrho$.

A final remark is that the definition of continuous non-malleability uses a special self-destruct feature, in which after the first *invalid* mauled secret sharing is found (i.e., a collection of shares $\tilde{s}^{(q)}_{\mathcal{T}_q}$ whose reconstruction equals an error symbol $\perp$), the answer to all *future* tampering queries is by default set to be $\perp$. As we show in §4.3, such a feature is necessary, in the sense that without it no CNMSS exists (even without considering leakage and concurrent reconstruction).

1.3 Tamper-Resilient Threshold Signatures

As an application, we consider a generalization of the classical transformation from standard security to tamper-proof security via non-malleable codes [DPW10], to the setting of threshold cryptography. For concreteness, we focus on threshold signature schemes which allow to secret share a signing key among n servers, in such a way that any subset of at least ϱ servers can interact in order to produce the signature of a message. The standard security guarantee here is that an adversary corrupting up to $\tau - 1$ servers cannot forge a valid signature, even after observing several transcripts of the signing protocol with the honest servers.

Given any CNMSS, we show how to compile a non-interactive threshold signature into an interactive (2-round) threshold signature that additionally is secure in the presence of *continuous* tampering attacks. More precisely, we imagine an external forger corrupting the memory of (possibly all!) the servers independently (say via a malware installed on each of the servers), and observing several signatures produced using arbitrarily modified secret-key shares.

A similar application was recently considered in [ADN+18]. The main advantage of our model is that the attacker is allowed to tamper continuously with the memory of the servers, and further can adaptively choose the subset of servers participating in each invocation of the signature protocol; on the negative side, our adversary is not allowed to fully corrupt any of the servers, whereas in the model of [ADN+18] the forger, after tampering once, obtains the secret-key shares of $\tau - 1$ servers. In our perspective, this difference stems from the fact that [ADN+18] makes a non-black-box usage of the underlying NMSS, which allows to exploit a slightly stronger form of non-malleability which, although not formalized by the authors, seems

to be met by their specific construction.[1] In contrast, our compiler only makes black-box calls to the underlying primitives.

1.4 Further Related Works

Robust secret sharing. In *robust* SS (see, e.g. [RB89, CSV93, RB07, BPRW16]), a monolithic adversary can (non-adaptively) corrupt up to τ players, and thus jointly tamper their shares. Robustness guarantees that given all the $\varrho = n$ shares, the reconstructed message is identical to the original shared value.

While robustness is a very strong form of non-malleability, it is clearly impossible whenever more than $n/2$ shares are corrupted (even in the computational setting).

Non-malleable codes. The concept of NMSS is intimately related to the notion of non-malleable codes (NMCs) [DPW10]. Intuitively, a NMC allows to encode a message in such a way that tampering with the resulting codeword via a function $f \in \mathcal{F}$, where $\mathcal{F}$ is a set of allowed tampering functions that is a parameter in the definition, yields a modified codeword that either decodes to the original message or to an unrelated value. Several constructions of NMCs exist in the literature, for different families $\mathcal{F}$; one of the most popular choices is to think of the tampering function as a sequence of n functions $f = (f_1, \ldots, f_n)$, where each function f_i modifies a different chunk of the codeword arbitrarily, yet independently. This is often known as the n-split-state model [DPW10, LL12, DKO13, CG14b, CG14a, ADL14, CZ14, ADKO15b, ADKO15a, CGL16, CKR16, Li17, KOS17, ADN+17], the most general case being the case $n = 2$.

As shown by Aggarwal *et al.* [ADKO15b], every NMC in the 2-split-state model is a 2-out-of-2 NMSS in disguise. Similarly, it is easy to see that any (leakage-resilient) *continuously* NMC (LR-CNMC) in the 2-split-state model [FMNV14, FNSV18, OPVV18, CFV19] is a 2-out-of-2 LR-CNMSS as per our definition.

Leakage-resilient codes. When no tampering is considered, our definition of LR-CNMSS collapses to that of leakage-resilient secret sharing, as originally introduced by Davì, Dziembowski, and Venturi, for the case $n = \tau = \varrho = 2$ [DDV10]. This topic recently received renewed attention, see, in particular, [ADN+18, SV18, KMS18].

1.5 Organization

In §2, we present the main technical ideas behind the proof of Theorem 1. Then, in §4, after setting up some basic notation and recalling a few necessary notions in §3, we describe our model for continuously non-malleable secret sharing against individual tampering and under adaptive concurrent reconstruction.

In §5 we describe and analyze a construction of LR-CNMSS with rate 0 from 1-to-1 OWFs, whereas in §6 we show how to generically transform any LR-CNMSS with rate 0 into a LR-CNMSS with rate 1, under the same assumption. The application of CNMSS to threshold signatures can be found in §7.

Finally, in §8, we conclude the paper with some remarks and directions for future research.

[1] I.e., non-malleability still holds even if the attacker learns a subset of the original shares, after tampering is over; such a property is sometimes known as *augmented* non-malleability in the non-malleable codes literature [AAG+16, CFV19].

2 Technical Highlights

Intuitively, the proof of Theorem 1 proceeds in two steps. In the first step, we show how to obtain LR-CNMSS with information rate asymptotically approaching 0, under the assumption of 1-to-1 OWFs. In the second step, we show how to boost the rate generically, from 0 to 1, under the same assumption. Below, we explain these two steps with some details, after presenting our security model informally.

2.1 Security Model

Let Σ be an n-party threshold secret sharing, with reconstruction threshold ϱ (i.e., given at least ϱ shares we can efficiently reconstruct the message) and privacy threshold τ (i.e., $\tau - 1$ shares reveal no information on the message to the eyes of a computationally-bounded adversary). In order to define continuous non-malleability for threshold secret sharing, we consider an efficient adversary interacting with a target secret sharing $s = (s_1, \ldots, s_n)$ of some message $m \in \mathcal{M}$, via the following queries.

- **Tampering:** The attacker can specify a sequence of efficiently-computable functions $(f_1^{(q)}, \ldots, f_n^{(q)})$, yielding mauled shares

$$\tilde{s}^{(q)} = (\tilde{s}_1^{(q)}, \ldots, \tilde{s}_n^{(q)}) = (f_1^{(q)}(s_1), \ldots, f_n^{(q)}(s_n)),$$

  along with a set $\mathcal{T}_q \subseteq [n]$, with size $\tilde{\varrho} \geq \varrho$. The answer to such a query is the message $\tilde{m}^{(q)}$ which is reconstructed using the shares $\tilde{s}_{\mathcal{T}_q}^{(q)}$. The above queries can be chosen in a fully-adaptive fashion for all $q \in [p]$, where p is an *arbitrary polynomial* in the security parameter; however, after the first tampering query generating an invalid message $\perp$ during reconstruction, the system switches to a *"self-destruct mode"* in which the answer to future tampering queries is automatically set to $\perp$.
- **Leakage:** The attacker can specify an efficiently-computable function g, along with an index $i \in [n]$, upon which it obtains $g(s_i)$. The above queries can be chosen in a fully-adaptive fashion, as long as the uncertainty of each share conditioned on the leakage (measured via its conditional average min-entropy [DORS08]) is reduced at most by a value $\ell \in \mathbb{N}$ that is a parameter of the scheme.

The formal definition of leakage-resilient continuous non-malleability essentially says that for each pair of messages $m_0, m_1 \in \mathcal{M}$, the adversary's view in the above experiment is computationally indistinguishable in the two cases where $m = m_0$ and $m = m_1$. Note that when $n = \tau = \varrho = 2$, and further when ℓ is an upper bound on the total amount of leakage, our definition collapses to the standard notion of a LR-CNMC in the split-state model [LL12, ADKO15b].

One might observe that our definition is game based, whereas all previous definitions of non-malleable secret sharing are simulation based. While it would be possible to give a simulation-based definition for LR-CNMSS, it is not hard to show that the two formulations would be equivalent, as long as the length of the shared value is super-logarithmic in the security parameter. The same equivalence, in fact, holds true for the case of LR-CNMCs [DPW10, OPVV18].

Finally, we would like to remark that the two limitations of computational security and self-destruct are somewhat inherent. First, as shown by [BS18], no TSS scheme with optimal parameters $\varrho = \tau$, and satisfying statistical privacy, can achieve information-theoretic continuous non-malleability w.r.t. an arbitrary polynomial number of tampering queries; as we explain in §4.2, however, the latter might still be possible with a non-zero gap $\tau < \varrho$. Second, as we formally prove in §4.3, it is also impossible to achieve continuous non-malleability without a self-destruct capability. The latter is reminiscent of similar impossibility results in the settings

of tamper-resilient cryptography and non-malleable codes [GLM+04, FMNV14]. Note that both these impossibility results hold even without considering leakage and concurrent reconstruction.

2.2 First Step: Achieving Continuous Non-Malleability (Poor Rate)

A scheme with low privacy. Consider the following simple idea, inspired by [GK18a], how to construct a 2-out-of-n CNMSS by leveraging any CNMC in the split-state model (i.e., any 2-out-of-2 CNMSS). To share a message $m \in \mathcal{M}$, we enumerate over all the possible pairs of distinct indices smaller than n, and for each such pair we compute a 2-out-of-2 CNMSS of the message. In other words, for each subset $\mathcal{H} = \{h_1, h_2\} \in \binom{[n]}{2}$, we consider a non-malleable split-state encoding $s_{\mathcal{H}} := (s_{\mathcal{H},h_1}, s_{\mathcal{H},h_2})$ of the message m, which we assign to the indices h_1 and h_2. The final share s_i^* for party $i \in [n]$ is then defined to be the collection of all the shares $s_{\mathcal{H},i}$, where $\mathcal{H}$ is such that $i \in \mathcal{H}$. Reconstruction is defined in the natural way, i.e. given an authorized set $\mathcal{H}' = \{h'_1, h'_2\}$, we simply ignore all the shares but $s_{\mathcal{H}'}$, and use $(s_{\mathcal{H}',h'_1}, s_{\mathcal{H}',h'_2})$ to reconstruct the message.

Intuitively, the above scheme is secure because the $\binom{n}{2}$ shares of the message m are independently sampled, and furthermore the reconstruction for an authorized set $\mathcal{H}$ is independent of all the shares but one. In particular, the 2-threshold privacy property follows easily by privacy of the underlying CNMC. As for continuous non-malleability, consider a sequence of hybrid experiments, one hybrid for each subset $\mathcal{H}$ in $\binom{[n]}{2}$ in lexicographic order: In each hybrid step, we change the distribution of the target secret sharing $s^* = (s_1^*, \ldots, s_n^*)$ by letting $(s_{\mathcal{H},h_1}, s_{\mathcal{H},h_2})$ be a 2-out-of-2 CNMSS of m_0 for all sets in $\binom{[n]}{2}$ up to $\mathcal{H}$, whereas we use m_1 to define the remaining shares.

For the proof, we can build a reduction to the continuous non-malleability of the underlying split-state encoding. In particular, the simulation of a generic tampering query of the form $(\mathcal{T}, (f_1, \ldots, f_n))$, proceeds as follows:

- If $\mathcal{T}$ and $\mathcal{H}$ do not share any index, then they cannot possibly interfere with each other. In particular, the reduction knows all the shares for the positions in $\mathcal{T}$, and therefore it can simulate the answer without even querying the underlying tampering oracle for the non-malleable code.
- If $\mathcal{T}$ and $\mathcal{H}$ share (at least) an index, then we can use the target tampering oracle to compute the mauled shares corresponding to $\mathcal{T}$ using the tampering oracle corresponding to $\mathcal{H}$. However, there is a catch. Let, e.g., be $\mathcal{T} = \{t_1, t_2\}$ and $\mathcal{H} = \{h_1, h_2\}$, and suppose $t_2 = h_1$. To compute the tampered share $\tilde{s}_{\mathcal{T},t_2}$, we need to know the value $s_{\mathcal{H},h_1}$, which is only accessible through the tampering oracle; as a consequence, the reduction would only be able to obtain the reconstructed message corresponding to $(\tilde{s}_{\mathcal{T},t_2}, \tilde{s}_{\mathcal{T},t_1})$, which is possibly different from the reconstructed message corresponding to $(\tilde{s}_{\mathcal{T},t_1}, \tilde{s}_{\mathcal{T},t_2})$.
  We bypass this problem by assuming that the underlying split-state CNMC has *symmetric decoding*, namely the decoding output is invariant w.r.t. the order of the two shares. As we explain later, this property is satisfied by known schemes.

Amplifying the privacy. Intuitively, the transformation above is based on the fact that by composing a secret sharing for an access structure $\mathcal{A}$ with a secret sharing for an access structure $\mathcal{A}'$, we obtain a new secret sharing for access structure $\mathcal{A} \cup \mathcal{A}'$. Unfortunately, we cannot generalize this idea to go from ϱ-out-of-ϱ to ϱ-out-of-n secret sharing for any $\varrho \leq n$, as for efficiency we need $\binom{n}{\varrho} \approx n^{\varrho}$ to be polynomial in n.

The key idea behind our main construction of CNMSS is to compose together $\binom{[n]}{2}$ secret sharing schemes with different access structures, such that their union gives the desired ϱ-threshold access structure. Specifically, consider the following construction of a ϱ-out-of-n TSS

based on a split-state CNMC, on an authenticated secret-key encryption (AE) scheme, and on an auxiliary $(\varrho-3)$-out-of-$(n-2)$ TSS.

For a fixed pair of indices $\mathcal{H}=\{h_1,h_2\}\in\binom{[n]}{2}$, pick a uniformly random key $\kappa_{\mathcal{H}}$ for the AE scheme, compute a split-state encoding of $\kappa_{\mathcal{H}}$, and call the resulting shares $(s_{\mathcal{H},h_1},s_{\mathcal{H},h_2})$; hence, encrypt the message m under the key $\kappa_{\mathcal{H}}$ obtaining a ciphertext $c_{\mathcal{H}}$, and secret share $c_{\mathcal{H}}$ using the auxiliary TSS, yielding shares $(s_{\mathcal{H},h_3},\ldots,s_{\mathcal{H},h_n})$ where $\{h_3,\ldots,h_n\}=[n]\setminus\mathcal{H}$. Notice that this scheme has access structure $\mathcal{A}_{\mathcal{H}}=\{\mathcal{S}\subset[n]:\ |\mathcal{S}|\geq\varrho,\mathcal{H}\subset\mathcal{S}\}$. By repeating the above procedure for each set $\mathcal{H}\in\binom{[n]}{2}$, we obtain that the final share s_i^* for party $i\in[n]$ is the collection of all the shares $s_{\mathcal{H},i}$, so that $\bigcup_{\mathcal{H}\in\binom{[n]}{2}}\mathcal{A}_{\mathcal{H}}$ yields the ϱ-threshold access structure, as desired. Moreover, the size of each share is still polynomial in the number of parties.

The proof of threshold privacy is rather straightforward, at least if we set the privacy threshold for the final scheme to be $\tau\leq\varrho-2$. However, in the computational setting, we can even show privacy $\tau\leq\varrho-1$. The key idea is that either the adversary has enough shares to reconstruct the underlying ciphertext (but in this case it does not have access to the secret key, and therefore it learns nothing by semantic security of the encryption scheme), or, the adversary knows at most $\varrho-3$ shares of the ciphertext (which by perfect privacy of the auxiliary TSS reveal nothing about the ciphertext).

Proving continuous non-malleability. The intuition for non-malleability of the inner secret sharing with access structure $\mathcal{A}_{\mathcal{H}}$ is that by tampering the shares corresponding to indices h_1,h_2, the adversary either obtains the original key or a completely unrelated value: In the former case, by the authenticity of the AE scheme, the adversary cannot produce a new ciphertext that decrypts correctly; in the latter case, by the semantic security of the AE scheme, the adversary cannot produce a ciphertext that decrypts to a related message (under the unrelated key generated via tampering).

However, in contrast to the simple 2-out-of-n TSS construction hinted above, in the new scheme the share of party i consists of both the shares of a split-state encoding of a key, and the shares of a ciphertext under an auxiliary standard TSS scheme. Hence, in a tampering query, the adversary could swap these two kinds of shares, with the consequence that the reconstruction procedure of the underlying $(\varrho-3)$-out-of-$(n-2)$ secret sharing would depend on one of the two shares of the non-malleable code. To resolve this problem we rely on two different ideas: First, we additionally assume that the split-state CNMC is resilient to *noisy leakage*; second, we make sure that the reconstruction procedure of the auxiliary threshold secret sharing scheme does not leak information about single shares.

The second idea is the most important one. In fact, by simply assuming leakage resilience we could at most tolerate an a priori bounded number of tampering queries. The reason for this is that, even if each reconstruction leaks just a single bit of a share $s_{\mathcal{H},i}$ under the split-state CNMC, after $|s_{\mathcal{H},i}|$ consecutive tampering queries this share could be leaked without provoking a self-destruct. The latter is better understood by looking at Shamir's threshold secret sharing, where to share $m\in\mathcal{M}$ we pick a random polynomial of degree ϱ that evaluates to m at point 0, and distribute to the i-th party the share s_i obtained by evaluating the polynomial at point $i\in[n]$. The reconstruction algorithm, given any set of ϱ shares s_i, interpolates the corresponding points, thus obtaining a polynomial that is evaluated on the origin. It is easy to see that such a reconstruction procedure, under tampering attacks, potentially leaks a lot of information about the single points (without the risk of self-destruct). In particular, the reconstruction algorithm is a linear function of the shares, and thus perturbing one point by a multiplicative factor, allows to recover the value of a share in full via a single tampering query.

We now show how to avoid the above leakage. Fix some index $i\in[n]$ for the i-th share. Given an authorized set of size ϱ, we let our reconstruction procedure select two different

subsets[2] of size $\varrho - 3$, such that one subset includes the index i, whereas the second subset excludes it. Thus, we run the standard reconstruction procedure twice, one for each subset, and we accept the reconstructed message if and only if the two runs yield the same value, otherwise we return an error message (which triggers a self-destruct). The main observation is that the second run of the reconstruction algorithm is independent of $s_{\mathcal{H},i}$, and thus, conditioned on the returned message not being $\perp$, the output of the reconstruction is independent of $s_{\mathcal{H},i}$. On the other hand, when the returned message is equal to $\perp$, the output of the reconstruction could indeed leak information about the share with index i, but notice that this situation triggers a self-destruct, and thus such leakage happens only once.

Finally, note that in order to lift this argument to all the indices within an authorized set, we could simply repeat the above mechanism for all subsets of size $\varrho - 3$. However, for the case of Shamir's secret sharing, this is not necessary, and in fact we can have a more efficient reconstruction procedure that only checks two subsets. In particular, if two different subsets of size $\varrho - 3$ yield polynomials with identical evaluation in the origin, then they must encode the same polynomial, and since these two subsets cover an entire authorized set, then we are ensured that using any other subset would yield the same reconstructed message.

Instantiating the construction. All that remains is to construct a split-state CNMC with the special symmetric decoding feature, and for which the non-malleability property still holds even in the presence of noisy (independent) leakage from the left and right shares.

We do this by revisiting the recent construction of Ostrovsky *et al.* [OPVV18], which gives a split-state CNMC assuming non-interactive, perfectly binding commitments (which in turn can be based on 1-to-1 OWFs). In their scheme, a split-state encoding of a message m is a pair of values $(L, R) = ((com, L'), (com, R'))$, where com is a non-interactive commitment to the message m using randomness δ, and (L', R') is a split-state encoding of the string $m||\delta$ obtained by running an auxiliary code satisfying leakage-resilient one-time non-malleability, in the information-theoretic setting and in the bounded-leakage model. The decoding algorithm first checks that the left and right share contain the same commitment. If not, it returns $\perp$. Else, it decodes (L', R') obtaining a string $m' = m||\delta$, and returns m if and only if δ is a valid opening of com w.r.t. m.

Our first observation is that the above code satisfies symmetric decoding, as long as the inner encoding (L', R') does. Additionally, we extend the security proof of [OPVV18] to show that if the auxiliary split-state code is secure in the noisy-leakage model, so is the final encoding. As a side result, and thanks to the power of noisy leakage, we even obtain a simpler proof.

The missing piece of the puzzle is then to exhibit a split-state code satisfying leakage-resilient one-time non-malleability, in the information-theoretic setting and in the noisy-leakage model, and with symmetric decoding. Luckily, it turns out that the coding scheme by Aggarwal *et al.* [ADKO15b], based on the inner-product extractor [CG88], already satisfies all these requirements. We refer the interested reader to §5.3 for the details.

2.3 Second Step: Amplifying the Rate

Next, we describe another generic transformation yielding LR-CNMSS with information rate asymptotically approaching 1, starting from a LR-CNMSS with asymptotic rate 0, and an AE scheme. Such transformations, in the setting of non-malleable codes, are sometimes known as rate compilers [AGM+15, AAG+16, CFV19].

[2] In retrospect, this is the reason why we set the reconstruction/privacy threshold of the underlying secret sharing scheme to $\varrho - 3$ (i.e., 2 shares for decoding the non-malleable encoding and $\varrho - 3 + 1 = \varrho - 2$ shares to run the reconstruction procedure of the threshold secret sharing twice).

Our rate compiler generalizes a beautiful construction by Agrawal *et al.* [AAG+16] in the setting of split-state one-time non-malleable codes, which has been very recently analyzed also in the case of continuous tampering [CFV19]. In order to secret share the message $m \in \mathcal{M}$, we first sample a uniformly random key κ for the AE scheme, and then we encrypt the message m under this key, yielding a ciphertext c. Hence, we secret share the key κ using the underlying rate-0 secret sharing scheme, yielding n shares $(\kappa_1, \ldots, \kappa_n)$. Finally, we set the share of party $i \in [n]$ to be $s_i = (\kappa_i, c)$. The reconstruction procedure, given ϱ shares, first checks that all shares contain the same ciphertext c. If not, an error is triggered. Else, the secret key is reconstructed from the shares and used in order to decrypt the unique ciphertext c.

Note that the length of the secret key is independent of the size of the message, and thus the above construction achieves information rate asymptotically approaching 1. As for security, it is not hard to show that the compiled scheme inherits the threshold privacy property from the underlying rate-0 secret sharing. Here, we additionally need to rely on the semantic security of the AE scheme to argue that the ciphertext c reveals nothing about the message.

Proving continuous non-malleability. Turning to continuous non-malleability, the main step of the proof is a game hop in which the values $(\kappa_1, \ldots, \kappa_n)$ result from a secret sharing of an unrelated key $\kappa' \neq \kappa$. In order to establish the indistinguishability between this modified experiment and the original experiment, we consider a reduction to the continuous non-malleability of the underlying LR-CNMSS. Such a reduction can interact with a target secret sharing $(\kappa_1, \ldots, \kappa_n)$ that is either a secret sharing of κ or of κ'. The main obstacle, here, comes from the simulation of tampering queries. In fact, although the reduction can perfectly emulate the distribution of the individual shares $s_i = (\kappa_i, c)$ inside the tampering oracle, as the ciphertext c can be sampled locally, the difficulty is that to emulate the output of the reconstruction w.r.t. a given subset $\mathcal{T} = \{t_1, \ldots, t_{\tilde{\varrho}}\}$ we need to: (i) ensure that all of the mauled shares $\tilde{s}_{t_j} = (\tilde{\kappa}_{t_j}, \tilde{c}_{t_j})$ actually contain the same ciphertext, i.e. $\tilde{c}_{t_1} = \ldots = \tilde{c}_{t_{\tilde{\varrho}}} = \tilde{c}$, and (ii) use the mauled secret key $\tilde{\kappa}$ received by the reduction in response to a tampering query in order to obtain the decryption of the unique ciphertext $\tilde{c}$ (if such a ciphertext exists).

We overcome both of the above obstacles by exploiting the fact that the starting CNMSS is resilient to noisy leakage. This is crucial in our setting, since the size of the ciphertext might very well exceed the maximal length of a share of the secret key. Hence, generalizing a trick from [FNSV18, CFV19], we proceed to check equality of all the ciphertexts in a block-wise fashion, by leaking blocks of λ bits from each share, where λ is the security parameter. This leakage routine continues until eventually we obtain the entire ciphertext $\tilde{c}$, unless some of the blocks leaked from each share differ, in which case we answer the tampering query by $\perp$ and trigger a self-destruct.

It remains to show that the above methodology does not result in too much leakage. Intuitively, this holds because up to the point where the leaked blocks of the ciphertexts are all the same, the leakage on each share can be thought of as a function of the other shares, so that this leakage does not decrease the min-entropy of each share more than conditioning on the other shares, which is fine since in known constructions the mutual information between the shares is very low. On the other hand, when a self-destruct is triggered, we reveal only λ bits of information; by a standard argument, this causes a min-entropy drop of roughly λ bits, which again is tolerated by the underlying scheme.

3 Preliminaries

3.1 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

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

The above naturally generalizes to statistical distance (in case of unbounded adversaries).

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 $\widetilde{\mathbb{H}}_\infty(\mathbf{X}|\mathbf{Y}) := -\log \mathbb{E}_y\left[2^{-\mathbb{H}_\infty(\mathbf{X}|\mathbf{Y}=y)}\right]$ [DORS08]. The lemma below is sometimes known as the "chain rule" for conditional average min-entropy.

Lemma 1 ([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, $\widetilde{\mathbb{H}}_\infty(\mathbf{X}|\mathbf{Y}) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{X},\mathbf{Y}) - \ell \geq \mathbb{H}_\infty(\mathbf{X}) - \ell$.*

3.2 Standard Cryptographic Primitives

3.2.1 Threshold Secret Sharing

An n-party secret sharing scheme Σ consists of a pair of polynomial-time algorithms $(\mathsf{Share}, \mathsf{Rec})$ specified as follows: (i) The randomized sharing algorithm Share takes as input a message $m \in \mathcal{M}$, and outputs n shares $s_1, \ldots, s_n$ where each $s_i \in \mathcal{S}_i$; (ii) The deterministic algorithm Rec takes as input a certain number of candidate shares and outputs a value in $\mathcal{M} \cup \{\perp\}$. Given $s = (s_1, \ldots, s_n)$ and a subset $\mathcal{I} \subseteq [n]$, we often write $s_{\mathcal{I}}$ to denote the shares $(s_i)_{i \in \mathcal{I}}$.

Definition 1 (Threshold secret sharing)**.** Let $n, \tau, \varrho \in \mathbb{N}$, with $\tau \leq \varrho \leq n$. We say that $\Sigma = (\mathsf{Share}, \mathsf{Rec})$ is an (n, τ, ϱ)-threshold secret sharing scheme ((n, τ, ϱ)-TSS for short) over 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 with the following properties.

(i) **ϱ-Threshold Reconstruction:** For all messages $m \in \mathcal{M}$, and for all subsets $\mathcal{I} \subseteq [n]$ such that $|\mathcal{I}| \geq \varrho$, we have that $\mathsf{Rec}((\mathsf{Share}(m))_{\mathcal{I}}) = m$, with overwhelming probability over the randomness of the sharing algorithm.

(ii) **τ-Threshold Privacy:** For all pairs of messages $m_0, m_1 \in \mathcal{M}$, and for all unqualified subsets $\mathcal{U} \subseteq [n]$ such that $|\mathcal{U}| < \tau$, we have that

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

If the ensembles $\{(\mathsf{Share}(1^\lambda, m_0))_{\mathcal{U}}\}_{\lambda \in \mathbb{N}}$ and $\{(\mathsf{Share}(1^\lambda, m_1))_{\mathcal{U}}\}_{\lambda \in \mathbb{N}}$ are statistically close (resp. identically distributed), we speak of *statistical* (resp. *perfect*) τ-threshold privacy.

Typical TSS schemes achieve the optimal parameters $\varrho = \tau$. However, having a small gap between the privacy and reconstruction threshold makes sense too, and looking ahead our constructions will have minimal gap $\varrho - \tau \geq 1$.

Special reconstruction. We will need TSS schemes meeting an additional reconstruction property, which we call *special reconstruction*. This means that for any subset $\mathcal{I} \subset [n]$ of size at least $\varrho + 1$, and for any $m \in \mathcal{M}$ which is secret shared as in $(s_1, \ldots, s_n) \leftarrow\!\!\$\ \mathsf{Share}(m)$, if there exists two subsets $\mathcal{I}_1, \mathcal{I}_2 \subset \mathcal{I}$ of size ϱ such that

$$\mathsf{Rec}((s_i)_{i \in \mathcal{I}_1}) = \mathsf{Rec}((s_i)_{i \in \mathcal{I}_2}),$$

then the above equation holds for all subsets $\mathcal{I}_1, \mathcal{I}_2 \subset \mathcal{I}$ of size ϱ.

3.2.2 Authenticated Encryption

A (secret-key) authenticated encryption (AE) scheme is a tuple of polynomial-time algorithms $\Pi = (\mathsf{KGen}, \mathsf{AEnc}, \mathsf{ADec})$ specified as follows: (i) The randomized algorithm KGen takes as input the security parameter $\lambda \in \mathbb{N}$, and outputs a uniform key $\kappa \leftarrow\!\!\$\ \mathcal{K}$; (ii) The randomized algorithm AEnc takes as input a key $\kappa \in \mathcal{K}$ and a message $m \in \mathcal{M}$, and outputs a ciphertext $c \in \mathcal{C}$; (iii) The deterministic algorithm ADec takes as input a key $\kappa \in \mathcal{K}$ and a ciphertext $c \in \{0,1\}^*$, and outputs a value $m \in \mathcal{M} \cup \{\perp\}$, where $\perp$ denotes an invalid ciphertext. We call $\mathcal{K}$, $\mathcal{M}$, $\mathcal{C}$, respectively, the key, message, and ciphertext space of Π.[3]

We say that Π meets correctness if for all $\kappa \in \mathcal{K}$, and all messages $m \in \mathcal{M}$, we have that $\mathbb{P}[\mathsf{ADec}(\kappa, \mathsf{AEnc}(\kappa, m)) = m] = 1$ (where the probability is taken over the randomness of AEnc). As for security, we will need AE schemes that satisfy two properties (see below for formal

[3]These sets typically depend on the security parameter, but we drop this dependency to simplify notation.

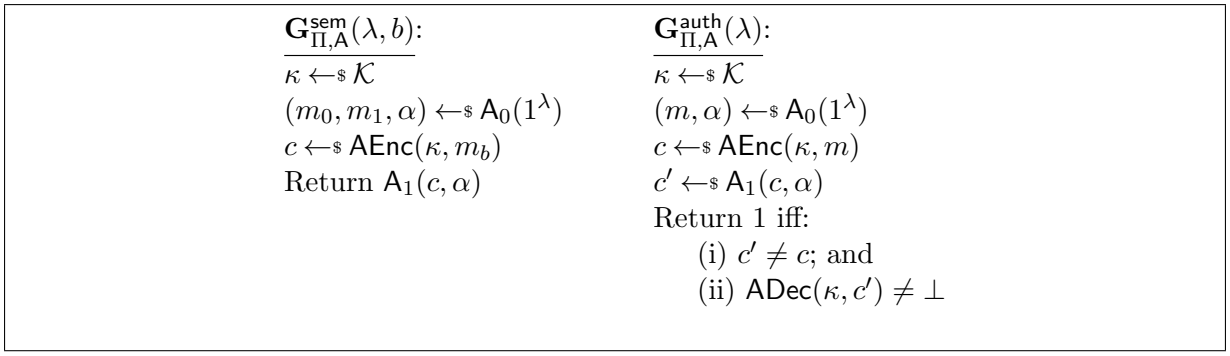

Figure 1: Experiments defining security of authenticated encryption.

definitions). The first property, usually known as *semantic security*, says that it is hard to distinguish the encryption of any two (adversarially chosen) messages. The second property, usually called *authenticity*, says that, without knowing the secret key, it is hard to produce a valid ciphertext (i.e., a ciphertext that does not decrypt to $\perp$).

Definition 2 (Security of AE)**.** Let $\Pi = (\mathsf{KGen}, \mathsf{AEnc}, \mathsf{ADec})$ be an AE scheme. We say that Π is secure if the following holds for the games defined in Fig. 1.

$$\forall \text{ PPT } \mathsf{A}: \ \mathbb{P}\left[\mathbf{G}^{\mathsf{auth}}_{\Pi,\mathsf{A}}(\lambda) = 1\right] \in \mathtt{negl}(\lambda)$$
$$\left\{\mathbf{G}^{\mathsf{sem}}_{\Pi,\mathsf{A}}(\lambda, 0)\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{\mathbf{G}^{\mathsf{sem}}_{\Pi,\mathsf{A}}(\lambda, 1)\right\}_{\lambda\in\mathbb{N}}.$$

Note that since both authenticity and semantic security are one-time guarantees, in principle, information-theoretic constructions with such properties are possible when $|\mathcal{K}| \geq |\mathcal{M}|$. However, we are interested in constructions where $|\mathcal{M}| \gg |\mathcal{K}|$, for which the existence of one-way functions is a necessary assumption.

3.3 Non-Malleable Codes

A split-state code $\Gamma = (\mathsf{Enc}, \mathsf{Dec})$ consists of a pair of polynomial-time algorithms specified as follows: (i) The randomized encoding algorithm Enc takes as input a message $m \in \mathcal{M}$ and returns a split-state codeword $(L, R) \in \mathcal{L} \times \mathcal{R}$; (ii) The (deterministic) decoding algorithm Dec takes as input a codeword $(L, R) \in (\{0,1\}^*)^2$ and outputs a value in $\mathcal{M} \cup \{\perp\}$, where $\perp$ denotes an *invalid* codeword. A codeword (L, R) such that $\mathsf{Dec}(L, R) \neq \perp$ is called a *valid* codeword; we call $\mathcal{M}$ the message space, and $\mathcal{L}, \mathcal{R}$ the left and right codeword space.

We say that Γ satisfies *correctness* if, for all $m \in \mathcal{M}$, we have that $\mathsf{Dec}(\mathsf{Enc}(m)) = m$ with overwhelming probability over the randomness of the encoding algorithm.

Noisy leakage. We will leverage codes where non-malleability (as defined below) is satisfied even in the presence of adversaries that can obtain *independent leakage* on the two shares of a target encoding (L, R).

Following a long tradition in leakage-resilient cryptography [DHLW10, NS12, FNV17], we model the leakage as an arbitrary function of its input. The only restriction is that the overall leakage on L does not decrease the min-entropy of L more than a fixed amount $\ell \in \mathbb{N}$ (that is a parameter of the scheme). Of course, an analogous condition must be satisfied for the leakage on the right side R. We formalize this restriction via a notion of *admissibility*, as defined below.

Definition 3 (Admissible adversaries for split-state codes)**.** Let $\Gamma = (\mathsf{Enc}, \mathsf{Dec})$ be a split-state code. We say that a PPT adversary A is ℓ-admissible if it outputs a sequences of leakage queries (chosen adaptively) $(g^{(q)}_{\mathsf{left}}, g^{(q)}_{\mathsf{right}})_{q\in[p]}$, with $p(\lambda) \in \mathtt{poly}(\lambda)$, such that for all messages $m \in \mathcal{M}$:

$$\widetilde{\mathbb{H}}_\infty\left(\mathbf{L}|\mathbf{R}, g^{(1)}_{\mathsf{left}}(\mathbf{L}), \cdots, g^{(p)}_{\mathsf{left}}(\mathbf{L})\right) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R}) - \ell$$
$$\widetilde{\mathbb{H}}_\infty\left(\mathbf{R}|\mathbf{L}, g^{(1)}_{\mathsf{right}}(\mathbf{R}), \cdots, g^{(p)}_{\mathsf{right}}(\mathbf{R})\right) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{R}|\mathbf{L}) - \ell,$$

where $(\mathbf{L}, \mathbf{R})$ is the joint random variable corresponding to $\mathsf{Enc}(1^\lambda, m)$.

Note that we measure the min-entropy drop due to the leakage w.r.t. the conditional average min-entropy of $L|R$ and $R|L$. We find this meaningful as it allows to capture automatically the correlation between L and R. Alternatively, we could define admissibility by conditioning only on the leakage (without further considering the other share in the equations above); we observe, however, that these two notions of admissibility are equivalent up to a small loss in the leakage parameter. This is due to the fact that, in known instantiations [ADKO15b, Li17], the mutual information between L and R is small, a property sometimes known as conditional independence [OPVV18, FNSV18, CFV19].

Continuous non-malleability. Intuitively, a split-state code is non-malleable [DPW10, LL12] if no adversary tampering *independently* (yet arbitrarily) with the two sides of a given target encoding (L, R) of some value m, can generate a modified codeword $(\tilde{L}, \tilde{R})$ that decodes to a value related to m. Continuous non-malleability [FMNV14] is a strengthening of this guarantee, where the attacker is allowed to tamper continuously, and adaptively, with (L, R), until a decoding error occurs, after which the system "self-destructs" and stops answering tampering queries. Such a self-destruct capability, that in practice might be implemented via a public write-once flag, is well known to be necessary for achieving continuous non-malleability, as otherwise simple attacks are possible [GLM+04].

We formalize continuous non-malleability for split-state non-malleable codes using a game-based definition. Simulation-based definitions also exist, but the two formulations are known to be equivalent as long as the messages to be encoded have super-logarithmic length in the security parameter [DPW10, OPVV18]. In order to model (split-state) tampering attacks, we use a stateless leakage oracle $\mathcal{O}_{\mathsf{leak}}$ and a stateful oracle $\mathcal{O}_{\mathsf{nmc}}$ that are initialized with a target encoding (L, R) of either of two messages $m_0, m_1 \in \mathcal{M}$. The goal of the attacker is to distinguish which message was encoded, while performing both leakage and tampering attacks: The leakage oracle allows the adversary to obtain information from L and R, while the tampering oracle allows the adversary to tamper with L and R independently. In case the decoded message corresponding to a modified codeword $(\tilde{L}, \tilde{R})$ is equal to one of the original messages m_0, m_1, the oracle returns a special symbol ♥, as otherwise it would be trivial to distinguish which message was encoded by querying the oracle with, e.g., the identity function.

Definition 4 (Split-state continuously non-malleable codes)**.** Let $\Gamma = (\mathsf{Enc}, \mathsf{Dec})$ be a split-state code. We say that Γ is an ℓ-noisy leakage-resilient split-state *continuously non-malleable* code (ℓ-LR-CNMC for short) if for all $m_0, m_1 \in \mathcal{M}$ and for all PPT ℓ-admissible adversaries A as per Def. 3, we have that

$$\{\mathbf{CNMC}_{\Gamma,\mathsf{A}}(\lambda, m_0, m_1, 0)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{CNMC}_{\Gamma,\mathsf{A}}(\lambda, m_0, m_1, 1)\}_{\lambda\in\mathbb{N}}, \tag{2}$$

where, for $b \in \{0, 1\}$, experiment $\mathbf{CNMC}_{\Gamma,\mathsf{A}}(\lambda, m_0, m_1, b)$ is depicted in Fig. 2.

$\underline{\mathbf{CNMC}_{\Gamma,\mathsf{A}}(\lambda, m_0, m_1, b)}$:
$(L, R) \leftarrow\!\!\$\ \mathsf{Enc}(m_b)$
$\mathtt{stop} \leftarrow \mathtt{false}$
Return $\mathsf{A}^{\mathcal{O}_{\mathsf{nmc}}((L,R),\cdot,\cdot),\mathcal{O}_{\mathsf{leak}}((L,R),\cdot,\cdot)}(1^\lambda)$
$\underline{\mathcal{O}_{\mathsf{leak}}((L, R), \mathtt{side}, g)}$:
If $\mathtt{side} = \mathtt{left}$
  Return $g(L)$
If $\mathtt{side} = \mathtt{right}$
  Return $g(R)$

$\underline{\text{Oracle } \mathcal{O}_{\mathsf{nmc}}((L, R), f_{\mathsf{left}}, f_{\mathsf{right}})}$:
If $\mathtt{stop} = \mathtt{true}$
  Return $\perp$
Else
  $(\tilde{L}, \tilde{R}) = (f_{\mathsf{left}}(L), f_{\mathsf{right}}(R))$
  $\tilde{m} = \mathsf{Dec}(\tilde{L}, \tilde{R})$
  If $\tilde{m} \in \{m_0, m_1\}$
    Return ♥
  If $\tilde{m} = \perp$
    Return $\perp$, and $\mathtt{stop} \leftarrow \mathtt{true}$
  Else
    Return $\tilde{m}$

Figure 2: Experiment defining continuously non-malleable codes in the split-state model. The tampering oracle $\mathcal{O}_{\mathsf{nmc}}$ is implicitly parameterized by the flag $\mathtt{stop}$.

Message uniqueness. An important property that must be satisfied by any split-state continuously non-malleable code is that of *message uniqueness* (MU) [FMNV14, OPVV18]. Informally, this means that if we fix the left side L of an encoding, there are no[4] two right sides R_1, R_2, such that both (L, R_1) and (L, R_2) are *valid* codewords that decode to *different* messages $m_1 \neq m_2$. (An analogous guarantee must hold if we fix the right side.)

A simple observation, due to [OPVV18], is that both the left side L and the right side R of a split-state non-malleable encoding constitute a perfectly binding commitment to the message.

Lemma 2 ([OPVV18])**.** *Let Γ be a split-state code satisfying MU. Then, for any string $L \in \{0,1\}^*$ (resp. $R \in \{0,1\}^*$), there exists at most a single value $m \in \mathcal{M}$ such that $\mathsf{Dec}(L, R) = m \neq \perp$ for some $R \in \{0,1\}^*$ (resp. for some $L \in \{0,1\}^*$).*

Additional properties. For our main construction, we will need CNMCs satisfying two additional properties as defined below. The first property, called symmetric decoding, says that for all possible inputs L, R, decoding (L, R) yields the same as decoding (R, L). Note that this implies some (very weak) form of resilience against tampering via permutations, in that any split-state continuously non-malleable code with symmetric decoding is still secure w.r.t. attackers that first tamper the two states (L, R) independently, and later swap L and R.

Definition 5 (Symmetric decoding)**.** We say that a split-state code $\Gamma = (\mathsf{Enc}, \mathsf{Dec})$ has symmetric decoding if for all $L, R \in (\{0,1\}^*)^2$, we have that $\mathsf{Dec}(L, R) = \mathsf{Dec}(R, L)$.

The second property, called codewords uniformity, requires that, for any message, the encoder outputs codewords that are uniform over the set of all possible encodings of the message.

Definition 6 (Codewords uniformity)**.** We say that a split-state code $\Gamma = (\mathsf{Enc}, \mathsf{Dec})$ has codewords uniformity if for all $m \in \mathcal{M}$, we have that $\mathsf{Enc}(1^\lambda, m)$ is distributed uniformly over the set of all possible pairs (L, R) such that $\mathsf{Dec}(L, R) = m$.

[4] Observe that "perfect" MU, as opposed to "computational" MU is wlog. in the plain model.

4 Continuously Non-Malleable Secret Sharing

4.1 Non-Malleability under Adaptive Concurrent Reconstruction

We now give the definition of leakage-resilient continuously non-malleable secret sharing (LR-CNMSS) under adaptive concurrent reconstruction. We focus on the case of threshold secret sharing, where the adversary is allowed to tamper (possibly all!) the shares arbitrarily, but independently. Non-malleability intuitively guarantees that the reconstructed message, where the indices $\mathcal{T}$ (with $|\mathcal{T}| = \tilde{\varrho} \geq \varrho$) used for reconstruction are also chosen by the adversary, is independent of the original message.

Importantly, in our model, the adversary is allowed to tamper continuously, and adaptively, with the same target secret sharing; the set used for reconstruction in each tampering attempt is also adversarial, and moreover can be chosen adaptively based on the outcome of previous queries. This feature, known as *concurrent reconstruction*, was already considered in previous work [ADN+18], although in a non-adaptive setting. There are only two limitations: (i) The adversary is computationally bounded; (ii) After the first tampering query yielding a mauled secret sharing that reconstructs to $\perp$, the answer to all future tampering queries will be $\perp$ by default. The second limitation is sometimes known as "self-destruct feature" in the literature of non-malleable codes [FMNV14]. Both of these limitations are somewhat *necessary* (see below).

In order to make our model even stronger, we further allow the adversary to leak information independently from all the shares. The only restriction here is that the leakage does not decrease the amount of uncertainty contained in each of the shares by too much. This leads to the notion of admissible adversary, which is similar in spirit to the notion of admissible adversaries for codes (cf. §3.3), as defined below.

Definition 7 (Admissible adversaries for secret sharing)**.** Let $\Sigma = (\mathsf{Share}, \mathsf{Rec})$ be an n-party secret sharing scheme. We say that a PPT adversary A is ℓ-admissible if it outputs a sequence of leakage queries (chosen adaptively) $(i, g_i^{(q)})_{i\in[n],q\in[p]}$, with $p(\lambda) \in \mathtt{poly}(\lambda)$, such that for all $i \in [n]$, and for all $m \in \mathcal{M}$:

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

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

Definition 8 (Continuously non-malleable threshold secret sharing)**.** Let $n, \tau, \varrho, \ell \in \mathbb{N}$. Let $\Sigma = (\mathsf{Share}, \mathsf{Rec})$ be an n-party secret sharing over message space $\mathcal{M}$ and share space $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$. We say that Σ is an ℓ-noisy leakage-resilient *continuously non-malleable* (n, τ, ϱ)-threshold secret sharing scheme under *adaptive concurrent reconstruction* ((n, τ, ϱ, ℓ)-LR-CNMSS for short) if it is an (n, τ, ϱ)-TSS as per Def. 1, and additionally for all pairs of messages $m_0, m_1 \in \mathcal{M}$, and all PPT ℓ-admissible adversaries A as per Def. 7, we have:

$$\{\mathbf{CNMSS}_{\Sigma,\mathsf{A}}(\lambda, m_0, m_1, 0)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{CNMSS}_{\Sigma,\mathsf{A}}(\lambda, m_0, m_1, 1)\}_{\lambda\in\mathbb{N}},$$

where, for $b \in \{0, 1\}$, experiment $\mathbf{CNMSS}_{\Sigma,\mathsf{A}}(\lambda, m_0, m_1, b)$ is depicted in Fig. 3.

Remark 1 (On game-based security)**.** *Note that Def. 8 is game based in spirit. This is in contrast with all previous definitions of non-malleable secret sharing, which instead are simulation based. While, one could also formulate a simulation-based definition for LR-CNMSS, it is not hard to show that the two formulations are equivalent as long as the shared value has super-logarithmic length in the security parameter. A similar equivalence holds for the case of (continuously) non-malleable codes [DPW10, OPVV18].*

$\underline{\mathbf{CNMSS}_{\Sigma,\mathsf{A}}(\lambda, m_0, m_1, b)}$:
$s := (s_1, \ldots, s_n) \leftarrow\!\!\$\ \mathsf{Share}(m_b)$
$\mathtt{stop} \leftarrow \mathtt{false}$
Return $\mathsf{A}^{\mathcal{O}_{\mathsf{nmss}}(s,\cdot,\cdot),\mathcal{O}_{\mathsf{leak}}(s,\cdot,\cdot)}(1^\lambda)$

$\underline{\text{Oracle } \mathcal{O}_{\mathsf{leak}}(s, i \in [n], g)}$:
Return $g(s_i)$

$\underline{\text{Oracle } \mathcal{O}_{\mathsf{nmss}}(s, \mathcal{T}, (f_1, \ldots, f_n))}$:
If $\mathtt{stop} = \mathtt{true}$
  Return $\perp$
Else
  $\mathcal{T} := \{t_1, \ldots, t_{\tilde{\varrho}}\}$
  $\tilde{s} := (\tilde{s}_1, \ldots, \tilde{s}_n) = (f_1(s_1), \ldots, f_n(s_n))$
  $\tilde{m} = \mathsf{Rec}(\tilde{s}_{t_1}, \ldots, \tilde{s}_{t_{\tilde{\varrho}}})$
  If $\tilde{m} \in \{m_0, m_1\}$
    Return ♥
  If $\tilde{m} = \perp$ return $\perp$, and $\mathtt{stop} \leftarrow \mathtt{true}$
  Else return $\tilde{m}$

Figure 3: Experiment defining leakage-resilient continuously non-malleable secret sharing against individual tampering, under adaptive concurrent reconstruction. Note that the oracle $\mathcal{O}_{\mathsf{nmss}}$ is implicitly parameterized by the flag $\mathtt{stop}$.

Remark 2 (On the relation with CNMCs)**.** *When $\ell = 0$, $n = 2$, and $\tau = \varrho = 2$, one obtains the definition of split-state CNMCs as a special case. In fact, similar to [ADKO15b], one can show that any split-state CNMC satisfies 2-threshold privacy.*

In the following subsections, we show that both limitations of computational security and self-destruct are somewhat inherent in our model (even when no leakage is allowed, i.e. $\ell = 0$). This is immediate for the case $n = 2 = \tau = \varrho = 2$, as the same limitations hold for the case of split-state CNMCs [FMNV14]. The theorems below[5] generalize the impossibility results of [FMNV14] for certain values of n, τ, ϱ.

4.2 Shared-Value Uniqueness

Consider the following natural generalization of the MU property for continuously non-malleable codes (cf. §3.3) to the case of TSS schemes.[6]

Definition 9 (Shared-value uniqueness)**.** Let $\Sigma = (\mathsf{Share}, \mathsf{Rec})$ be an n-party TSS with reconstruction threshold $\varrho \leq n$. We say that Σ satisfies shared-value uniqueness (SVU) if for all subsets $\mathcal{I} = \{i_1, \ldots, i_\varrho\} \subseteq [n]$, there exists $j^* \in [\varrho]$ such that for all shares $s_{i_1}, \ldots, s_{i_{j^*-1}}, s_{i_{j^*+1}}, \ldots, s_{i_\varrho}$, and for all $s_{i_{j^*}}, s'_{i_{j^*}}$, we have that either

$$m = \mathsf{Rec}(s_{i_1}, \ldots, s_{i_{j^*}}, \ldots, s_{i_\varrho}) = \mathsf{Rec}(s_{i_1}, \ldots, s'_{i_{j^*}}, \ldots, s_{i_\varrho}) = m', \tag{3}$$

where $m, m' \in \mathcal{M}$, or at least one of m, m' equals $\perp$.

Intuitively, the above property says that for every possible choice of an authorized set $\mathcal{I}$, there exists at least one index $i_{j^*} \in \mathcal{I}$, such that if we fix arbitrarily all the shares but the one in position i_{j^*}, the reconstruction process can possibly output a single outcome within the space of all valid messages. The theorem below says that SVU is necessary for achieving continuous non-malleability (without leakage) for threshold secret sharing, in the computational setting.

Theorem 2. *For any $n, \tau, \varrho \in \mathbb{N}$, with $\tau \leq \varrho \leq n$, every $(n, \tau, \varrho, 0)$-LR-CNMSS must also satisfy SVU.*

[5] We stress that the attacks described in the proof of Thm. 2 and Thm. 3 do not require to change the reconstruction set $\mathcal{T}$ among different queries, and thus even hold without considering concurrent reconstruction.

[6] As for MU, "perfect" SVU, rather than "computational" SVU, is wlog. in the plain model.

Proof. By contradiction, assume that there is a set $\mathcal{I} = \{i_1, \ldots, i_\varrho\}$ for which SVU does not hold. This means that for all $j^* \in [\varrho]$, there are strings $s_{i_1}, \ldots, s_{i_{j^*-1}}, s_{i_{j^*+1}}, \ldots, s_{i_\varrho}$ and $s_{i_{j^*}}, s'_{i_{j^*}}$, such that Eq. (3) does not hold, i.e.

$$\perp \neq m = \mathsf{Rec}(s_{i_1}, \ldots, s_{i_{j^*}}, \ldots, s_{i_\varrho}) \neq \mathsf{Rec}(s_{i_1}, \ldots, s'_{i_{j^*}}, \ldots, s_{i_\varrho}) = m' \neq \perp,$$

for some $m, m' \in \mathcal{M}$. Let now $(s^*_1, \ldots, s^*_n)$ be a target secret sharing in the experiment of Fig. 5, and denote by $\sigma_i = \log |\mathcal{S}_i|$ the size of the i-th share. For each $j^* \in [\varrho]$, consider the following sequence of (efficiently computable) tampering functions with hard-wired values $s_{i_1}, \ldots, s_{i_{j^*-1}}, s_{i_{j^*+1}}, \ldots, s_{i_\varrho}$, $s_{i_{j^*}}, s'_{i_{j^*}}$, and parameterized by a value $k \in [\sigma_{i_{j^*}}]$:

$$f_{i_j}(s^*_{i_j}) = \begin{cases} s_{i_j} & \text{if } j \neq j^* \\ s_{i_{j^*}} & \text{if } j = j^* \text{ and } s^*_{i_{j^*}}[k] = 0 \\ s'_{i_{j^*}} & \text{if } j = j^* \text{ and } s^*_{i_{j^*}}[k] = 1. \end{cases}$$

Consider a tampering query $(\mathcal{T} = \mathcal{I}, (f_1, \ldots, f_n))$, where f_i equals the identity function for each $i \in [n] \setminus \mathcal{T}$ (and is defined as above otherwise). We note that the answer to such a tampering query either equals m (if $s^*_{i_{j^*}}[k] = 0$) or m' (if $s^*_{i_{j^*}}[k] = 1$), and thus it allows to learn the value $s^*_{i_{j^*}}[k]$ without the risk of self-destruct.

Repeating the above query for each $k \in [\sigma_{i_{j^*}}]$, and for each $j^* \in [\varrho]$, reveals all the shares $(s^*_{i_1}, \ldots, s^*_{i_\varrho})$, which in turn clearly allows to break continuous non-malleability (as long as the messages m_0, m_1 are different from all the messages m, m' that contradict SVU). □

Notice that in the information-theoretic setting, when the privacy threshold τ equals the reconstruction threshold ϱ, and when considering the authorized set $\mathcal{I} = [\varrho]$, statistical privacy implies that for each $i^* \in [\varrho]$ there always exist shares $(s_1, \ldots, s_{i^*-1}, s_{i^*}, s_{i^*+1}, \ldots, s_\varrho)$ and $(s_1, \ldots, s_{i^*-1}, s'_{i^*}, s_{i^*+1}, \ldots, s_\varrho)$ that violate SVU. Hence, we obtain that CNMSS with the optimal parameters $\tau = \varrho$ is impossible in the information-theoretic setting, a fact recently established in [BS18].

Corollary 1 ([BS18])**.** *For any $n, \tau, \varrho \in \mathbb{N}$, with $\tau = \varrho \leq n$, there is no $(n, \tau, \varrho, 0)$-LR-CNMSS in the information-theoretic setting.*

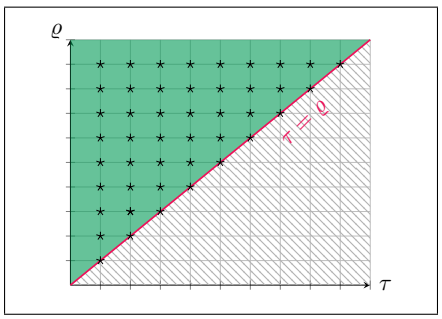

Figure 4: Possible parameters ϱ, τ of CNMSS. Values on the red line require computational assumptions.

Mind the gap. What if there is a small gap between the reconstruction threshold ϱ and the privacy threshold τ (e.g., $\tau \leq \varrho - 1$)? In such a case, the above impossibility result does not apply. For concreteness, let Σ be an $(n, \varrho - 1, \varrho)$-TSS and consider the reconstruction set $\mathcal{I} = [\varrho]$. By perfect privacy, since any collection of $\varrho - 2$ shares reveals no information on the shared value, for every possible sequence of shares $s_1, \ldots, s_{\varrho-2}$, and for every message $\hat{m} \in \mathcal{M}$, there always exist at least two shares $\hat{s}_{\varrho-1}, \hat{s}_\varrho$ such that running the reconstruction algorithm upon $(s_1, \ldots, s_{\varrho-2}, \hat{s}_{\varrho-1}, \hat{s}_\varrho)$ yields $\hat{m}$ as output.

However, there is no guarantee that a pair of shares $(\hat{s}'_{\varrho-1}, \hat{s}'_\varrho)$ yielding another message $\hat{m}' \neq \hat{m}$, and such that, e.g., $\hat{s}'_{\varrho-1} = \hat{s}_{\varrho-1}$, actually exists. This circumvents the attack described above. Put differently, whenever $\tau = \varrho - 1$, given any collection of $\varrho - 1$ shares, we can consider two cases (cf. also Fig. 4):

- There are at least two possible *valid* outcomes for the reconstruction procedure. In this case, a computationally unbounded attacker can still find a sequence of shares violating SVU, and thus continuous non-malleability requires computational assumptions.
- The shared value is information-theoretically determined, i.e. there exists an inefficient algorithm which can reconstruct the message. In this case, SVU is not violated, and thus it is plausible that TSS with perfect privacy and statistical continuous non-malleability exists.

4.3 Necessity of Self-Destruct

Finally, we show that continuous non-malleability as per Def. 8 is impossible without assuming self-destruct. This fact is reminiscent of a similar impossibility result for continuously non-malleable codes [FMNV14], and tamper-resilient cryptography [GLM+04].

Theorem 3. *For any $n, \tau, \varrho \in \mathbb{N}$, with $\tau \leq \varrho \leq n$, there is no $(n, \tau, \varrho, 0)$-LR-CNMSS without assuming the self-destruct capability.*

Proof. Fix an arbitrary (n, τ, ϱ)-TSS Σ. For simplicity we assume that Σ has perfect correctness, and that the shares are all of the same size $\log |\mathcal{S}_i| = \sigma$. A generalization is immediate. Consider the following attacker, attempting to break Def. 8 w.r.t. messages $m_0 = 01 \cdots 1$ and $m_1 = 1 \cdots 1$:

1. Initialize $\hat{s}_1, \ldots, \hat{s}_\varrho$ to the empty string.
2. For each $i \in [\varrho]$ and $k \in [\sigma]$, query the tampering oracle $\mathcal{O}_{\mathsf{nmss}}$ with authorized set $\mathcal{T} = [\varrho]$, and with a sequence of tampering functions

$$(\underbrace{\hat{s}_1, \ldots, \hat{s}_{i-1}}_{\#i-1}, \mathsf{set}_{i,k}, \underbrace{\mathsf{id}, \ldots, \mathsf{id}}_{\#\varrho-i}), \tag{4}$$

   where id is the identity function and $\mathsf{set}_{i,k}$ is the function that takes as input the i-th share and sets the bit in position k to zero. Hence:

   (a) If the answer from the oracle is 💔, let $\hat{s}_i[k] = 0$.
   (b) Else, in case the answer from the oracle is a value in $\mathcal{M} \cup \{\perp\}$, let $\hat{s}_i[k] = 1$.

3. Run $\mathsf{Rec}(\hat{s}_1, \ldots, \hat{s}_\varrho)$, and if the result is $m_{b'}$ return b', and otherwise abort.

Let $(s_1, \ldots, s_n)$ be the target secret sharing in the experiment $\mathbf{CNMSS}_{\Sigma,\mathsf{A}}(\lambda, m_0, m_1, b)$ (stripped of the self-destruct capability), and denote by **Abort** the event that A aborts in step 3. For each $i \in [\varrho]$, and $k \in [\sigma]$, let $\hat{m}_{i,k}$ be the message reconstructed inside the $\mathcal{O}_{\mathsf{nmss}}$ oracle corresponding to the tampering query of Eq. (4). Note that, by perfect correctness of the secret sharing scheme, whenever the tampering function $\mathsf{set}_{i,k}$ causes the execution of step 2b, we must conclude that it flipped the k-th bit of the i-th share, so that A correctly sets $\hat{s}_i[k] = 1$. On the other hand, whenever the attacker sets $\hat{s}_i[k] = 0$, only three cases are possible: Either (i) $s_i[k] = 0$, or (ii) $s_i[k] = 1$ and $\hat{m}_{i,k} = m_b$, or (iii) $s_i[k] = 1$ and $\hat{m}_{i,k} = m_{1-b}$.

We claim that for all values of $b \in \{0, 1\}$, and for each tampering query of Eq. (4), case (iii) only happens with negligible probability. To see this, fix $b = 0$. Then case (iii) means that by setting to 0 the k-th bit of the i-th share of a secret sharing of $m_0 = 01 \cdots 1$ yields with non-negligible probability a mauled secret sharing whose reconstruction (w.r.t. authorized set $\mathcal{T} = [\varrho]$) is a related value $m_1 = 1 \cdots 1$. However, the latter violates even one-time non-malleability of Σ w.r.t. messages $m'_0 = m_0$ and, say, $m'_1 = 0 \cdots 0$.

Thus, by a union bound, we conclude that $\mathbb{P}[\mathbf{Abort}] \in \mathtt{negl}(\lambda)$. Additionally, conditioning on **Abort** not happening, we have that the bit b' returned by A always (i.e., with probability one) equals the bit b in experiment $\mathbf{CNMSS}_{\Sigma,\mathsf{A}}(\lambda, m_0, m_1, b)$. Hence, A has overwhelming distinguishing advantage, concluding the proof. □

5 A Scheme with Poor Rate

5.1 Main Construction

Before describing our scheme, we introduce some useful notation. The shares will be of the form $s_i^* = (s_{\mathcal{H},i})_{\mathcal{H}\in\binom{[n]}{2}}$ (see Fig. 5), where $i \in [n]$. Given a set $\mathcal{A} \subseteq [n]$, we identify with $\hat{\mathcal{A}}$ the first two indices of $\mathcal{A}$.

Our threshold secret sharing $\Sigma^* = (\mathsf{Share}^*, \mathsf{Rec}^*)$, which is formally depicted in Fig. 5, is based upon the following ingredients:

- An authenticated secret-key encryption (AE) scheme $\Pi = (\mathsf{AEnc}, \mathsf{ADec})$ (cf. §3.2.2), with message space $\mathcal{M}$, ciphertext space $\mathcal{C}$, and key space $\mathcal{K} = \{0,1\}^\lambda$.
- An $(n-2)$-party secret sharing scheme $\Sigma = (\mathsf{Share}, \mathsf{Rec})$, with reconstruction threshold equal to $\varrho - 3$, message space $\mathcal{C}$, and share space $\mathcal{S}^{n-2}$ (cf. §3.2.1).
- A split-state encoding $\Gamma = (\mathsf{Enc}, \mathsf{Dec})$, with message space $\mathcal{K}$ and codeword space $\mathcal{L} \times \mathcal{R}$ (cf. §3.3).

5.2 Security Analysis

Theorem 4. *Let $n, \varrho, \ell, \ell^* \in \mathbb{N}$ be such that $n \geq \varrho > 2$. Assuming that Π is a secure AE scheme, that Σ is a $(n-2, \varrho-3, \varrho-3)$-TSS with perfect threshold privacy and with the special reconstruction property, and that Γ is an ℓ-LR-CNMC with symmetric decoding and with codewords uniformity, the secret sharing scheme Σ^* of Fig. 3 is an $(n, \varrho-1, \varrho, \ell^*)$-LR-CNMSS, as long as $\ell = \ell^* + 2\gamma + O(\log\lambda)$ where $\gamma = \log|\mathcal{C}|$ is the size of a ciphertext under Π.*

Basic Construction of LR-CNMSS

Let $\Pi = (\mathsf{AEnc}, \mathsf{ADec})$, $\Sigma = (\mathsf{Share}, \mathsf{Rec})$, and $\Gamma = (\mathsf{Enc}, \mathsf{Dec})$ be as described in the text. Consider the following construction of an n-party secret sharing $\Sigma^* = (\mathsf{Share}^*, \mathsf{Rec}^*)$ with reconstruction threshold $\varrho \leq n$, and message space $\mathcal{M}^* = \mathcal{M}$.

Sharing function $\mathsf{Share}^*(m)$**:** The secret sharing of a message $m \in \mathcal{M}^*$ is a collection of shares $s^* = (s_1^*, \ldots, s_n^*)$, where $s_i^* = (s_{\mathcal{H},i})_{\mathcal{H}\in\binom{[n]}{2}}$ and for any $\mathcal{H} = \{h_1, h_2\} \in \binom{[n]}{2}$ the share $s_{\mathcal{H},i}$ is computed following the steps below:

1. Let $\bar{\mathcal{H}} = [n] \setminus \mathcal{H} = \{h_3, \ldots, h_n\}$;
2. Sample $\kappa_{\mathcal{H}} \leftarrow\!\!\$\ \mathcal{K}$ and run $c_{\mathcal{H}} \leftarrow\!\!\$\ \mathsf{AEnc}(\kappa_{\mathcal{H}}, m)$;
3. Compute $(s_{\mathcal{H},h_1}, s_{\mathcal{H},h_2}) \leftarrow\!\!\$\ \mathsf{Enc}(\kappa_{\mathcal{H}})$ and $(s_{\mathcal{H},h_3}, \ldots, s_{\mathcal{H},h_n}) \leftarrow\!\!\$\ \mathsf{Share}(c_{\mathcal{H}})$.

Reconstruction function $\mathsf{Rec}^*(s_{\mathcal{I}}^*)$: Let $\mathcal{I} = \{i_1, \ldots, i_\varrho\}$. Wlog. we assume that the set $\mathcal{I}$ is ordered and that is made of exactly ϱ indices. (If not, we can just order it and use only the first ϱ indices.)

1. Let $\hat{\mathcal{I}} = \{i_1, i_2\}$, and parse $s_{\mathcal{I}}^* = (s_{i_1}^*, \ldots, s_{i_\varrho}^*)$, where for each $j \in [\varrho]$ we have $s_{i_j}^* = (s_{\mathcal{H},i_j})_{\mathcal{H}\in\binom{[n]}{2}}$;
2. Compute $\kappa = \mathsf{Dec}(s_{\hat{\mathcal{I}},i_1}, s_{\hat{\mathcal{I}},i_2})$, and for sets $\mathcal{A}_1 = \{i_3, \ldots, i_{\varrho-1}\}$ and $\mathcal{A}_2 = \{i_4, \ldots, i_\varrho\}$ let $c_1 = \mathsf{Rec}((s_{\hat{\mathcal{I}},a})_{a\in\mathcal{A}_1})$ and $c_2 = \mathsf{Rec}((s_{\hat{\mathcal{I}},a})_{a\in\mathcal{A}_2})$;
3. If $c_1 \neq c_2$ output $\perp$, else let $c = c_1 = c_2$ and return $m = \mathsf{ADec}(\kappa, c)$.

Figure 5: A construction of leakage-resilient continuously non-malleable secret sharing for threshold access structures, in the computational setting.

Proof. We need to prove that Σ^* satisfies both threshold privacy and continuous non-malleability. Both proofs proceed with a hybrid argument enumerating over all sets in $\binom{[n]}{2}$. We assume a fixed order between these sets, namely, let $N = \binom{n}{2}$ and let $\mathcal{H}_1, \mathcal{H}_2, \ldots, \mathcal{H}_N$ be such an order.

Threshold privacy. $\tau^* \leq \varrho - 1$ The goal is to show that for all messages $m_0, m_1 \in \mathcal{M}^*$, and for all unauthorized subset $\mathcal{U} = \{u_1, \ldots, u_{\tau^*-1}\}$, with $\tau^* \leq \varrho - 1$, the two ensembles $\{(\mathsf{Share}^*(1^\lambda, m_0))_\mathcal{U}\}_{\lambda\in\mathbb{N}}$ and $\{(\mathsf{Share}^*(1^\lambda, m_1))_\mathcal{U}\}_{\lambda\in\mathbb{N}}$ are computationally indistinguishable. Fix $m_0, m_1 \in \mathcal{M}$, and $\mathcal{U}$. Consider the following sequence of hybrid experiments.

Hybrid $\mathbf{H}_0(\lambda, b)$: Distributed identically to $\mathsf{Share}^*(1^\lambda, m_b))_\mathcal{U}$.
Hybrid $\mathbf{H}'_j(\lambda, b)$: Identical to $\mathbf{H}_{j-1}(\lambda, b)$, but if $\mathcal{H}_j \not\subset \mathcal{U}$ then $(s_{\mathcal{H}_j,h_1}, s_{\mathcal{H}_j,h_2}) \leftarrow\!\!\$\ \mathsf{Enc}(0^\lambda)$.
Hybrid $\mathbf{H}_j(\lambda, b)$: Identical to $\mathbf{H}'_j(\lambda, b)$, but $c_{\mathcal{H}_j} \leftarrow\!\!\$\ \mathsf{AEnc}(\kappa_{\mathcal{H}_j}, \hat{m})$ for a fixed $\hat{m} \in \mathcal{M}\backslash\{m_0, m_1\}$.

Lemma 3. *$\forall j \in [N]$, $\forall b \in \{0,1\}$: $\{\mathbf{H}'_j(\lambda, b)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{H}_{j-1}(\lambda, b)\}_{\lambda\in\mathbb{N}}$.*

Proof. Fix $j \in [N]$ and $b \in \{0,1\}$. We reduce to the 2-threshold privacy of Γ (cf. Remark 2). In particular, if the intersection $\mathcal{H}_j \cap \mathcal{U}$ has size ≥ 1, then by threshold privacy of the underlying CNMC the hybrids $\mathbf{H}_{j-1}$ and $\mathbf{H}'_j$ are computationally indistinguishable. On the other hand, if the size of the intersection is 0, then the hybrids are equivalently distributed. □

Lemma 4. *$\forall j \in [N]$, $\forall b \in \{0,1\}$: $\{\mathbf{H}'_j(\lambda, b)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{H}_j(\lambda, b)\}_{\lambda\in\mathbb{N}}$.*

Proof. Fix $j \in [N]$ and $b \in \{0,1\}$. We either reduce to the perfect privacy of Σ, or to the semantic security of Π. In particular, we consider two cases:

- If $\mathcal{H}_j \subset \mathcal{U}$, then for any randomness r, key $\kappa_{\mathcal{H}_j}$, and ciphertexts $c^0_{\mathcal{H}_j}, c^1_{\mathcal{H}_j}$ such that $c^b_{\mathcal{H}_j} = \mathsf{AEnc}(\kappa_{\mathcal{H}_j}, m_b; r)$, the distributions $(\mathsf{Share}(c^0_{\mathcal{H}_j}))_{\mathcal{U}\setminus\mathcal{H}_j}$ and $(\mathsf{Share}(c^1_{\mathcal{H}_j}))_{\mathcal{U}\setminus\mathcal{H}_j}$ are equivalently distributed, as $|\mathcal{U} \setminus \mathcal{H}_j| < (\rho - 1) - 2 = \rho - 3$.
- Else, if $\mathcal{H}_j \not\subset \mathcal{U}$, we have that the secret key $\kappa_{\mathcal{H}_j}$, which is used to compute the ciphertext $c_{\mathcal{H}_j}$, is not included in the adversary's view (given the change introduced in $\mathbf{H}'_j$), and therefore by semantic security the adversary cannot distinguish an encryption of m_b from one of $\hat{m}$. The reduction is straightforward, and therefore omitted.

□

It is clear that the ensembles $\{\mathbf{H}_N(\lambda, 0)\}_{\lambda\in\mathbb{N}}$ and $\{\mathbf{H}_N(\lambda, 1)\}_{\lambda\in\mathbb{N}}$ are identically distributed, which concludes the proof of threshold privacy.

Continuous non-malleability. We need to show that for all messages $m_0, m_1 \in \mathcal{M}$, and for all PPT ℓ^*-admissible adversaries A, the following holds for the experiment defined in Fig. 3:

$$\{\mathbf{CNMSS}_{\Sigma^*,\mathsf{A}}(\lambda, m_0, m_1, 0)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{CNMSS}_{\Sigma^*,\mathsf{A}}(\lambda, m_0, m_1, 1)\}_{\lambda\in\mathbb{N}}.$$

Fix $m_0, m_1 \in \mathcal{M}$. Consider the hybrids defined below.

Hybrid $\mathbf{H}_0(\lambda, b)$: Identical to the experiment $\mathbf{CNMSS}_{\Sigma^*,\mathsf{A}}(\lambda, m_0, m_1, b)$, except that we additionally sample a dummy random message. Specifically, the experiment proceeds as follows (the difference with the original game are underlined in red).

1. Sample $\hat{m} \leftarrow\!\!\$\ \mathcal{M}$;
2. For each $\mathcal{H} \in \binom{[n]}{2}$, let $\mathcal{H} = \{h_1, h_2\}$ and $\bar{\mathcal{H}} = \{h_3, \ldots, h_n\}$. Hence:
   (a) Sample $\kappa_\mathcal{H} \leftarrow\!\!\$\ \mathcal{K}$;
   (b) Compute $c_\mathcal{H} \leftarrow\!\!\$\ \mathsf{AEnc}(\kappa_\mathcal{H}, m)$;

(c) Compute $(s_{\mathcal{H},h_1}, s_{\mathcal{H},h_2}) \leftarrow\!\!\$\ \mathsf{Enc}(\kappa_{\mathcal{H}})$
(d) Compute $(s_{\mathcal{H},h_3}, \ldots, s_{\mathcal{H},h_n}) \leftarrow\!\!\$\ \mathsf{Share}(c_{\mathcal{H}})$.

3. Set $s_i^* = (s_{\mathcal{H},i})_{\mathcal{H} \in \binom{[n]}{2}}$ for all $i \in [n]$.
4. Upon input a leakage query (i, g) from A, return $g(s_i^*)$.
5. Upon input a tampering query $(\mathcal{T}, (f_1, \ldots, f_n))$ from A, where wlog. we assume that $\mathcal{T} = \{t_1, \ldots, t_\varrho\}$ is ordered,[7] answer as follows:
   (a) For all $j \in [\varrho]$, compute $f_{t_j}(s_{t_j}^*) = \tilde{s}_{t_j}^* = (\tilde{s}_{\mathcal{H},t_j})_{\mathcal{H} \in \binom{[n]}{2}}$;
   (b) Compute $\tilde{\kappa} = \mathsf{Dec}(\tilde{s}_{\hat{\mathcal{T}},t_1}, \tilde{s}_{\hat{\mathcal{T}},t_2})$;
   (c) For the subsets $\mathcal{A}_1 = \{t_3, \ldots, t_{\varrho-1}\}$ and $\mathcal{A}_2 = \{t_4, \ldots, t_\varrho\}$, compute $\tilde{c}_1 = \mathsf{Rec}((\tilde{s}_{\hat{\mathcal{T}},a})_{a \in \mathcal{A}_1})$ and $\tilde{c}_2 = \mathsf{Rec}((\tilde{s}_{\hat{\mathcal{T}},a})_{a \in \mathcal{A}_2})$;
   (d) If $\tilde{c}_1 \neq \tilde{c}_2$, then return $\perp$ and self-destruct;
   (e) Else, let $\tilde{c}$ be the (unique) reconstructed ciphertext and $\tilde{m} = \mathsf{ADec}(\tilde{\kappa}, \tilde{c})$;
   (f) If $\tilde{m} = \perp$, then return $\perp$ and self-destruct; else if $\tilde{m} \in \{m_0, m_1, \hat{m}\}$ return ♥, and otherwise return $\tilde{m}$.

Hybrid $\mathbf{H}_j(\lambda, b)$: Identical to $\mathbf{H}_{j-1}(\lambda, b)$, except that the distribution of the target secret sharing corresponding to the hybrid set $\mathcal{H}_j = \{h_1, h_2\}$ is modified as follows:

- In step 2a, additionally sample $\hat{\kappa}_{\mathcal{H}_j} \leftarrow\!\!\$\ \mathcal{K}$;
- In step 2b, let $c_{\mathcal{H}_j} \leftarrow\!\!\$\ \mathsf{AEnc}(\kappa_{\mathcal{H}_j}, \hat{m})$;
- In step 2c, let $(s_{\mathcal{H}_j,h_1}, s_{\mathcal{H}_j,h_2}) \leftarrow\!\!\$\ \mathsf{Enc}(\hat{\kappa}_{\mathcal{H}_j})$;
- In step 5e, additionally check if $\tilde{\kappa} = \hat{\kappa}_{\mathcal{H}_j}$ but $\tilde{c} \neq c_{\mathcal{H}_j}$: in case that happens, return $\perp$ and self-destruct, else if $\tilde{\kappa} = \hat{\kappa}_{\mathcal{H}_j}$ and $\tilde{c} = c_{\mathcal{H}_j}$ set $\tilde{m} =$ ♥.

In order to prove the indistinguishability between $\mathbf{H}_j$ and $\mathbf{H}_{j-1}$ we also introduce the following intermediate hybrids.

Hybrid $\mathbf{H}_{j,1}(\lambda, b)$: Identical to $\mathbf{H}_{j-1}(\lambda, b)$, but where only step 2a and step 2c are modified.
Hybrid $\mathbf{H}_{j,2}(\lambda, b)$: Identical to $\mathbf{H}_{j,1}(\lambda, b)$, but where step 5e is modified.
Hybrid $\mathbf{H}_{j,3}(\lambda, b)$: Identical to $\mathbf{H}_{j,2}(\lambda, b)$ but where step 2b is modified.

Lemma 5. *$\forall j \in [N]$, $\forall b \in \{0,1\}$: $\{\mathbf{H}_{j-1}(\lambda, b)\}_{\lambda \in \mathbb{N}} \approx_c \{\mathbf{H}_{j,1}(\lambda, b)\}_{\lambda \in \mathbb{N}}$.*

Proof. By contradiction, assume that there exist some $j \in [N]$, a value $b \in \{0,1\}$, and an ℓ^*-admissible PPT adversary A able to distinguish the interactive experiments $\mathbf{H}_{j-1}(\lambda, 0)$ and $\mathbf{H}_{j,1}(\lambda, b)$ with non-negligible probability. We construct an ℓ-admissible PPT adversary B, for ℓ as in the statement of the theorem, attacking the code Γ in the experiment of Def. 4 w.r.t. messages $\kappa_{\mathcal{H}_j}, \hat{\kappa}_{\mathcal{H}_j} \leftarrow\!\!\$\ \mathcal{K}$. The reduction B makes use of different types of leakage and tampering queries, which we summarize in Fig. 6.

Adversary B (playing the game of Fig. 2):

1. For each set $\mathcal{H} \in \binom{[n]}{2} \setminus \mathcal{H}_j$, run step 2 as described in $\mathbf{H}_{j-1}(\lambda, b)$.
2. For the hybrid set $\mathcal{H}_j = \{h_1, h_2\}$, run step 1, 2a, 2b, and 2d, as described in $\mathbf{H}_{j-1}(\lambda, b)$. (While step 2c is not executed.)
3. Note that by executing the above steps, the reduction knows all the shares s_i^* except for $s_{h_1}^*$ and $s_{h_2}^*$, for which the values $s_{\mathcal{H}_j,h_1}, s_{\mathcal{H}_j,h_2}$ are missing, and will be defined through the leakage and tampering oracles. We write $\hat{s}_{h_1}^*$ and $\hat{s}_{h_2}^*$ for the partial information known about the shares $s_{h_1}^*$ and $s_{h_2}^*$.
4. Upon input a leakage query from A, of the form (i, g), answer as follows:

[7] Assuming that the size of each reconstruction set $\mathcal{T}$ is exactly equal to ϱ is wlog. due to the fact that the reconstruction algorithm in Fig. 5 only uses the first ϱ shares.

- If $i \notin \mathcal{H}_j$, return $g(s_i^*)$, where the value s_i^* is known by the reduction;
- Else, if $i \in \mathcal{H}_j = \{h_1, h_2\}$, forward to the leakage oracle either $(\mathtt{left}, g')$, in case $i = h_1$, or $(\mathtt{right}, g')$, in case $i = h_2$, where the function g' hard-wires the value $\hat{s}_i^*$, and outputs $g(s_i^*)$.

5. Upon input a tampering query from A, of the form $(\mathcal{T}, (f_1, \ldots, f_n))$ for $\mathcal{T} = \{t_1, \ldots, t_\varrho\}$, compute locally $\tilde{s}_t = (\tilde{s}_{\mathcal{H},t})_{\mathcal{H} \in \binom{[n]}{2}} = f_t(s_t^*)$ for each $t \in \mathcal{T} \setminus \mathcal{H}_j$, and then answer as follows.
   (a) Case $\mathcal{H}_j \subset \mathcal{T}$ and $|\mathcal{H}_j \cap \{t_1, t_2\}| = 1$:
      i. Forward the query below to the tampering oracle:

$$(f_{\mathsf{left}}, f_{\mathsf{right}}) = \begin{cases} (f_{\mathsf{move}}^{t_1}, f_{\mathsf{set}}^{t_2}) & \text{if } t_1 = h_1 \\ (f_{\mathsf{set}}^{t_1}, f_{\mathsf{move}}^{t_2}) & \text{if } t_2 = h_2 \\ (f_{\mathsf{set}}^{t_2}, f_{\mathsf{move}}^{t_1}) & \text{if } t_1 = h_2 \\ (f_{\mathsf{move}}^{t_2}, f_{\mathsf{set}}^{t_1}) & \text{if } t_2 = h_1. \end{cases} \tag{5}$$

      ii. Let $\tilde{\kappa} \in \{0,1\}^\lambda \cup \{\bot, \heartsuit\}$ be the answer from the oracle. If $\tilde{\kappa} = \bot$, output $\bot$ and self-destruct. Else, if $\tilde{\kappa} = \heartsuit$, then set $\tilde{\kappa} = \kappa_{\hat{\mathcal{T}}}$.
      iii. Let $\tilde{\mathcal{A}}_1 = \{t_3, \ldots, t_\varrho\} \setminus \mathcal{H}_j$; compute locally $\tilde{c}_1 = \mathsf{Rec}((\tilde{s}_{\hat{\mathcal{T}},a})_{a \in \tilde{\mathcal{A}}_1})$.
      iv. Let $\tilde{\mathcal{A}}_2 \subset \{t_3, \ldots, t_\varrho\}$ be such that $|\tilde{\mathcal{A}}_2| = \varrho - 4$, and $\mathcal{H}_j \cap \tilde{\mathcal{A}}_2 \neq \emptyset$; compute $\tilde{c}_2$ using the leakage oracle. Specifically, if $h_1 \in \tilde{\mathcal{A}}_2$ then send the leakage query $(\mathtt{left}, g_{\mathsf{rec}}^{\tilde{\mathcal{A}}_2, h_1})$, whereas if $h_2 \in \tilde{\mathcal{A}}_2$ then send the leakage query $(\mathtt{right}, g_{\mathsf{rec}}^{\tilde{\mathcal{A}}_2, h_2})$.
      v. If $\tilde{c}_1 \neq \tilde{c}_2$, then return $\bot$ and self-destruct; else set $\tilde{m} = \mathsf{ADec}(\tilde{\kappa}, \tilde{c}_1)$.
   (b) Case $\mathcal{T} \cap \mathcal{H}_j \neq \emptyset$ and $\mathcal{H}_j \cap \{t_1, t_2\} = \emptyset$:
      i. Compute $\tilde{\kappa} = \mathsf{Dec}(\tilde{s}_{\hat{\mathcal{T}},t_1}, \tilde{s}_{\hat{\mathcal{T}},t_2})$;
      ii. In case $|\mathcal{T} \cap \mathcal{H}_j| = 1$, then let $\tilde{\mathcal{A}}_1 = \{t_3, \ldots, t_\varrho\} \setminus \mathcal{H}_j$, compute (locally) $\tilde{c}_1 = \mathsf{Rec}((\tilde{s}_{\hat{\mathcal{T}},t})_{t \in \tilde{\mathcal{A}}_1})$, and let $\tilde{\mathcal{A}}_2 \subset \mathcal{T}$ be such that $|\tilde{\mathcal{A}}_2| = \varrho - 4$ and either $h_1 \in \tilde{\mathcal{A}}_2$ or $h_2 \in \tilde{\mathcal{A}}_2$.
         - If $h_1 \in \tilde{\mathcal{A}}_2$, send the leakage query $(\mathtt{left}, g_{\mathsf{rec}}^{\tilde{\mathcal{A}}_2, h_1})$;
         - Else, send the leakage query $(\mathtt{right}, g_{\mathsf{rec}}^{\tilde{\mathcal{A}}_2, h_2})$.

         In both cases, let $\tilde{c}_2$ be the answer obtained from the leakage oracle
      iii. In case $|\mathcal{T} \cap \mathcal{H}_j| = 2$, then let $\tilde{\mathcal{A}} = \{t_3, \ldots, t_\varrho\} \setminus \mathcal{H}_j$, and define $\tilde{\mathcal{A}}_1 = \tilde{\mathcal{A}} \cup \{h_1\}$ and $\tilde{\mathcal{A}}_2 = \tilde{\mathcal{A}} \cup \{h_2\}$.
         - Send the leakage query $(\mathtt{left}, g_{\mathsf{rec}}^{\tilde{\mathcal{A}}_1, h_1})$;
         - Send the leakage query $(\mathtt{right}, g_{\mathsf{rec}}^{\tilde{\mathcal{A}}_2, h_2})$.

         Denote by $\tilde{c}_1$ and $\tilde{c}_2$ the corresponding answers obtained from the leakage oracle.
      iv. If $\tilde{c}_1 \neq \tilde{c}_2$, then return $\bot$ and self-destruct. Else, set $\tilde{m} = \mathsf{ADec}(\tilde{\kappa}, \tilde{c}_1)$.
   (c) Case $\mathcal{H}_j = \{t_1, t_2\}$:
      i. Send the tampering query $(f_{\mathsf{move}}^{h_1}, f_{\mathsf{move}}^{h_2})$, and denote by $\tilde{\kappa}$ the corresponding output. If $\tilde{\kappa} = \bot$, output $\bot$ and self-destruct. Else, if $\tilde{\kappa} = \heartsuit$, then set $\tilde{\kappa} = \kappa_{\hat{\mathcal{T}}}$.
      ii. Attempt to compute the unique ciphertext $\tilde{c}$ as described in the original reconstruction algorithm, using the tampered shares $\tilde{s}_{t_3}, \ldots, \tilde{s}_{t_\varrho}$.
      iii. If the ciphertext is not unique, output $\bot$ and self-destruct. Else, set $\tilde{m} = \mathsf{ADec}(\tilde{\kappa}, \tilde{c}_1)$.

Tampering functions:

- For $X \in \{L, R\}$, let $f^t_{\mathsf{move}}(X)$ be the function that sets $s_{\mathcal{H}_j,t} = X$, computes $\tilde{s}^*_t = (\tilde{s}_{\mathcal{H},t})_{\mathcal{H}\in\binom{[n]}{2}} = f_t(s^*_t)$, and returns $\tilde{s}_{\hat{\mathcal{T}},t}$.
- For $X \in \{L, R\}$, let $f^t_{\mathsf{set}}(X)$ be the function that ignores the input and returns $\tilde{s}_{\hat{\mathcal{T}},t}$;

Leakage Functions:

- For $X \in \{L, R\}$, let $g^{\mathcal{A},t}_{\mathsf{rec}}(X)$ be the function that sets $s_{\mathcal{H}_j,t} = X$, computes $\tilde{s}^*_t = f_t(s^*_t)$, and outputs $\mathsf{Rec}((\tilde{s}_{\hat{T},a})_{a\in\mathcal{A}\setminus\{t\}}, \tilde{s}_{\hat{\mathcal{T}},t})$.

Figure 6: Tampering and leakage functions used by the reduction B in the proof of Thm. 4. We implicitly assume that each function has additionally hard-coded the shares $\tilde{s}^*_{t_3}, \ldots, \tilde{s}^*_{t_\varrho}$; the tampering function f^t_{set} is well defined when $t \notin \{h_1, h_2\}$.

(d) Case $\mathcal{T} \cap \mathcal{H}_j = \emptyset$:
   i. This is the simplest case, as the entire reconstruction procedure can be run locally using the shares $\tilde{s}_{t_1}, \ldots, \tilde{s}_{t_\varrho}$, which are known to the reduction.
   ii. If an error is generated while running the reconstruction algorithm, output $\perp$ and self-destruct, else let $\tilde{m}$ be the reconstructed message.

(e) Whenever $\tilde{m} = \perp$, return $\perp$ and self-destruct. Otherwise, if $\tilde{m} \in \{m_0, m_1, \hat{m}\}$ output ♥, and else output $\tilde{m}$.

6. Return whatever A outputs.

Next, we show that the reduction B perfectly simulates the view of A in the hybrid experiment. Namely,

$$\{\mathbf{CNMSS}_{\Sigma,\mathsf{B}}(\lambda, \kappa_{\mathcal{H}_j}, \hat{\kappa}_{\mathcal{H}_j}, 0)\}_{\lambda\in\mathbb{N}} \equiv \{\mathbf{H}_{j-1}(\lambda, b)\}_{\lambda\in\mathbb{N}}$$
$$\{\mathbf{CNMSS}_{\Sigma,\mathsf{B}}(\lambda, \kappa_{\mathcal{H}_j}, \hat{\kappa}_{\mathcal{H}_j}, 1)\}_{\lambda\in\mathbb{N}} \equiv \{\mathbf{H}_{j,1}(\lambda, b)\}_{\lambda\in\mathbb{N}},$$

where $\kappa_{\mathcal{H}_j}, \hat{\kappa}_{\mathcal{H}_j} \leftarrow\!\!\$\ \mathcal{K}$.

First, note that the reduction perfectly emulates the distribution of the target secret sharing $s^* = (s^*_1, \ldots, s^*_n)$. This is because all the shares s^*_i with $i \in [n] \setminus \mathcal{H}_j$ are computed locally by the reduction, whereas the missing shares $s^*_{h_1}$ and $s^*_{h_2}$ are accessible through the leakage and tampering oracles given the values $\hat{s}^*_{h_1}$ and $\hat{s}^*_{h_2}$, which are known by the reduction. In particular, if the target encoding $(L, R) = (s_{\mathcal{H}_j,h_1}, s_{\mathcal{H}_j,h_2})$ is an encoding of $\kappa_{\mathcal{H}_j}$ (resp. $\hat{\kappa}_{\mathcal{H}_j}$), then s^* as emulated by the reduction is distributed exactly as in $\mathbf{H}_{j-1}(\lambda, b)$ (resp. $\mathbf{H}_{j,1}(\lambda, b)$). The latter implies that the answer to leakage queries, as described in step 4, is distributed correctly.

It remains to analyze how B treats A's tampering queries. Given a generic tampering query $(\mathcal{T}, (f_1, \ldots, f_n))$, we consider 4 cases as described in the reduction.

(a) Case $\mathcal{H}_j \subset \mathcal{T}$ and $|\mathcal{H}_j \cap \{t_1, t_2\}| = 1$. The reduction B reconstructs the ciphertexts $\tilde{c}_1$ and $\tilde{c}_2$ using the sets $\tilde{\mathcal{A}}_1$ and $\tilde{\mathcal{A}}_2$, which might be different from the sets $\mathcal{A}_1$ and $\mathcal{A}_2$ used in the hybrids. However, whenever $\tilde{c}_1 = \tilde{c}_2$, by the special reconstruction property of Σ, we can conclude that for any subset of $\{t_3, \ldots, t_\varrho\}$ with cardinality $\varrho - 3$ the reconstructed ciphertexts would be the same, and thus the ciphertexts obtained by the reduction are the same as obtained in the hybrids.

Conversely, if $\tilde{c}_1 \neq \tilde{c}_2$, then also the reconstruction with the sets $\mathcal{A}_1$ and $\mathcal{A}_2$ must yield different ciphertexts. Therefore, whenever the hybrid returns $\perp$ to A due to a failure of the check performed in step 5d, the reduction B outputs $\perp$ too.
Next, we claim that the the output of B's tampering oracle is distributed exactly as $\mathsf{Dec}(\tilde{s}_{\hat{\mathcal{T}},t_1}, \tilde{s}_{\hat{\mathcal{T}},t_2})$ as computed in the hybrid, yielding a perfect simulation of the tampered key $\tilde{\kappa}$. The latter can be seen by considering the same cases as done by the reduction in Eq. (5).

- The first and second case can be analyzed together. Here, we have $t_k = h_k$ for some $k \in \{1,2\}$. While in this case the reduction cannot compute $s^*_{t_k}$ locally, as it does not know the value $s_{\mathcal{H}_j,t_k}$, the values $\tilde{s}_{\mathcal{T},t_{3-k}}$ can be computed locally, since the index $3-k \in \{1,2\}$ is not in the set $\mathcal{H}_j$. Moreover, the tampering function $f^{t_{3-k}}_{\mathsf{set}}$ sets one part of the tampered codeword to such a value.
  Furthermore, the tampering function $f^{t_k}_{\mathsf{move}}$ computes the value $\tilde{s}_{\mathcal{T},t_k}$. Therefore, by sending the tampering query $(f^{t_1}_{\mathsf{move}}, f^{t_2}_{\mathsf{set}})$, in case $t_1 = h_1$, and $(f^{t_1}_{\mathsf{set}}, f^{t_2}_{\mathsf{move}})$, in case $t_2 = h_2$, the tampering oracle returns exactly the same as $\mathsf{Dec}(\tilde{s}_{\hat{\mathcal{T}},t_1}, \tilde{s}_{\hat{\mathcal{T}},t_2})$.
- The third and fourth case can also be analyzed together. Here, we have $t_k = h_{3-k}$ for some $k \in \{1,2\}$. This case is slightly more complex because the share $s_{\mathcal{T},t_k}$ contains the left part (resp. the right part) of the target codeword relative to the set $\mathcal{T}$, and the right part (resp. the left part) of the target codeword relative to the set $\mathcal{H}$. In particular, when $t_1 = h_2$, the reduction sends the tampering query $(f^{t_2}_{\mathsf{set}}, f^{t_1}_{\mathsf{move}})$, which yields the same as
$$\mathsf{Dec}(\tilde{s}_{\hat{\mathcal{T}},t_2}, \tilde{s}_{\hat{\mathcal{T}},t_1}) = \mathsf{Dec}(\tilde{s}_{\hat{\mathcal{T}},t_1}, \tilde{s}_{\hat{\mathcal{T}},t_2}),$$
  thanks to the fact that the code Γ satisfies symmetric decoding. An identical argument holds for the case $t_2 = h_1$.

Note that the above analysis implies that whenever the hybrid experiment returns $\perp$ due to an invalid decoding of $(\tilde{s}_{\hat{\mathcal{T}},t_1}, \tilde{s}_{\hat{\mathcal{T}},t_2})$, the reduction B returns $\perp$ too, and the same also happens whenever the unique reconstructed ciphertext $\tilde{c}$ decrypts to $\perp$ (so that we covered all cases in which the reduction returns $\perp$ to A).

(b) Case $\mathcal{T} \cap \mathcal{H}_j \neq \emptyset$ and $\mathcal{H}_j \cap \{t_1, t_2\} = \emptyset$. The proof is similar to the previous case. The difference is that here the reduction can compute the tampered key $\tilde{\kappa}$ locally, as the indices t_1, t_2 do not belong to $\mathcal{H}_j$. Moreover, the ciphertexts $\tilde{c}_1$ and $\tilde{c}_2$ are reconstructed via leakage queries, although using subsets $\tilde{\mathcal{A}}_1$ and $\tilde{\mathcal{A}}_2$ which are different from the sets $\mathcal{A}_1$ and $\mathcal{A}_2$ used in the hybrid. However, by the special reconstruction property of Σ, this makes no difference, and thus also the distribution of $\tilde{c}$ is perfectly emulated.

(c) Case $\mathcal{H}_j = \{t_1, t_2\}$. In this case, the tampered ciphertext $\tilde{c}$ is computed exactly as in the hybrid experiment. Moreover, the tampering query $(f^{t_1}_{\mathsf{move}}, f^{t_2}_{\mathsf{move}})$ yields exactly the decoding of $(\tilde{s}_{\hat{\mathcal{T}},t_1}, \tilde{s}_{\hat{\mathcal{T}},t_2})$, and thus B's simulation is perfect.

(d) Case $\mathcal{H}_j \cap \mathcal{T} = \emptyset$. This is the simplest case. In fact, here, A's tampering query does not depend on $s_{\mathcal{H}_j,h_1}, s_{\mathcal{H}_j,h_2}$, and thus the answer to such a query can be computed by the reduction locally (yielding a perfect simulation).

In order to conclude the proof, we now show that the reduction B is $(\ell^* + 2\gamma + O(\log\lambda))$-admissible. Let $\mathbf{\Lambda}^{\mathsf{B}} = (\mathbf{\Lambda}^{\mathsf{B}}_1, \ldots, \mathbf{\Lambda}^{\mathsf{B}}_n)$ and $\mathbf{\Lambda}^{\mathsf{A}} = (\mathbf{\Lambda}^{\mathsf{A}}_1, \ldots, \mathbf{\Lambda}^{\mathsf{A}}_n)$ be the concatenation of all answers to leakage oracle queries received by B and A, respectively (where $\mathbf{\Lambda}^{\mathsf{B}}_i, \mathbf{\Lambda}^{\mathsf{A}}_i$ represent the leakage from the i-th share). Recall that B makes leakage queries for simulating both A's leakage and tampering queries; moreover, for each tampering query, the reduction considers 4 different cases. Wlog., we will assume that A's tampering queries either always fall in case (a) (i.e., $\mathcal{H}_j \subset \mathcal{T}$ and $|\mathcal{H}_j \cap \{t_1, t_2\}| = 1$), or in case (b) (i.e., $\mathcal{T} \cap \mathcal{H}_j \neq \emptyset$ and $\mathcal{H}_j \cap \{t_1, t_2\} = \emptyset$). The reason why

this can be done is that in the other two cases no extra leakage is required in order to simulate A's tampering query.

Our goal is to show that the following equations hold:

$$\begin{aligned}\widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R},\mathbf{\Lambda}^{\mathsf{B}}) &\geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R}) - (\ell^* + O(\log\lambda) + 2\gamma) \\ \widetilde{\mathbb{H}}_\infty(\mathbf{R}|\mathbf{L},\mathbf{\Lambda}^{\mathsf{B}}) &\geq \widetilde{\mathbb{H}}_\infty(\mathbf{R}|\mathbf{L}) - (\ell^* + O(\log\lambda) + 2\gamma),\end{aligned} \tag{6}$$

where $(\mathbf{L},\mathbf{R})$ is the random variable corresponding to the target codeword in the experiment with Γ. Since the proof of the two equations is analogous, in what follows we only focus on showing Eq. (6). Denote by $q^* \in [p]$, where $p(\lambda) \in \mathtt{poly}(\lambda)$, the index corresponding to the tampering query, if any, in which the reduction triggers a self-destruct. Note that the latter happens only when either: (i) $\tilde{c}_1^{(q^*)} \neq \tilde{c}_2^{(q^*)}$; or (ii) the tampering oracle returns $\perp$; or (iii) $\mathsf{ADec}(\tilde{\kappa}^{(q^*)},\tilde{c}^{(q^*)}) = \perp$. Since the query where a self-destruct happens depends on the target encoding, the index of the special self-destruct query is actually a random variable, which we denote by $\mathbf{q}^*$. We also write $\mathbf{\Lambda}^{(\mathbf{q}^*)}$ for the random variable corresponding to the leakage required to simulate the last tampering query before self-destruct. Finally, let $\hat{\mathbf{S}} = ((\mathbf{S}_i^*)_{i\in[n]\setminus\mathcal{H}_j}, \hat{\mathbf{S}}_{h_1}^*, \hat{\mathbf{S}}_{h_2}^*)$, where $\hat{\mathbf{S}}_h^* = (\mathbf{S}_{\mathcal{H},h})_{\mathcal{H}\in\binom{[n]}{2}\setminus\mathcal{H}_j}$, be the random variable corresponding to the shares sampled by the reduction at the beginning of the simulation (recall that these values are sampled independently from $\mathbf{L}$), and $\tilde{\mathbf{K}} = (\tilde{\mathbf{K}}^{(1)},\ldots,\tilde{\mathbf{K}}^{(\mathbf{q}^*-1)})$ be the random variable corresponding to the collection of mauled keys (one for each tampering query before self-destruct). Since conditioning can only decrease the average min-entropy, we can write:

$$\widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R},\mathbf{\Lambda}^{\mathsf{B}}) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R},\mathbf{\Lambda}^{\mathsf{B}},\hat{\mathbf{S}},\tilde{\mathbf{K}}). \tag{7}$$

We now specify more explicitly what the leakage $\mathbf{\Lambda}^{\mathsf{B}}$ is. Let $\mathcal{Q}_1$ (resp. $\mathcal{Q}_2$) be the set of indices $q < q^*$ such that $q \in \mathcal{Q}_1$ (resp. $q \in \mathcal{Q}_2$) if and only if the q-th tampering query is of type (a) (resp. of type (b)). With more granularity, we can further partition the set $\mathcal{Q}_2$ into two sets $\mathcal{Q}_{2,1}$ and $\mathcal{Q}_{2,2}$, corresponding to the indices $q < q^*$ such that $q \in \mathcal{Q}_{2,1}$ and $q \in \mathcal{Q}_{2,2}$ if and only if the q-th tampering query is of type (b)ii. (i.e., $|\mathcal{T}\cap\mathcal{H}_j| = 1$) or of type (b)iii. (i.e., $|\mathcal{T}\cap\mathcal{H}_j| = 2$), respectively. Hence, the random variable $\mathbf{\Lambda}^{\mathsf{B}}$ can be parsed as:

$$\mathbf{\Lambda}^{\mathsf{B}} \equiv \left((\tilde{\mathbf{C}}_2^{(q)})_{q\in\mathcal{Q}_1}, (\tilde{\mathbf{C}}_1^{(q)},\tilde{\mathbf{C}}_2^{(q)})_{q\in\mathcal{Q}_{2,1}}, (\tilde{\mathbf{C}}_2^{(q)})_{q\in\mathcal{Q}_{2,2}}, \mathbf{\Lambda}^{(\mathbf{q}^*)}, \mathbf{\Lambda}^{\mathsf{A}}\right),$$

where $\tilde{\mathbf{C}}_1^{(q)},\tilde{\mathbf{C}}_2^{(q)}$ are the random variables corresponding to the mauled ciphertexts obtained from the leakage oracle during the q-th tampering query. By plugging the expression for $\mathbf{\Lambda}^{\mathsf{B}}$ into Eq. (7), we obtain:

$$\begin{aligned}\widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R},\mathbf{\Lambda}^{\mathsf{B}}) &\geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R},(\tilde{\mathbf{C}}_1^{(q)},\tilde{\mathbf{C}}_2^{(q)})_{q\in\mathcal{Q}_1\cup\mathcal{Q}_{2,1}\cup\mathcal{Q}_{2,2}},\mathbf{\Lambda}^{(\mathbf{q}^*)},\mathbf{\Lambda}^{\mathsf{A}},\hat{\mathbf{S}},\tilde{\mathbf{K}}) && (8)\\ &\geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R},(\tilde{\mathbf{C}}_1^{(q)})_{q\in\mathcal{Q}_1},(\tilde{\mathbf{C}}_1^{(q)})_{q\in\mathcal{Q}_{2,1}},(\tilde{\mathbf{C}}_2^{(q)})_{q\in\mathcal{Q}_{2,2}},\mathbf{\Lambda}^{(\mathbf{q}^*)},\mathbf{\Lambda}^{\mathsf{A}},\hat{\mathbf{S}},\tilde{\mathbf{K}}) && (9)\\ &\geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R},\mathbf{\Lambda}^{(\mathbf{q}^*)},\mathbf{\Lambda}^{\mathsf{A}},\hat{\mathbf{S}},\tilde{\mathbf{K}}) && (10)\\ &\geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R},\mathbf{\Lambda}^{(\mathbf{q}^*)},\mathbf{\Lambda}^{\mathsf{A}}) && (11)\\ &\geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R},\mathbf{\Lambda}^{\mathsf{A}}) - O(\log\lambda) - 2\gamma. && (12)\end{aligned}$$

In the above derivation, Eq. (8) follows again by the fact that conditioning can only decrease the average min-entropy. Eq. (9) and Eq. (10), instead, contain the crux of our argument. Here, we exploit the fact that for all tampering queries $q < q^*$ the random variables $\tilde{\mathbf{C}}_1^{(q)}$ and $\tilde{\mathbf{C}}_2^{(q)}$ are the same, and further $(\tilde{\mathbf{C}}_1^{(q)})_{q\in\mathcal{Q}_1}$ (resp. $(\tilde{\mathbf{C}}_1^{(q)})_{q\in\mathcal{Q}_{2,1}}$ and $(\tilde{\mathbf{C}}_2^{(q)})_{q\in\mathcal{Q}_{2,2}}$) are a deterministic function of $\hat{\mathbf{S}},\tilde{\mathbf{K}}$. Eq. (11) holds because $\hat{\mathbf{S}}$ is sampled independently of $\mathbf{L}$, whereas by Lemma 2

$\tilde{\mathbf{K}}$ is uniquely determined given $\mathbf{R}$. Eq. (12) holds because of the chain rule for conditional average min-entropy, as in order to describe $\mathbf{\Lambda}^{(\mathsf{q}^*)}$ we need $O(\log \lambda)$ bits to represent the index $q^* \leq p(\lambda) \in \mathtt{poly}(\lambda)$, plus 2γ bits to represent the two ciphertexts $\tilde{c}_1^{(q^*)}$ and $\tilde{c}_2^{(q^*)}$ (where now potentially $\tilde{c}_1^{(q^*)} \neq \tilde{c}_2^{(q^*)}$).

Finally, we prove that $\widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R}, \mathbf{\Lambda}^{\mathsf{A}}) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R}) - \ell^*$, which combined with Eq. (12) implies that B is ℓ-admissible, thus concluding the proof of the theorem. Here, we use the fact that adversary A is ℓ^*-admissible. Recall that the latter implies that:

$$\widetilde{\mathbb{H}}_\infty(\mathbf{S}^*_{h_1}|(\mathbf{S}^*_h)_{h\neq h_1}, \mathbf{\Lambda}^{\mathsf{A}}) = \widetilde{\mathbb{H}}_\infty(\mathbf{L}, \hat{\mathbf{S}}^*_{h_1}|\mathbf{R}, (\mathbf{S}^*_h)_{h\in[n]\setminus\mathcal{H}_j}, \hat{\mathbf{S}}^*_{h_2}, \mathbf{\Lambda}^{\mathsf{A}}) \tag{13}$$

$$\geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}, \hat{\mathbf{S}}^*_{h_1}|\mathbf{R}, (\mathbf{S}^*_h)_{h\in[n]\setminus\mathcal{H}_j}, \hat{\mathbf{S}}^*_{h_2}) - \ell^*, \tag{14}$$

where Eq. (13) follows by definition of $\mathbf{S}^*_{h_1}$ and $\mathbf{S}^*_{h_2}$, and Eq. (14) follows by definition of admissibility. Intuitively, since A leaks at most ℓ^* bits of information from $(\mathbf{L}, \hat{\mathbf{S}}^*_{h_1})$, then A must leak at most ℓ bits of information about $\mathbf{L}$. More formally, Eq. (14) implies:

$$\widetilde{\mathbb{H}}_\infty(\mathbf{L}, \hat{\mathbf{S}}^*_{h_1}|\mathbf{R}, \mathbf{\Lambda}^{\mathsf{A}}) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}, \hat{\mathbf{S}}^*_{h_1}|\mathbf{R}) - \ell^*, \tag{15}$$

which holds true since the values $(\mathbf{S}^*_h)_{h\in[n]\setminus\mathcal{H}_j}, \hat{\mathbf{S}}^*_{h_2}$ are sampled independently of $\mathbf{L}$ and $\mathbf{R}$. Hence, let $\hat{\mathcal{S}}^*$ be the set of all possible assignments for $\hat{\mathbf{S}}^*_{h_1}$:

$$\widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R}, \mathbf{\Lambda}^{\mathsf{A}}) = \widetilde{\mathbb{H}}_\infty(\mathbf{L}|\hat{\mathbf{S}}^*_{h_1}, \mathbf{R}, \mathbf{\Lambda}^{\mathsf{A}}) \tag{16}$$

$$\geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}, \hat{\mathbf{S}}^*_{h_1}|\mathbf{R}, \mathbf{\Lambda}^{\mathsf{A}}) - \log|\mathcal{S}^*| \tag{17}$$

$$\geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}, \hat{\mathbf{S}}^*_{h_1}|\mathbf{R}) - \ell^* - \log|\hat{\mathcal{S}}^*| \tag{18}$$

$$\geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}|\mathbf{R}) + \log|\hat{\mathcal{S}}^*| - \ell^* - \log|\hat{\mathcal{S}}^*|, \tag{19}$$

where Eq. (16) and Eq. (19) hold by independence of $\mathbf{L}$ and $\hat{\mathbf{S}}^*_{h_1}$ together with the fact that $\hat{\mathbf{S}}^*_{h_1}$ is uniformly distributed over $\hat{\mathcal{S}}^*$ (by perfect secrecy of Σ and codewords uniformity of Γ), Eq. (17) follows by the chain rule for conditional average min-entropy (cf. Lemma 1), and Eq. (18) holds by Eq. (15). This finishes the proof. □

Lemma 6. $\forall j \in [N]$, $\forall b \in \{0,1\}$: $\{\mathbf{H}_{j,1}(\lambda, b)\}_{\lambda\in\mathbb{N}} \approx \{\mathbf{H}_{j,2}(\lambda, b)\}_{\lambda\in\mathbb{N}}$.

Proof. The only difference between the two hybrids is that, in step 5e, $\mathbf{H}_{j,2}(\lambda, b)$ additionally checks whether $\tilde{\kappa} = \hat{\kappa}_{\mathcal{H}_j}$ but $\tilde{c} \neq c_{\mathcal{H}_j}$, and if that is the case it then returns $\perp$ to A (with consequent self-destruct). Else, if $\tilde{\kappa} = \hat{\kappa}_{\mathcal{H}_j}$ and $\tilde{c} = c_{\mathcal{H}_j}$, the hybrids sets $\tilde{m} = ♥$.

The proof of indistinguishability is down to the authenticity property of the AE scheme Π. In particular, by Shoup's lemma [Sho04], we can bound the computational distance between the two hybrids by the probability of the event that $\tilde{\kappa} = \hat{\kappa}_{\mathcal{H}_j}$ and $\tilde{c} \neq c_{\mathcal{H}_j}$, but $\mathsf{ADec}(\kappa_{\mathcal{H}_j}, \tilde{c}) \neq \perp$.

Notice that the above event is identical to the winning condition of an adversary B in game $\mathbf{G}^{\mathsf{auth}}_{\Pi,\mathsf{B}}(\lambda)$. The reduction is obvious, and therefore omitted. □

Lemma 7. $\forall j \in [N]$, $\forall b \in \{0,1\}$: $\{\mathbf{H}_{j,2}(\lambda, b)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{H}_{j,3}(\lambda, b)\}_{\lambda\in\mathbb{N}}$.

Proof. The only difference between the two hybrids is that in $\mathbf{H}_{j,3}(\lambda, b)$ the ciphertext $c_{\mathcal{H}_j}$ is computed as $\mathsf{AEnc}(\kappa_{\mathcal{H}_j}, \hat{m})$ instead of $\mathsf{AEnc}(\kappa_{\mathcal{H}_j}, m_b)$ as in $\mathbf{H}_{j,2}(\lambda, b)$.

The proof of indistinguishability is down to semantic security of the AE scheme Π. In particular, given the change introduced in $\mathbf{H}_{j,1}(\lambda, b)$, we notice that the key $\kappa_{\mathcal{H}_j}$ is not part of the view of the adversary A. Furthermore, due to the change introduced in $\mathbf{H}_{j,2}(\lambda, b)$, we never run decryption with the original key $\kappa_{\mathcal{H}_j}$. Given these observations the reduction is obvious, and therefore omitted. □

Let Com be a non-interactive commitment scheme with message space $\mathcal{M}$, randomness space $\mathcal{D}$, and commitment space $\mathcal{C}$. Let $\Gamma' = (\mathsf{Enc}', \mathsf{Dec}')$ be a split-state code with message space $\mathcal{M}' = \mathcal{M} \times \mathcal{D}$. Define the following split-state code, with message space $\mathcal{M}$.

Encoding: Upon input a value $m \in \mathcal{M}$, sample random coins $\delta \leftarrow_{\$} \mathcal{D}$ and compute $com = \mathsf{Com}(m; \delta)$ and $(L', R') \leftarrow_{\$} \mathsf{Enc}'(m||\delta)$. Return the codeword $(L, R) := ((com, L'), (com, R'))$.

Decoding: Upon input a pair (L, R), parse $(L, R) := ((com_0, L'), (com_1, R'))$. Hence, proceed as follows:

(a) If $com_0 \neq com_1$, return $\perp$; else, let $com = com_0 = com_1$.

(b) Run $m' = \mathsf{Dec}'(L', R')$; if $m' = \perp$ return $\perp$.

(c) Parse $m' := m||\delta$; if $com = \mathsf{Com}(m; \delta)$ return m, else return $\perp$.

Figure 7: The non-malleable code from [OPVV18].

Note that $\{\mathbf{H}_{j,3}(\lambda, b)\}_{\lambda \in \mathbb{N}}$ is identically distributed to $\{\mathbf{H}_j(\lambda, b)\}_{\lambda \in \mathbb{N}}$. Moreover, the view of the adversary in $\mathbf{H}_N(\lambda, b)$ is independent of the bit b, i.e. $\{\mathbf{H}_N(\lambda, 0)\}_{\lambda \in \mathbb{N}} \equiv \{\mathbf{H}_N(\lambda, 1)\}_{\lambda \in \mathbb{N}}$. Thus, the theorem statement follows by combining the above lemmas. □

5.3 Instantiating the Construction

We now show how to instantiate Thm. 4, under the assumption of 1-to-1 OWFs. It is well-known that authenticated encryption can be constructed in a black-box way from any OWF, whereas we can use the classical Shamir's construction [Sha79] for the underlying TSS scheme. The latter is easily seen to meet the special reconstruction property.

Thus, in what follows, we focus on the interesting part which is the instantiation of the split-state continuously non-malleable code. We start by recalling the recent construction of [OPVV18], which is based on the following building blocks.

- **Perfectly binding commitments:** Let Com be a non-interactive commitment taking as input a message $m \in \mathcal{M}$ and returning a commitment $com = \mathsf{Com}(m; \delta) \in \mathcal{C}$, where $\delta \in \mathcal{D}$; the pair (m, δ) is called the opening. Intuitively, Com is perfectly binding if any commitment com can be opened in a single way; additionally, we say that Com is computationally hiding if for all messages $m_0, m_1 \in \mathcal{M}$ it is hard to distinguish whether com is a commitment to m_0 or to m_1. Such commitments, for single-bit messages, can be based on any 1-to-1 OWF ϕ, by letting $com = (\phi(\delta_1), \langle \delta_1, \delta_2 \rangle \oplus m)$, where $\delta_1, \delta_2 \leftarrow_{\$} \{0,1\}^\lambda$ and $\langle \cdot, \cdot \rangle$ denotes the inner product.[8]
- **Split-state one-time statistically NMC:** Let $\Gamma' = (\mathsf{Enc}', \mathsf{Dec}')$ be a split-state code over message space $\mathcal{M}' = \mathcal{M} \times \mathcal{D}$. We need that Γ' satisfies one-time statistical non-malleability under noisy leakage, i.e. it must satisfy Definition 4 w.r.t. all ℓ-admissible *computationally unbounded* adversaries limited to ask a *single* tampering query.

To encode a message $m \in \mathcal{M}$, Ostrovsky *et al.* [OPVV18] first commit to m, obtaining $com = \mathsf{Com}(m; \delta)$, and then compute a non-malleable encoding (L', R') of the value $m||\delta$. Their construction is formally described in Fig. 7. Below, we extend their analysis by showing that the final construction inherits the noisy-leakage resilience of the underlying code Γ'; we also make the observation that the original security analysis can be simplified when the code Γ' is noisy-leakage resilient.

[8] To extend the domain to messages of arbitrary polynomial length, one just commits to each bit of the message individually.

Theorem 5. *Assume that* Com *is a non-interactive perfectly binding and computationally hiding commitment, and that* Γ' *is a split-state* ℓ'*-noisy leakage-resilient and one-time statistically non-malleable code. Then, the code* Γ *described in Fig. 7 is an* ℓ*-LR-CNMC, as long as* $\ell' = \ell + \gamma + O(\log \lambda)$ *where* $\gamma = \log |\mathcal{C}|$ *is the size of a commitment under* Com*. Moreover, if* Γ' *has symmetric decoding and codewords uniformity, so does* Γ*.*

Proof sketch. The fact that the decoding is symmetric follows by inspection of the description of the decoding process in Fig. 7. Codewords uniformity follows because Γ has codewords uniformity, and furthermore in the instantiation of perfectly binding commitments from 1-to-1 OWFs the value *com* is distributed uniformly over the set of all possible commitments to the message. The proof of non-malleability follows closely that of [OPVV18, Theorem 3], hence below we only emphasize the differences between the two proofs and assume the reader is familiar with the security proof in [OPVV18].

Fix $m_0, m_1 \in \mathcal{M}$, and let $\mathbf{T}(\lambda, b) \equiv \mathbf{CNMC}_{\Gamma,\mathsf{A}}(\lambda, m_0, m_1, b)$. As in the original proof, we consider a mental experiment $\mathbf{H}(\lambda, b)$, where we replace (L', R') with an encoding of a random and independent value $m' \leftarrow\!\!\$\, \mathcal{M}'$; when answering tampering queries, the experiment returns ♥ in case the mauled message $\tilde{m}$ happens to be equal to m'. The two lemmas below conclude the proof.

Lemma 8. $\forall b \in \{0,1\} : \ \{\mathbf{T}(\lambda, b)\}_{\lambda\in\mathbb{N}} \approx_s \{\mathbf{H}(\lambda, b)\}_{\lambda\in\mathbb{N}}$.

Proof. The proof is by induction over the number of tampering queries $p \in \mathtt{poly}(\lambda)$ asked by adversary A. The induction basis is identical as in the proof of [OPVV18, Lemma 2]; the only difference is that we additionally need to consider the fact that A can ask noisy-leakage queries. Below, we explain how to do this directly for the more complicated case of the inductive step.

As for the inductive step, assume that the statement of the lemma holds w.r.t. all adversaries that ask up to $q \in [p-1]$ tampering queries; we show that this implies the statement for all adversaries asking up to $q+1$ tampering queries, via a reduction to the one-time statistical non-malleability of the underlying code Γ'. The main difference between our reduction and the one in [OPVV18] is that, instead of performing the simulation of tampering queries via optimistic answers, we directly leak the modified commitments $\tilde{com}^{(j)}_{\mathsf{left}}, \tilde{com}^{(j)}_{\mathsf{right}}$ corresponding to the first q tampering queries $(f^{(j)}_{\mathsf{left}}, f^{(j)}_{\mathsf{right}})$ asked by the adversary; this is done up to the first query, if any, such that $\tilde{com}^{(j)}_{\mathsf{left}} \neq \tilde{com}^{(j)}_{\mathsf{right}}$, upon which a self-destruct is triggered. Thus, we show that such strategy still yields an ℓ'-admissible reduction.

Fix $b = 0$ (the proof for the other case being identical). Let (L', R') be the target encoding in the tampering experiment relative to $(\mathsf{Enc}', \mathsf{Dec}')$. Here, (L', R') is either an encoding of $m'_0 = m_0||\delta$ or an encoding of a random value $m' = m'_1$. Adversary A', on input (m_0, m'_0, m'_1), proceeds as follows:

1. Parse $m'_0 = m_0||\delta$ and compute $com := \mathsf{Com}(m_0; \delta)$. Run $\mathsf{A}(1^\lambda)$.
2. For each leakage query $(\mathtt{left}, g_{\mathsf{left}})$ (resp. $(\mathtt{right}, g_{\mathsf{right}})$) asked by A, define the leakage function h_{left} (resp. h_{right}) that hard-wires (a description of) g_{left}, com (resp. g_{right}, com), and returns the same as $g_{\mathsf{left}}(com, L')$ (resp. $g_{\mathsf{right}}(com, R')$). Forward $(\mathtt{left}, h_{\mathsf{left}})$ (resp. $(\mathtt{right}, h_{\mathsf{right}})$) to the target leakage oracle.
3. For each $j \in [q]$, upon input the j-th tampering query $(f^{(j)}_{\mathsf{left}}, f^{(j)}_{\mathsf{right}})$, where $f^{(j)}_{\mathsf{left}}, f^{(j)}_{\mathsf{right}}$ are polynomial-time computable, proceed as follows.
   (a) Define the leakage function $\hat{h}^{(j)}_{\mathsf{left}}$ (resp. $\hat{h}^{(j)}_{\mathsf{right}}$) that hard-wires (a description of) $f^{(j)}_{\mathsf{left}}$ (resp. $f^{(j)}_{\mathsf{right}}$), and returns the commitment $\tilde{com}^{(j)}_{\mathsf{left}}$ (resp. $\tilde{com}^{(j)}_{\mathsf{right}}$) such that $f^{(j)}_{\mathsf{left}}(com, L') = (\tilde{com}^{(j)}_{\mathsf{left}}, \tilde{L}')$ (resp. $f^{(j)}_{\mathsf{right}}(com, R') = (\tilde{com}^{(j)}_{\mathsf{right}}, \tilde{R}')$).

(b) Forward $(\texttt{left}, \hat{h}^{(j)}_{\mathsf{left}})$ and $(\texttt{right}, \hat{h}^{(j)}_{\mathsf{right}})$ to the target leakage oracle, obtaining values $\tilde{com}^{(j)}_{\mathsf{left}}$ and $\tilde{com}^{(j)}_{\mathsf{right}}$.

(c) If $\tilde{com}^{(j)}_{\mathsf{left}} \neq \tilde{com}^{(j)}_{\mathsf{right}}$, set $\tilde{m}^{(j)} = \perp$ and self-destruct. Else, let $\tilde{com}^{(j)} = \tilde{com}^{(j)}_{\mathsf{left}} = \tilde{com}^{(j)}_{\mathsf{right}}$ and find by brute force the corresponding opening $\tilde{m}^{(j)}$ (i.e., $\tilde{com}^{(j)} = \mathsf{Com}(\tilde{m}^{(j)}; \tilde{\delta}^{(j)})$ for some $\tilde{\delta}^{(j)} \in \mathcal{D}$).

(d) If $\tilde{m}^{(j)} \in \{m_0, m_1, m'\}$, return ♥ to A, and otherwise return $\tilde{m}^{(j)}$ to A.

4. Upon input the last tampering query $(f^{(q+1)}_{\mathsf{left}}, f^{(q+1)}_{\mathsf{right}})$, where $f^{(q+1)}_{\mathsf{left}}, f^{(q+1)}_{\mathsf{right}}$ are polynomial-time computable, proceed as follows.

(a) Define the same functions $\hat{h}^{(q+1)}_{\mathsf{left}}$ and $\hat{g}^{(q+1)}_{\mathsf{right}}$ considered in step 3a, and forward them to the target leakage oracle, obtaining values $\tilde{com}^{(q+1)}_{\mathsf{left}}$ and $\tilde{com}^{(q+1)}_{\mathsf{right}}$.

(b) Define the polynomial-time computable tampering function f'_{left} (resp. f'_{right}) that hard-wires (a description of) com and $f^{(q+1)}_{\mathsf{left}}$ (resp. $f^{(q+1)}_{\mathsf{right}}$), and, upon input L' (resp. R'), returns the value $\tilde{L}'$ (resp. $\tilde{R}'$) specified by $f^{(q+1)}_{\mathsf{left}}(com, L') = (\tilde{com}^{(q+1)}_{\mathsf{left}}, \tilde{L}')$ (resp. $f^{(q+1)}_{\mathsf{right}}(com, R') = (\tilde{com}^{(q+1)}_{\mathsf{right}}, \tilde{R}')$).

(c) Forward $(f'_{\mathsf{left}}, f'_{\mathsf{right}})$ to the target tampering oracle, obtaining a value $\tilde{m}^{(q+1)} \in \mathcal{M} \cup \{\perp, ♥\}$.

(d) If $\tilde{com}^{(q+1)}_{\mathsf{left}} \neq \tilde{com}^{(q+1)}_{\mathsf{right}}$, or the value $\tilde{m}^{(q+1)}$ is not a valid opening, return $\perp$ to A; else, return $\tilde{m}^{(q+1)}$.

5. Check that the simulation up to the first q queries did not cause any inconsistency, due to the fact that the outcome of one of the tampering queries should have been $\perp$ because $(\tilde{L}', \tilde{R}')$ was not a valid encoding. As shown in [OPVV18], the latter can be done at the price of one additional bit of leakage, and we can do exactly the same here, with the adaptation that such a leakage query now also hard-wires the answers to leakage queries as computed in step 2. More in details:

(a) The reduction keeps a list of all codewords that are possible encodings of the messages m'_0, m'_1; let $\hat{\mathcal{L}}^{(1)}$ and $\hat{\mathcal{R}}^{(1)}$ be the initial sets. During the simulation these sets are updated to $\hat{\mathcal{L}}^{(j+1)} \subseteq \hat{\mathcal{L}}^{(j)}$ and $\hat{\mathcal{R}}^{(j+1)} \subseteq \hat{\mathcal{R}}^{(j)}$, by removing all codewords that are not compatible with the answer to the j-th tampering query.

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

(c) Define the leakage query h_{check} that hard-wires a description of A, the values (com, m_0, m_1), a description of the final tampering query $(f^{(q+1)}_{\mathsf{left}}, f^{(q+1)}_{\mathsf{right}})$, the answer to previous tampering queries $(\tilde{m}^{(1)}, \ldots, \tilde{m}^{(q)})$, the answer to all leakage queries Λ, and the set $\hat{\mathcal{R}}^{(q+1)}$. The output of the function is a bit $\tilde{b}$ such that $\tilde{b} = 1$ if and only if $\mathsf{A}(\tilde{m}^{(1)}, \ldots, \tilde{m}^{(q)}, \tilde{m}^*, \Lambda) = 0$ more often when $\mathsf{Dec}((com, L'), (com, \hat{R}')) = m_0$, where $\hat{R}'_1 \in \hat{\mathcal{R}}^{(q+1)}$ and $\tilde{m}^* = \mathsf{Dec}(f^{(q+1)}_{\mathsf{left}}(com, L'), f^{(q+1)}_{\mathsf{right}}(com, \hat{R}'))$. (Unless the decoding yields any of m_0, m_1, m', in which case $\tilde{m}^* = ♥$.)

(d) Forward $(\texttt{left}, h_{\mathsf{check}})$ to the target leakage oracle, obtaining a bit $\tilde{b}$.

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

Notice that attacker A' runs in exponential time. Moreover, an analysis identical to that of [OPVV18, Lemma 3] shows that the distinguishing advantage of A' is negligibly close to that of A. In order to conclude the proof, it remains to show that A' is ℓ'-admissible, for ℓ' as in the statement of the theorem. We do this below.

Note that adversary A' makes leakage queries at steps 2, 3b, 4a, and 5c, but the leakage queries of step 4a and 5c is executed only once, and for a total of at most $\gamma+1$ bits of leakage on each share. Let $\mathbf{q}^* \in \mathbb{N}$ be the random variable corresponding to the the index of the tampering query, if any, where the retrieved commitments happen to be different. Clearly, the leakage queries of step 3b are executed exactly $\mathbf{q}^*$ times. Denote by $\mathbf{\Lambda}_{\mathsf{left}}$ (resp. $\mathbf{\Lambda}_{\mathsf{right}}$) the random variable corresponding to the leakage performed by the reduction on the left (resp. right) side of the target encoding $(\mathbf{L}', \mathbf{R}')$. We can write:

$$\widetilde{\mathbb{H}}_\infty(\mathbf{L}'|\mathbf{R}', \mathbf{\Lambda}_{\mathsf{left}}) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}'|\mathbf{R}', \hat{h}^{(1)}_{\mathsf{left}}(\mathbf{L}), \ldots, \hat{h}^{(\mathbf{q}^*)}_{\mathsf{left}}(\mathbf{L}')) - \ell - \gamma - 1 \tag{20}$$

$$= \widetilde{\mathbb{H}}_\infty(\mathbf{L}'|\hat{h}^{(1)}_{\mathsf{right}}(\mathbf{R}'), \ldots, \hat{h}^{(\mathbf{q}^*)}_{\mathsf{right}}(\mathbf{R}')) - \ell - \gamma - 1 \tag{21}$$

$$\geq \widetilde{\mathbb{H}}_\infty(\mathbf{L}'|\mathbf{R}') - \ell - \gamma - 1 - O(\log \lambda). \tag{22}$$

where Eq. (20) follows by definition of $\mathbf{\Lambda}_{\mathsf{left}}$, since the adversary A is ℓ-admissible and furthermore the leakage performed in steps 3b and 5c consists of at most $\gamma + 1$ bits, Eq. (21) follows by the fact that, for each $q \leq q^*$, the commitments leaked in step 4a on the left and on the right are identical, and Eq. (22) follows by interpreting the tuple $(\hat{h}^{(1)}_{\mathsf{right}}(\mathbf{R}'), \ldots, \hat{h}^{(\mathbf{q}^*)}_{\mathsf{right}}(\mathbf{R}'))$ as a function of $(\mathbf{R}', \mathbf{q}^*)$ and by further noting that $q^* \leq p(\lambda) \in \mathtt{poly}(\lambda)$. The lemma follows. □

Lemma 9. $\{\mathbf{H}(\lambda, 0)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{H}(\lambda, 1)\}_{\lambda\in\mathbb{N}}$.

Proof. This proof is identical to [OPVV18, Lemma 5]. The only difference is that, once again, we additionally need to take care of simulating the answer of A's leakage query. However, this is easy because the reduction to the hiding property of the commitment scheme knows a perfectly distributed target codeword for A. □

□

It remains to instantiate Theorem 5 by exhibiting a split-state code that is noisy-leakage resilient and one-time statistically non-malleable. We do this by using the construction of Aggarwal *et al.* [ADKO15b], which we briefly recall below. Let $\mathbb{F}$ be a finite field. The encoder first encodes the underlying message $m' \in \mathcal{M}'$ using an auxiliary split-state one-time statistically non-malleable (but not leakage resilient) code $\Gamma'' = (\mathsf{Enc}'', \mathsf{Dec}'')$, obtaining $(L'', R'') \in [N]\times[N]$, where $[N]$ is a sparse subset of $\mathbb{F}$ with size $N \ll |\mathbb{F}|$. Hence, each share L'', R'' is processed using a slight variant of the inner-product extractor, i.e. L'' (resp. R'') is encoded via two additional shares $(L''_1, R''_1) \in \mathbb{F}^{2t}$ (resp. $(L''_2, R''_2) \in \mathbb{F}^{2t}$) such that $\psi(\langle L''_1, R''_1\rangle) = L''$ (resp. $\psi(\langle L''_2, R''_2\rangle) = R''$), where $\psi : \mathbb{F} \to [N]$ is an arbitrary bijection. The final encoding is then defined to be $(L', R') = ((L''_1, L''_2), (R''_1, R''_2)) \in \mathbb{F}^{2t} \times \mathbb{F}^{2t}$. It is easy to see that Γ' has codewords uniformity; moreover, as long as Γ'' has symmetric decoding, Γ' has symmetric decoding too.

By plugging in the above construction the split-state non-malleable code of [ADL18], which has $\log N \in O(k^7)$, and choosing statistical error $\epsilon := 2^{-k^2}$, we obtain a split-state noisy-leakage-resilient one-time statistically non-malleable code with leakage parameter $\ell' \approx k^{14}/12$ (cf. [ADKO15b, Corollary 4.2]). Moreover, since the code of [ADL18] has symmetric decoding, so does the code we obtain. It is important to note that the definition of leakage-resilient non-malleability considered in [ADKO15b] is simulation based, and not indistinguishability based as our Definition 4. However, as argued already in [OPVV18], the former implies the latter.

Finally, we observe that the security of the code in [ADKO15b] was analyzed in the bounded leakage model. However, it is not too hard to see that their analysis can be extended to the case of noisy leakage. This is because all their proof requires is that the conditional average min-entropy of the left side $\mathbf{L}'$ and the right side $\mathbf{R}'$ of the codeword conditioned on the leakage is high enough (with overwhelming probability), and the latter holds true not only when the

length of the leakage is ℓ'-bounded, but more generally as long as the adversary is ℓ'-admissible. Given the above condition, the rest of the proof follows by considering the same partitioning strategy as in [ADKO15b, Lemma 6.1].

6 Boosting the Rate

6.1 Information Rate of Secret Sharing

An important measure of the efficiency of a secret sharing scheme is its information rate, defined as the asymptotic ratio between the size of the message and the maximum size of a share, as the message length goes to infinity.[9]

Definition 10 (Rate of secret sharing)**.** Let $\Sigma = (\mathsf{Share}, \mathsf{Rec})$ be an n-party secret sharing over message space $\mathcal{M}$ and share space $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$. We define the information rate of Σ to be $\rho(\mu, n)$ if and only if for all $\lambda \in \mathbb{N}$ there exists $\mu_0 \in \mathbb{N}$ such that for for any $\mu \geq \mu_0$:

$$\min_{i \in [n]} \frac{\mu}{\sigma_i(\mu, n, \lambda)} \geq \rho(\mu, n),$$

where $\mu = \log |\mathcal{M}|$ and $\sigma_i(\mu, n, \lambda) = \log |\mathcal{S}_i|$ denote, respectively, the bit-length of the message and of the i-th share under Σ. Moreover, we say that Σ has rate 0 (resp. rate 1) if $\lim_{\mu \to \infty} \rho(\mu, n)$ is 0 (resp. 1).

Note that the length of a share under the threshold secret sharing scheme of §5 is $O(1/n^2)$. Hence, we have obtained:

Corollary 2. *Under the assumption of 1-to-1 OWFs, there exists a noisy-leakage-resilient continuously non-malleable threshold secret sharing with information rate $O(1/n^2)$.*

6.2 A Rate-Optimizing Compiler

In this section, we show how to optimize the rate of any LR-CNMSS, under computational assumptions. We will achieve this through a so-called rate compiler, i.e. a black-box transformation that takes any rate-0 LR-CNMSS and returns a rate-1 LR-CNMSS.

Our compiler is formally described in Fig. 8, and is inspired by a beautiful idea of Aggarwal *et al.* [AAG+16], who considered a similar question for the case of (one-time) non-malleable codes against split-state tampering; recently, their approach was also analyzed in the case of continuous tampering [CFV19]. Intuitively, the construction works as follows. The sharing function samples a uniformly random key κ for a symmetric encryption scheme, and secret shares κ using the underlying rate-0 threshold secret sharing, obtaining shares $\kappa_1, \ldots, \kappa_n$. Next, the input message m is encrypted under the key κ, yielding a ciphertext c, and the final share of each player is defined to be $s_i = (\kappa_i, c)$. Importantly, the reconstruction function, before obtaining the key κ and decrypting the ciphertext c, checks that the ciphertext contained in every given share is the same.

Note that when the initial secret sharing scheme is a 2-out-of-2 TSS, i.e. Σ' is actually a split-state LR-CNMC, we obtain as a special case one of the rate compilers analyzed in [CFV19]. A notable advantage of our result, however, is that we can instantiate the construction in the plain model (whereas Coretti *et al.* assume a CRS).

[9] One can also define a more general notion of information rate for secret sharing schemes [BSGV96], which depends on the entropy of the distribution $\mathbf{M}$ of the input message. The above definition is obtained as a special case, by considering the uniform distribution.

Rate-Optimizing Compiler for LR-CNMSS

Let $\Sigma' = (\mathsf{Share}', \mathsf{Rec}')$ be an n-party TSS over message space $\mathcal{M}' := \mathcal{K}$ and share space $\mathcal{S}'^n$. Let $\Pi = (\mathsf{AEnc}, \mathsf{ADec})$ be an authenticated secret-key encryption scheme with key space $\mathcal{K}$, message space $\mathcal{M}$, and ciphertext space $\mathcal{C}$. Consider the following construction of a derived n-party TSS over message space $\mathcal{M}$ and share space $\mathcal{S} := (\mathcal{S}' \times \mathcal{C})^n$.

Sharing function $\mathsf{Share}(m)$**:** Sample $\kappa \leftarrow\!\!\$\, \mathcal{K}$, and compute $c \leftarrow\!\!\$\, \mathsf{AEnc}(\kappa, m)$. Let $(\kappa_1, \ldots, \kappa_n) \leftarrow\!\!\$\, \mathsf{Share}'(\kappa)$. Output $s = (s_1, \ldots, s_n)$, where $s_i := (\kappa_i, c)$ for all $i \in [n]$.

Reconstruction function $\mathsf{Rec}(s_{\mathcal{I}})$**:** Parse $s_{\mathcal{I}} = (s_{i_1}, \ldots, s_{i_\varrho})$, where $s_{i_j} = (\kappa_{i_j}, c_{i_j})$ for all $j \in [\varrho]$. Let $\kappa = \mathsf{Rec}'(\kappa_{i_1}, \ldots, \kappa_{i_\varrho})$; if $\kappa = \perp$, return $\perp$. Else, if $c_{i_1} = \ldots = c_{i_\varrho} := c$, output $\mathsf{ADec}(\kappa, c)$, and otherwise output $\perp$.

Figure 8: Boosting the rate of any leakage-resilient continuously non-malleable secret sharing (in the computational setting).

6.3 Security Analysis

We establish the following result:

Theorem 6. *Let $n, \tau, \varrho \in \mathbb{N}$, with $\tau \leq \varrho \leq n$. Assuming that Σ' is an $(n, \tau, \varrho, \ell')$-LR-CNMSS, and that Π is a secure AE scheme, the secret sharing scheme Σ of Fig. 8 is an (n, τ, ϱ, ℓ)-LR-CNMSS as long as $\ell' = \ell + \lambda + O(\log \lambda)$.*

Before proving the above theorem, we compute the rate of the scheme of Fig. 8. Note that the size of the ciphertext is $\mu + \mathtt{poly}(\lambda)$, whereas the size of the key is independent of the message, so that the rate of the obtained scheme is $\frac{\mu}{\mu + \mathtt{poly}(\lambda)}$, thus yielding a rate-1 scheme.

Corollary 3. *Under the assumption of 1-to-1 OWFs, there exists a noisy-leakage-resilient continuously non-malleable threshold secret sharing with information rate (asymptotically) 1.*

Proof of Theorem 6. We proceed to show each of the required properties.

Threshold reconstruction. Since Σ' satisfies ϱ-threshold reconstruction, we get that with overwhelming probability the reconstruction function Rec yields the same key κ sampled during sharing. Hence, the correctness of the AE scheme ensures that $\mathsf{ADec}(\kappa, c)$ yields the original message, with overwhelming probability. Thus, Σ satisfies ϱ-threshold reconstruction as well.

Threshold privacy. We need to show that for all $m_0, m_1 \in \mathcal{M}$, and for each unauthorized subset $\mathcal{U} \subset [n]$ with $|\mathcal{U}| \leq \tau - 1$, the two ensembles $\{\mathsf{Share}(1^\lambda, m_0)_{\mathcal{U}}\}_{\lambda \in \mathbb{N}}$ and $\{\mathsf{Share}(1^\lambda, m_1)_{\mathcal{U}}\}_{\lambda \in \mathbb{N}}$ are computationally close. Fix $m_0, m_1 \in \mathcal{M}$, and $\mathcal{U}$. We transition to a mental experiment where the shares $(\kappa_1, \ldots, \kappa_n)$ are computed using an independent random key $\kappa' \leftarrow\!\!\$\, \mathcal{K}$. A straightforward reduction to the τ-threshold privacy of Σ shows that, for all $b \in \{0, 1\}$, the distribution $\{\mathsf{Share}(1^\lambda, m_b)_{\mathcal{U}}\}_{\lambda \in \mathbb{N}}$ in the original experiment is computationally close to that of the modified experiment.

Now, a straightforward reduction to the semantic security of the SKE scheme shows that, in the above defined mental experiment, and for all $m_0, m_1 \in \mathcal{M}$, we have $\{\mathsf{Share}(1^\lambda, m_0)_{\mathcal{U}}\}_{\lambda \in \mathbb{N}} \approx_c \{\mathsf{Share}(1^\lambda, m_1)_{\mathcal{U}}\}_{\lambda \in \mathbb{N}}$. This concludes the proof of τ-threshold privacy.

Continuous non-malleability. Fix $m_0, m_1 \in \mathcal{M}$, and consider the following hybrids.

Hybrid $\mathbf{H}_0(\lambda, b)$: Distributed identically to $\mathbf{CNMSS}_{\Sigma,\mathsf{A}}(\lambda, m_0, m_1, b)$.

Hybrid $\mathbf{H}_1(\lambda, b)$: Identical to the previous experiment, except that the sharing function now samples an independent key $\kappa' \leftarrow\!\!\$\ \mathcal{K}$ and runs $(\kappa_1, \ldots, \kappa_n) \leftarrow\!\!\$\ \mathsf{Share}'(\kappa')$. The reconstruction function proceeds identically to Rec, except that if the key κ' is reconstructed correctly, the original key κ is used to decrypt the (unique) ciphertext c.

Hybrid $\mathbf{H}_2(\lambda, b)$: Identical to the previous experiment, except that the reconstruction function always outputs $\perp$ in case the reconstructed key is equal to the key κ' sampled by the sharing function, but the (unique) ciphertext contained in the given shares is different from the ciphertext c computed during sharing.

Lemma 10. *$\forall b \in \{0,1\}$: $\{\mathbf{H}_0(\lambda, b)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{H}_1(\lambda, b)\}_{\lambda\in\mathbb{N}}$.*

Proof. The proof is down to the continuous non-malleability of the underlying secret sharing scheme Σ'. For simplicity, fix $b = 0$ (the proof for $b = 1$ is identical). By contradiction, assume that there exists a PPT adversary A, and a polynomial $p_{0,1}(\lambda) \in \mathtt{poly}(\lambda)$, such that for infinitely many values $\lambda \in \mathbb{N}$ we have:

$$|\mathbb{P}\left[\mathbf{H}_0(\lambda, 0) = 1\right] - \mathbb{P}\left[\mathbf{H}_1(\lambda, 0) = 1\right]| \geq \frac{1}{p_{0,1}(\lambda)}.$$

Note that the probability in the above equation is taken over the randomness of the sharing function Share, over the coins for A, and over the choice of the keys $\kappa, \kappa' \in \mathcal{K}$. By a standard averaging argument, this means that there exist at least two values κ, κ' such that the above equation holds even when we fix these particular values of κ, κ'. We construct a PPT adversary A' that violates continuous non-malleability of Σ' w.r.t. the pair of messages (κ, κ'). In what follows, denote by $s' := (\kappa_1, \ldots, \kappa_n)$ the target secret sharing in the experiment with A' (i.e., s' is either a secret sharing of κ or of κ' under Σ'); wlog. we also assume that the bit-size of a ciphertext is made of $\eta \in \mathbb{N}$ blocks of size $\lambda \in \mathbb{N}$.

Adversary A (playing the game of Fig. 3):

1. Run $\mathsf{A}(1^\lambda)$, and sample $c \leftarrow\!\!\$\ \mathsf{AEnc}(\kappa, m_0)$.
2. Upon input a leakage query (i, g) from A answer as follows:
   (a) Define the leakage function $g'(\cdot) := g(\cdot, c)$.
   (b) Submit (i, g') to the target $\mathcal{O}_{\mathsf{leak}}(s', \cdot, \cdot)$ oracle, and feed the answer from the oracle back to A.
3. Upon input a tampering query $(\mathcal{T}, (f_1, \ldots, f_n))$ from A answer as follows:
   (a) Parse $\mathcal{T} = \{t_1, \ldots, t_{\tilde{\varrho}}\}$, where $n \geq \tilde{\varrho} \geq \varrho$.
   (b) For $i \in [n]$ and $k \in [\eta]$, define the leakage function $h'_{i,k}(\kappa_i)$ that hard-wires a description of c, f_i, runs $(\tilde{\kappa}_i, \tilde{c}_i) = f_i(\kappa_i, c)$, and then outputs the k-th block of $\tilde{c}_i$.
   (c) Consider the following sequence of leakage queries:

$$(t_1, h'_{t_1,1}), \ldots, (t_{\tilde{\varrho}}, h'_{t_{\tilde{\varrho}},1}), \ldots, (t_1, h'_{t_1,\eta}), \ldots, (t_{\tilde{\varrho}}, h'_{t_{\tilde{\varrho}},\eta}).$$

   Start asking the above queries to the target $\mathcal{O}_{\mathsf{leak}}(s', \cdot, \cdot)$ oracle from left to right, but the first time it happens that the answer to

$$(t_j, h'_{t_j,k}) \qquad \text{and} \qquad (t_{j'}, h'_{t_{j'},k})$$

differ, for values $k \in [\eta]$ and $j, j' \in [\tilde{\varrho}]$ such that $j \neq j'$, output $\perp$ and self-destruct. Else, go to the next step with a uniquely determined ciphertext $\tilde{c}$ which is obtained wlog. by concatenating the η blocks corresponding to the leakage queries $(t_1, h'_{t_1,1}), \ldots, (t_1, h'_{t_1,\eta})$.

(d) Forward to the target $\mathcal{O}_{\mathsf{nmss}}(s', \cdot, \cdot)$ oracle the tampering query $(\mathcal{T}, (f'_1, \ldots, f'_n))$, where for each $i \in [n]$ we set $f'_i(\cdot) = f'_i(\cdot, c)$, obtaining an answer $\tilde{\kappa}$. Hence, if $\tilde{\kappa} = \perp$, output $\perp$ and self-destruct. Else, if $\tilde{\kappa} = \heartsuit$, let $\tilde{m} = \mathsf{ADec}(\kappa, \tilde{c})$, whereas if $\tilde{\kappa} \neq \heartsuit$, let $\tilde{m} = \mathsf{ADec}(\tilde{\kappa}, \tilde{c})$. Finally, if $\tilde{m} = \perp$, output $\perp$ and self-destruct; else, return $\tilde{m}$.

4. Output whatever A outputs.

For the analysis, first note that A' clearly runs in polynomial time. Furthermore, we claim that the view of A is perfectly simulated, meaning that if the target s' is a secret sharing of κ, then the reduction yields a distribution which is identical to that of experiment $\mathbf{H}_0(\lambda, 0)$, whereas if the target s' is a secret sharing of κ', then the reduction yields a distribution which is identical to that of experiment $\mathbf{H}_1(\lambda, 0)$. We support this claim below:

- **(Target secret sharing.)** Note that the distribution of $c \leftarrow\!\!\$\ \mathsf{AEnc}(\kappa, m_0)$ is identical in both experiments $\mathbf{H}_0(\lambda, 0)$ and $\mathbf{H}_1(\lambda, 0)$. Since every leakage and tampering query hard-wires the ciphertext c, this implies that the reduction perfectly emulates the distribution of the target secret sharing $s = (\kappa_i, c)_{i \in [n]}$ inside the leakage and tampering oracles.
- **(Answer to leakage queries.)** By the above argument, the simulation of leakage queries is perfect.
- **(Answer to tampering queries.)** Let $(\mathcal{T}, (f_1, \ldots, f_n))$ be a generic tampering query, where $\mathcal{T} = \{t_1, \ldots, t_{\tilde{\varrho}}\}$, and denote by $\tilde{s}_i = (\tilde{\kappa}_i, \tilde{c}_i)$ the i-th tampered share. By the above argument, the distribution of each $\tilde{s}_i$ is perfectly simulated inside the tampering oracle. Hence, the reduction checks via leakage queries if $\tilde{c}_{t_1} = \cdots = \tilde{c}_{t_{\tilde{\varrho}}}$; if this is not the case it answer's A tampering query with $\perp$ and self-destructs. Otherwise, it obtains a perfectly distributed reconstructed key $\tilde{\kappa} = \mathsf{Rec}'(\tilde{\kappa}_{t_1}, \ldots, \tilde{\kappa}_{t_{\tilde{\varrho}}})$ which is used to complete the simulation of $\mathsf{Rec}(\tilde{s}_{t_1}, \ldots, \tilde{s}_{t_{\tilde{\varrho}}})$ for A as done in both experiments $\mathbf{H}_0(\lambda, 0)$ and $\mathbf{H}_1(\lambda, 0)$.

It remains to show that if A is ℓ admissible, then A' is ℓ'-admissible for ℓ' as in the statement of the theorem. To do that, let $\mathbf{S}'_1, \ldots, \mathbf{S}'_n$ be the random variable corresponding to the target secret sharing in the experiment with A', and denote by $\mathbf{\Lambda}^1_i$ the random variable corresponding to the overall leakage performed by the reduction on each share $\mathbf{S}'_i$ in step 2b. Similarly, let $(\mathcal{T}_q, (f_1^{(q)}, \ldots, f_n^{(q)}))$ be the q-th tampering query, where $\mathcal{T} = \{t_j\}_{j \in [\tilde{\varrho}]}$. We denote by $\mathbf{\Lambda}^2_{q,t_j,k}$ the random variable corresponding to the leakage obtained from each function $h'_{t_j,k}$ used by the reduction to answer the q-th tampering query in step 3c. Additionally, denote by $q^* \in [p]$ and $k^* \in [\eta]$ the special indices (if any exist) for which $\mathbf{\Lambda}^2_{q^*,t_j,k^*} \neq \mathbf{\Lambda}^2_{q^*,t_{j'},k^*}$ for some $j \neq j'$; note that the actual value of q^*, k^* depends on the random coins used to sample the target secret sharing, and on the randomness of the adversary, thus q^*, k^* are actually random variables which we denote by $\mathbf{q}^*, \mathbf{k}^*$. Finally, we also write $\mathbf{\Lambda}^2_i$ for the random variable corresponding to the cumulative leakage performed by the reduction on the share $\mathbf{S}'_i$, in order to answer all the tampering queries asked by A.

Now, for each $i \in [n]$, we can write:

$$\widetilde{\mathbb{H}}_\infty(\mathbf{S}'_i | (\mathbf{S})_{j \neq i}, \mathbf{\Lambda}^1_i, \mathbf{\Lambda}^2_i) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{S}'_i | (\mathbf{S}'_j)_{j \neq i}, \mathbf{\Lambda}^2_i) - \ell \tag{23}$$

$$\geq \widetilde{\mathbb{H}}_\infty((\mathbf{S}'_j)_{j \neq i}, \mathbf{q}^*, \mathbf{k}^*) - \ell - \lambda \tag{24}$$

$$\geq \widetilde{\mathbb{H}}_\infty((\mathbf{S}'_j)_{j \neq i}) - \ell - \lambda - O(\log \lambda), \tag{25}$$

where Eq. (23) follows by the fact that A is ℓ-admissible, and Eq. (25) follows because the bit-length of both q^* and k^* is polynomial in the security parameter. To see that also Eq. (24) holds true, let $\mathcal{T}_q$ be the target set corresponding to the q-th tampering query, for any $q < q^*$. There are two cases: either $i \notin \mathcal{T}_q$, or $i \in \mathcal{T}_q$. In both cases, all the leakages $\mathbf{\Lambda}^2_{q,t_j,k}$ can be computed as a deterministic function of the other shares $(\mathbf{S}_j)_{j \neq i}$. This is trivially the case when $i \notin \mathcal{T}_q$, whereas when $i \in \mathcal{T}_q$ it follows by the fact that this is not the self-destruct round, which in particular means that the leakages from each share are identical. The same argument applies to the special query q^*, with target set $\mathcal{T}_{q^*} = \mathcal{T}^*$, for all leakage queries h'_{k,t_j} with index $k < k^*$, while for the special index k^*, in case $i \in \mathcal{T}^*$, it could happen that $\mathbf{\Lambda}^2_{q^*,i,k^*}$ is different from all other leakages, which accounts for at most λ bits of loss in the average min-entropy.

The above arguments imply that A' perfectly emulates the view of A, and additionally A' is ℓ'-admissible. Thus, A' has a non-negligible distinguishing advantage, which concludes the proof of the lemma. □

Lemma 11. $\forall b \in \{0,1\}$*:* $\{\mathbf{H}_1(\lambda, b)\}_{\lambda \in \mathbb{N}} \approx_c \{\mathbf{H}_2(\lambda, b)\}_{\lambda \in \mathbb{N}}$.

Proof. The proof is down to the authenticity of the underlying AE scheme Π. For simplicity, fix $b = 0$ (the proof for $b = 1$ is identical). Denote by $\mathbf{Forge}_q$ the event that the q-th tampering query asked by A is such that the tampered reconstructed key $\tilde{\kappa}$ is equal to κ', whereas all mauled ciphertexts are equal to a single ciphertext $\tilde{c}$ that is different from the original ciphertext c of the target secret sharing, and moreover this ciphertext is a valid ciphertext w.r.t. key κ (i.e., $\mathsf{ADec}(\kappa, \tilde{c}) \neq \bot$). Let $\mathbf{Forge} = \mathbf{Forge}_1 \vee \cdots \vee \mathbf{Forge}_q$. It is easy to see that conditioned on $\overline{\mathbf{Forge}}$ the distribution of $\mathbf{H}_1(\lambda, 0)$ and $\mathbf{H}_2(\lambda, 0)$ are identical. Thus, by Shoup's lemma [Sho04],

$$|\mathbb{P}[\mathbf{H}_1(\lambda, 0) = 1] - \mathbb{P}[\mathbf{H}_2(\lambda, 0) = 1]| \leq \mathbb{P}[\mathbf{Forge}].$$

We will show that $\mathbf{Forge}$ only happens with negligible probability, which implies the statement.

By contradiction, assume that there exists a PPT adversary A, and a polynomial $p_{1,2}(\lambda) \in \mathtt{poly}(\lambda)$, such that for infinitely many values $\lambda \in \mathbb{N}$ we have that A provokes event $\mathbf{Forge}$ in a run of $\mathbf{H}_2(\lambda, 0)$. Denote by $p \in \mathtt{poly}(\lambda)$ the number of tampering queries performed by A. We build a reduction B (using A) which attempts at winning the game $\mathbf{G}^{\mathsf{auth}}_{\Pi,\mathsf{B}}(\lambda)$ of Def. 2.

Adversary B (playing the left game of Fig. 1):

1. Run $\mathsf{A}(1^\lambda)$. Sample $\kappa' \leftarrow\!\!\$\ \mathcal{K}$, and $(\kappa_1, \ldots, \kappa_n) \leftarrow\!\!\$\ \mathsf{Share}'(\kappa')$. Pick a random index $q^* \leftarrow\!\!\$\ [p]$.
2. Forward m_0 to the challenger, obtaining a ciphertext c (i.e., an encryption of m_0 under some unknown key $\kappa \in \mathcal{K}$). Set $s_i = (\kappa_i, c)$ for all $i \in [n]$.
3. Upon input a leakage query (i, g) from A, answer with $g(s_i)$.
4. Upon input a tampering query $(\mathcal{T}, (f_1, \ldots, f_n))$ from A, let $\mathcal{T} = (t_1, \ldots, t_{\tilde{\varrho}})$. For each $j \in [\tilde{\varrho}]$, compute the tampered share $\tilde{s}_{t_j} = f_{t_j}(s_{t_j}) = (\tilde{\kappa}_{t_j}, \tilde{c}_{t_j})$. Hence:
   (a) Let $\tilde{\kappa} = \mathsf{Rec}(\tilde{\kappa}_{t_1}, \ldots, \tilde{\kappa}_{t_{\tilde{\varrho}}})$; if $\tilde{\kappa} = \bot$, output $\bot$ and self-destruct.
   (b) Check that $\tilde{c}_{t_1} = \cdots = \tilde{c}_{t_{\tilde{\varrho}}}$; if that is not the case, output $\bot$ and self-destruct. Else, let $\tilde{c} = \tilde{c}_{t_1}$.
   (c) In case $\tilde{\kappa} \neq \kappa'$ return $\mathsf{ADec}(\tilde{\kappa}, \tilde{c})$ to A. Else, if $\tilde{\kappa} = \kappa'$, further check whether $\tilde{c} = c$: If it does, return ♥ to A, and otherwise return $\bot$, unless this is the q^*-th tampering query, in which case forward $\tilde{c}$ to the challenger and terminate the execution with A.

For the analysis, first note that B runs in polynomial time. Moreover, the secret sharing $s = (s_i)_{i \in [n]}$ emulated by the reduction is distributed exactly as in both experiments $\mathbf{H}_1(\lambda, 0)$

and $\mathbf{H}_2(\lambda, 0)$. As a consequence, B's simulation of A's leakage and tampering queries is perfect. Thus, A will provoke event **Forge** with probability $1/p_{1,2}(\lambda)$. Assuming that B guesses correctly the index q^* corresponding to the first tampering query in which event **Forge** is provoked, we obtain that the reduction, in step 4c, forwards to the challenger of game $\mathbf{G}^{\mathsf{auth}}_{\Pi,\mathsf{B}}(\lambda)$ a ciphertext $\tilde{c} \neq c$ such that $\mathsf{ADec}(\kappa, \tilde{c}) \neq \perp$. We conclude that B breaks the authenticity game with non-negligible probability $1/p(\lambda) \cdot 1/p_{1,2}(\lambda)$, a contradiction. □

Lemma 12. $\{\mathbf{H}_2(\lambda, 0)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{H}_2(\lambda, 1)\}_{\lambda\in\mathbb{N}}$.

Proof. The proof is down to the semantic security of the underlying AE scheme Π. By contradiction, assume that there exists a PPT adversary A, and a polynomial $p_2(\lambda) \in \mathtt{poly}(\lambda)$, such that for infinitely many values $\lambda \in \mathbb{N}$ we have:

$$|\mathbb{P}[\mathbf{H}_2(\lambda, 0) = 1] - \mathbb{P}[\mathbf{H}_2(\lambda, 1) = 1]| \geq \frac{1}{p_2(\lambda)}.$$

We build a reduction B (using A) which attempts at winning the game $\mathbf{G}^{\mathsf{sem}}_{\Pi,\mathsf{B}}(\lambda)$ of Def. 2.

Adversary B (playing the right game of Fig. 1):

1. Run $\mathsf{A}(1^\lambda)$. Sample $\kappa' \leftarrow\!\!\$\ \mathcal{K}$, and $(\kappa_1, \ldots, \kappa_n) \leftarrow\!\!\$\ \mathsf{Share}'(\kappa')$.
2. Forward (m_0, m_1) to the challenger, obtaining a ciphertext c (i.e., either an encryption of m_0 or an encryption of m_1, under some unknown key $\kappa \in \mathcal{K}$). Set $s_i = (\kappa_i, c)$ for all $i \in [n]$.
3. Upon input a leakage query (i, g) from A, answer with $g(s_i)$.
4. Upon input a tampering query $(\mathcal{T}, (f_1, \ldots, f_n))$ from A, let $\mathcal{T} = (t_1, \ldots, t_{\tilde{\varrho}})$. For each $j \in [\tilde{\varrho}]$, compute the tampered share $\tilde{s}_{t_j} = f_{t_j}(s_{t_j}) = (\tilde{\kappa}_{t_j}, \tilde{c}_{t_j})$. Hence:
   (a) Let $\tilde{\kappa} = \mathsf{Rec}(\tilde{\kappa}_{t_1}, \ldots, \tilde{\kappa}_{t_{\tilde{\varrho}}})$; if $\tilde{\kappa} = \perp$, output $\perp$ and self-destruct.
   (b) Check that $\tilde{c}_{t_1} = \cdots = \tilde{c}_{t_{\tilde{\varrho}}}$; if that is not the case, output $\perp$ and self-destruct. Else, let $\tilde{c} = \tilde{c}_{t_1} = \cdots = \tilde{c}_{t_{\tilde{\varrho}}}$.
   (c) In case $\tilde{\kappa} \neq \kappa'$ return $\mathsf{ADec}(\tilde{\kappa}, \tilde{c})$ to A. Else, if $\tilde{\kappa} = \kappa'$, further check whether $\tilde{c} = c$: If it does, return ♥ to A, and otherwise output $\perp$ and self-destruct.

For the analysis, first note that B runs in polynomial time. Moreover, depending on the target ciphertext c being either an encryption of m_0 or an encryption of m_1, the secret sharing $s = (s_i)_{i\in[n]}$ emulated by the reduction is distributed exactly as in either experiment $\mathbf{H}_2(\lambda, 0)$ or in experiment $\mathbf{H}_2(\lambda, 1)$. As a consequence, B's simulation of A's leakage and tampering queries is perfect; in particular, the latter statement follows by the fact that in experiment $\mathbf{H}_2(\lambda, b)$ we never need the original secret key κ in order to answer a tampering query from A. We conclude that B breaks the semantic security game with non-negligible probability, a contradiction. □

The proof of the theorem follows by combining the above lemmas. □

7 Threshold Signatures under Adaptive Memory Corruptions

7.1 Syntax

An n-party threshold signature is a tuple $\Pi = (\mathsf{KGen}, \Xi, \mathsf{Vrfy})$ specified as follows. (i) The PPT algorithm KGen takes as input the security parameter, and outputs a verification key $vk \in \mathcal{VK}$, and n secret keys $sk_1, \ldots, sk_n \in \mathcal{SK}$; (ii) $\Xi = (\mathsf{P}_1, \ldots, \mathsf{P}_n)$ specifies a set of protocols which can be run by a subset $\mathcal{I}$ of n interactive PPT Turing machines $\mathsf{P}_1, \ldots, \mathsf{P}_n$, where each P_i takes as input a message $m \in \mathcal{M}$ and secret key sk_i, and where we denote by $(\sigma, \xi) \xleftarrow{\Xi}\!\!\$\ \langle \mathsf{P}_i(sk_i, m)\rangle_{i\in\mathcal{I}}$ a

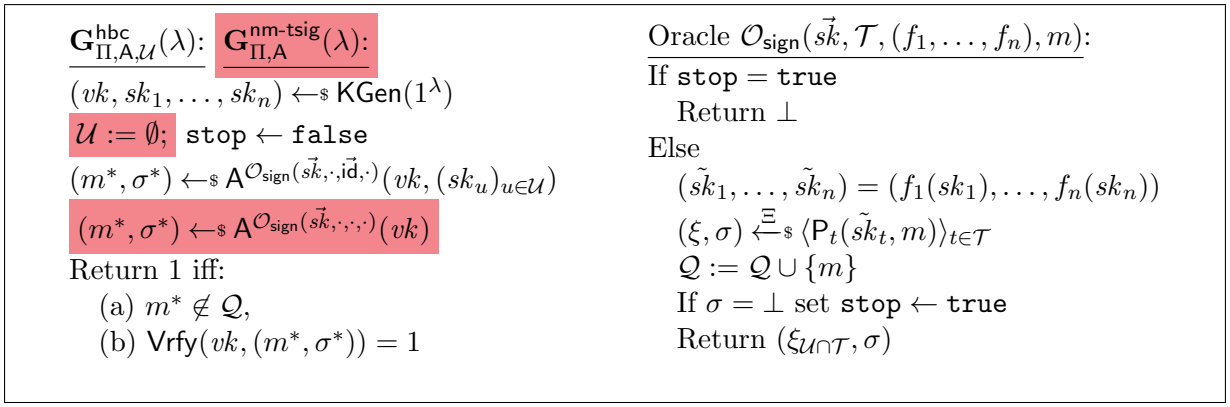

Figure 9: Experiments defining privacy and continuous non-malleability for threshold signatures. The vector $\vec{\mathsf{id}}$ contains the identity function (repeated n times).

run of Ξ by the parties $(\mathsf{P}_i)_{i\in\mathcal{I}}$, yielding a signature σ and transcript ξ. (iii) The deterministic polynomial-time algorithm Vrfy takes as input the verification key vk, and a pair (m, σ), and returns a decision bit.

For a parameter $\varrho \leq n$, we say that an n-party threshold signature is ϱ-correct if for all $\lambda \in \mathbb{N}$, all $(vk, sk_1, \ldots, sk_n)$ output by $\mathsf{KGen}(1^\lambda)$, all messages $m \in \mathcal{M}$, and all subsets $\mathcal{I}$ such that $|\mathcal{I}| \geq \varrho$, the following holds:

$$\mathbb{P}\left[\mathsf{Vrfy}(vk, (m, \sigma)) = 1 : \ (\sigma, \xi) \xleftarrow{\Xi}\!\!\$\ \langle \mathsf{P}_i(sk_i, m)\rangle_{i\in\mathcal{I}}\right] = 1.$$

We also consider *non-interactive* threshold signature schemes. Such schemes are fully specified by a tuple of polynomial-time algorithms $(\mathsf{KGen}, \mathsf{TSign}, \mathsf{Combine}, \mathsf{Vrfy})$, such that $\mathsf{KGen}, \mathsf{Vrfy}$ are as in the interactive case, whereas the protocol Ξ, run by a subset $\mathcal{I}$ of the parties, has the following simple structure:

- For each $i \in \mathcal{I}$, party P_i computes locally $\sigma_i \leftarrow\!\!\$\ \mathsf{TSign}(sk_i, m)$ and broadcasts the resulting signature share σ_i;
- For each $i \in \mathcal{I}$, party P_i locally computes $\sigma \leftarrow\!\!\$\ \mathsf{Combine}(vk, (\sigma_i)_{i\in\mathcal{I}})$; most notably, algorithm $\mathsf{Combine}$ only uses public information.

7.2 Security Model

We assume authenticated and private channels between each pair of parties, and an authenticated broadcast channel. The standard security notion for threshold signatures deals with an adversary A statically corrupting a subset $\mathcal{U}$ of the players, with size below the reconstruction threshold of the scheme. The guarantee is that the attacker should not be able to forge a valid signature on a fresh message, even after seeing a polynomial number of executions of the signature protocol on several messages and involving different subsets of the players; note that, for each such subset $\mathcal{I}$, the attacker learns the transcript of the signature protocol relative to the players in $\mathcal{U} \cap \mathcal{I}$. Below, we formalize this guarantee in the honest-but-curious case.

Definition 11 (Privacy for threshold signatures)**.** Let $\Pi = (\mathsf{KGen}, \Xi, \mathsf{Vrfy})$ be an n-party threshold signature scheme. We say that Π is τ-private against honest-but-curious adversaries if for all PPT attackers A, and all subsets $\mathcal{U} \subset [n]$ such that $|\mathcal{U}| < \tau$:

$$\mathbb{P}\left[\mathbf{G}^{\mathsf{hbc}}_{\Pi,\mathsf{A},\mathcal{U}}(\lambda) = 1\right] \in \mathtt{negl}(\lambda).$$

where the game $\mathbf{G}^{\mathsf{hbc}}_{\Pi,\mathsf{A},\mathcal{U}}(\lambda)$ is described in Fig. 9.

Non-malleability. Next, we consider an adversary able to corrupt the memory of each party independently. The security guarantee is still that of existential unforgeability, except that the attacker can now see a polynomial number of executions of the signature protocol under related secret-key shares, where both the modified shares and the subset of parties used for each signature computation, can be chosen adaptively. However, since in this case no player is actually corrupted and the protocol's messages are sent via private channels, for each run of the signature protocol the attacker only learns the signature (but not the transcript).

Definition 12 (Tamper-resilient threshold signatures)**.** Let $\Pi = (\mathsf{KGen}, \Xi, \mathsf{Vrfy})$ be an n-party threshold signature scheme. We say that Π is secure under continuous memory tampering if for all PPT adversaries A:

$$\mathbb{P}\left[\mathbf{G}^{\mathsf{nm\text{-}tsig}}_{\Pi,\mathsf{A}}(\lambda) = 1\right] \in \mathtt{negl}(\lambda),$$

where the game $\mathbf{G}^{\mathsf{nm\text{-}tsig}}_{\Pi,\mathsf{A}}(\lambda)$ is described in Fig. 9.

7.3 The Compiler

Given an n-party threshold signature $\Pi = (\mathsf{KGen}, \Xi, \mathsf{Vrfy})$, and an n-party TSS $\Sigma = (\mathsf{Share}, \mathsf{Rec})$, consider the following modified n-party threshold signature $\Pi^* = (\mathsf{KGen}^*, \Xi^*, \mathsf{Vrfy}^*)$.

- **Key generation $\mathsf{KGen}^*(1^\lambda)$:** Upon input the security parameter, run $(vk, sk_1, \ldots, sk_n) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$, compute $(sk_{i,1}, \ldots, sk_{i,n}) \leftarrow\!\!\$\ \mathsf{Share}(sk_i)$ for each $i \in [n]$, set $sk^*_i = (sk_{i',i})_{i' \in [n]}$, and output $(vk, sk^*_1, \ldots, sk^*_n)$.
- **Signature protocol $\Xi^* = (\mathsf{P}^*_1, \ldots, \mathsf{P}^*_n)$:** For any subset $\mathcal{I} \subset [n]$, and any message $m \in \mathcal{M}$, the protocol $\langle \mathsf{P}^*_i(sk^*_i, m)\rangle_{i\in\mathcal{I}}$ proceeds as follows:
  - Party P^*_i parses $sk^*_i = (sk_{i',i})_{i'\in[n]}$ and sends $sk_{i',i}$ to the i'-th party, for every $i' \in \mathcal{I} \setminus \{i\}$.
  - Party P^*_i waits to receive the messages $sk_{i,i''}$ for every $i'' \in \mathcal{I} \setminus \{i\}$, and afterwards it computes $sk_i = \mathsf{Rec}((sk_{i,i''})_{i''\in\mathcal{I}})$.
  - The players run $(\xi, \sigma) \overset{\Xi}{\leftarrow\!\!\$} \langle \mathsf{P}_i(sk_i, m)\rangle_{i\in\mathcal{I}}$.
- **Verification algorithm Vrfy^*:** Return the same as $\mathsf{Vrfy}(vk, (m, \sigma))$.

Intuitively, in the above protocol we first create a verification key vk and secret-key shares $(sk_1, \ldots, sk_n)$ under Π; hence, each value sk_i is further divided into n shares $(sk_{i,1}, \ldots, sk_{i,n})$ via the secret sharing Σ. The final secret-key share sk^*_i for the i-th party consists of the shares $(sk_{1,i}, \ldots, sk_{n,i})$, i.e. the collection of all the i-th shares under Σ. In order to sign a message, each player first sends to each other player the corresponding share. This way, party P_i can reconstruct sk_i, and the involved players can then run the original signature protocol Ξ.

Theorem 7. *For any $n, \varrho, \tau \in \mathbb{N}$ such that $n \geq \varrho \geq \tau$, assuming that Π is non-interactive, ϱ-correct, and τ-private against honest-but-curious adversaries, and that Σ is an $(n, \tau, \varrho, 0)$-LR-CNMSS, then the above defined threshold signature Π^* is ϱ-correct, τ-private against honest-but-curious adversaries, and secure under continuous memory tampering.*

Proof. The fact that Π^* satisfies ϱ-correctness follows immediately by the correctness property of Σ, and by ϱ-correctness of Π.

Privacy. In order to show that Π^* satisfies τ-privacy against honest-but-curious adversaries, fix any unauthorized subset $\mathcal{U} \subset [n]$, with $|\mathcal{U}| \leq \tau - 1$. Then, consider the following sequence of hybrids.

Hybrid $\mathbf{H}_j(\lambda)$: Let $\bar{\mathcal{U}} = [n] \setminus \mathcal{U}$. For the first j values $(sk_u)_{u\in\bar{\mathcal{U}}}$, compute $(sk_{u,1}, \ldots, sk_{u,n})$ as $\mathsf{Share}(sk'_u)$, for random and independent $sk'_u \leftarrow\!\!\$\ \mathcal{SK}$. Note that the answer to signature queries is still computed as in the original experiment, namely using the secret-key shares $(sk_i)_{i\in\mathcal{I}}$, where $\mathcal{I}$ is the reconstruction subset specified by the adversary.

Clearly, $\{\mathbf{H}_0(\lambda)\}_{\lambda\in\mathbb{N}} \equiv \{\mathbf{G}^{\mathsf{hbc}}_{\Pi^*,\mathsf{A},\mathcal{U}}(\lambda)\}_{\lambda\in\mathbb{N}}$. The two lemmas below conclude the proof of privacy.

Lemma 13. *$\forall j \in [n-\tau+1]$: $\{\mathbf{H}_j(\lambda)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{H}_{j-1}(\lambda)\}_{\lambda\in\mathbb{N}}$.*

Proof. By contradiction, assume that there is an index $j \in [0, n-\tau+1]$, and a PPT adversary A telling apart $\mathbf{H}_j(\lambda)$ and $\mathbf{H}_{j-1}(\lambda)$ with non-negligible probability. Let $\mathcal{U} = \{u_1, \ldots, u_{\tau-1}\}$ and $\bar{\mathcal{U}} = \{u_\tau, \ldots, u_n\}$. We construct a PPT attacker B that violates τ-threshold privacy of Σ w.r.t. subset $\mathcal{U}$.

Adversary B (attacking privacy of Σ):

1. Pick $(vk, sk_1, \ldots, sk_n) \leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$ and $sk'_{u_{\tau+j-1}} \leftarrow\!\!\$\ \mathcal{SK}$, and return $m_0^* = sk_{u_{\tau+j-1}}$, $m_1^* = sk'_{u_{\tau+j-1}}$ to the challenger, receiving back the shares $(sk_{u_{\tau+j-1},u})_{u\in\mathcal{U}}$.
2. For all other indices $i \in [n] \setminus \{\tau+j-1\}$, let sk_i^* be as defined in $\mathbf{H}_{j-1}(\lambda)$, then run A upon input vk and $(sk_u^*)_{u\in\mathcal{U}}$.
3. Upon input a signature query with subset $\mathcal{I}$ from A, answer as follows:
   (a) Run $(\sigma, \xi) \xleftarrow{\Xi}\!\!\$\ \langle \mathsf{P}_i(sk_i, m)\rangle_{i\in\mathcal{I}}$.
   (b) Let $\xi_i^* = \xi_i || sk_{i,i''}$ for all $i \in [n]$ and $i'' \in \mathcal{I} \setminus \{i\}$.
   (c) Return $(\sigma, (\xi_i^*)_{i\in\mathcal{U}\cap\mathcal{I}})$.
4. Whenever A outputs a forgery (m^*, σ^*), return the same as $\mathsf{Vrfy}(vk, (m^*, \sigma^*))$.
5. Output whatever A outputs.

Note that attacker B perfectly simulates the transcript of an execution of the signature protocol for Π^*; this is because the signature σ is computed honestly (using the real secret-key shares), whereas the partial transcript ξ_i^* is obtained by appending the corresponding part ξ_i in a run of the signature protocol for Π to the values the values $sk_{i,i''}$ received by the i-th player. It follows that in case the challenge $(sk_{u_{\tau+j-1},u})_{u\in\mathcal{U}}$ is a secret sharing of m_0^* the reduction perfectly emulates the view as in $\mathbf{H}_{j-1}(\lambda)$, whereas in case the challenge is a secret sharing of m_1^* the reduction perfectly emulates the view as in $\mathbf{H}_j(\lambda)$. The statement follows. □

Lemma 14. *$\forall$ PPT A: $\mathbb{P}[\mathbf{H}_{n-\tau+1}(\lambda) = 1] \in \mathtt{negl}(\lambda)$.*

Proof. The proof is down to τ-privacy of the underlying non-interactive threshold signature Π.[10] The main idea is that the reduction knows all the shares $(sk_u)_{u\in\mathcal{U}}$ directly, whereas it can use random and independent values $(sk'_u)_{u\in\bar{\mathcal{U}}}$ in order to obtain shares $(sk_i^*)_{i\in[n]}$ that are distributed exactly like in $\mathbf{H}_{n-\tau+1}(\lambda)$.

This knowledge allows to perfectly simulate signature queries as in the reduction from the previous lemma, i.e. we can use the target signature oracle to obtain the value σ and the partial transcripts $(\xi_i)_{i\in\mathcal{U}\cap\mathcal{I}}$, and append $(sk_{i,i''})_{i''\in\mathcal{I}\setminus\{i\}}$ to the latter in order to emulate $(\xi_i^*)_{i\in\mathcal{U}\cap\mathcal{I}}$. The formal reduction is obvious, and therefore omitted. □

[10] Strictly speaking, for this particular step of the proof we do not need to assume that Π is non-interactive.

Continuous memory tampering. Finally, we prove that Π^* is secure against continuous memory tampering. To this end consider the following hybrid experiments.

Hybrid $\mathbf{H}_j(\lambda)$: For the first j values $(sk_i)_{i\in[n]}$, compute $(sk_{i,1},\ldots,sk_{i,n})$ as $\mathsf{Share}(sk'_i)$, for random and independent $sk'_i \leftarrow\!\!\$\ \mathcal{SK}$. Note that for each signature query $(\mathcal{T},(f_1,\ldots,f_n),m)$, whenever the reconstructed secret-key share $\tilde{sk}_i$ equals sk'_i, for some $i\in[j]$, the signature σ on message m is computed using the original secret-key share sk_i.

Observe that $\{\mathbf{H}_0(\lambda)\}_{\lambda\in\mathbb{N}} \equiv \{\mathbf{G}^{\mathsf{nm\text{-}tsig}}_{\Pi^*,\mathsf{A}}(\lambda)\}_{\lambda\in\mathbb{N}}$. The two lemmas below conclude the proof.

Lemma 15. $\forall j\in[n]$: $\{\mathbf{H}_j(\lambda)\}_{\lambda\in\mathbb{N}} \approx_c \{\mathbf{H}_{j-1}(\lambda)\}_{\lambda\in\mathbb{N}}$.

Proof. By contradiction, assume that there exists an index $j\in[0,n]$, and a PPT adversary A telling apart $\mathbf{H}_j(\lambda)$ and $\mathbf{H}_{j-1}(\lambda)$ with non-negligible probability. We construct an attacker B that violates continuous non-malleability of Σ.

Adversary B (playing the game in Fig. 3):

1. Pick $(vk,sk_1,\ldots,sk_n)\leftarrow\!\!\$\ \mathsf{KGen}(1^\lambda)$ and $sk'_j\leftarrow\!\!\$\ \mathcal{SK}$, and return $m_0 = sk_j$, $m_1 = sk'_j$ to the challenger.
2. For all other indices $i\in[n]\setminus\{j\}$, let sk^*_i be as defined in $\mathbf{H}_{j-1}(\lambda)$, then run $\mathsf{A}(vk)$.
3. Upon input a signature query $(\mathcal{T},(f_1,\ldots,f_n),m)$ from A, answer as follows:
   (a) For each $i,i''\in[n]$, consider the tampering function $f'_{i,i''}$ that hard-wires $(sk_{i',i})_{i'\in[n]\setminus\{j\}}$, runs $(\tilde{sk}_{1,i},\ldots,\tilde{sk}_{n,i}) = f_i(sk_{1,i},\ldots,sk_{1,n})$, and outputs the tampered secret-key share $\tilde{sk}_{i'',i}$.
   (b) For each $t\in\mathcal{T}$, forward $(\mathcal{T},(f'_{1,t},\ldots,f'_{n,t}))$ to the target tampering oracle, obtaining back values $(\tilde{sk}_t)_{t\in\mathcal{T}}\in\mathcal{SK}\cup\{\heartsuit,\bot\}$.
   (c) If there exists $t\in\mathcal{T}$ such that $\tilde{sk}_t=\bot$, output $\bot$ and self-destruct. Else, for each $t\in\mathcal{T}$ let $\tilde{sk}'_t = sk_t$ in case $\tilde{sk}_t=\heartsuit$, and $\tilde{sk}'_t=\tilde{sk}_t$ otherwise.
   (d) Run $(\sigma,\xi)\overset{\Xi}{\leftarrow}\!\!\$\ \langle \mathsf{P}_t(\tilde{sk}'_t,m)\rangle_{t\in\mathcal{T}}$, and forward σ to A.
4. Whenever A outputs a forgery (m^*,σ^*), return the same as $\mathsf{Vrfy}(vk,(m^*,\sigma^*))$.
5. Output whatever A outputs.

Note that attacker B knows all the shares $sk_{i,i''}$, as defined in $\mathbf{H}_{j-1}(\lambda)$, except for $(sk_{j,1},\ldots,sk_{j,n})$ which are available through the tampering oracle. We claim that the simulation of A's signature queries is perfect. This is because for each query $(\mathcal{T},(f_1,\ldots,f_n),m)$ from A, the reduction obtains a perfectly distributed copy of $\tilde{sk}_t = \mathsf{Rec}((\tilde{sk}_{t,t''})_{t''\in\mathcal{T}})$ for each index $t\in\mathcal{T}$, and given such values it can perfectly simulate the computation of a signature as defined in either $\mathbf{H}_{j-1}(\lambda)$ or $\mathbf{H}_j(\lambda)$ (depending on the distribution of B's target oracle). It follows that B retains the same distinguishing advantage as that of A, concluding the proof. □

Lemma 16. $\forall$ *PPT* A: $\mathbb{P}\left[\mathbf{H}_n(\lambda)=1\right]\in\mathtt{negl}(\lambda)$.

Proof. By contradiction, assume that there exists a PPT adversary A winning game $\mathbf{H}_n(\lambda)$ with non-negligible probability. We build a PPT attacker B that breaks the privacy property of Π w.r.t. the unqualified subset $\mathcal{U}=\emptyset$. A description of B follows.

Adversary B (playing the un-boxed game in Fig. 9):

1. For all $i\in[n]$, sample $sk'_i\leftarrow\!\!\$\ \mathcal{SK}$, compute $(sk_{i,1},\ldots,sk_{i,n})\leftarrow\!\!\$\ \mathsf{Share}(sk_i)$, and set $sk^*_i = (sk_{i',i})_{i'\in[n]}$.

2. Run the adversary $\mathsf{A}(vk)$, and upon input each oracle query $(\mathcal{T}, (f_1, \ldots, f_n), m)$ from A, answer as follows:
   (a) For each $t \in \mathcal{T}$, let $\tilde{sk}_t^* = f_t(sk_t^*)$ and $\tilde{sk}_t = \mathsf{Rec}((sk_{t,t''})_{t'' \in \mathcal{T}})$.
   (b) If there exists an index $t \in \mathcal{T}$ such that $\tilde{sk}_t = \perp$, output $\perp$ and self-destruct (namely, answer $\perp$ to all future queries of A);
   (c) Else, query the target signature oracle upon input $(\mathcal{T}, m)$, receiving back a signature σ;
   (d) For each $t \in \mathcal{T}$, if $\tilde{sk}_t = sk_t$ set $\sigma'_t := \sigma_t$, else set $\sigma'_t \leftarrow\!\!\$\ \mathsf{TSign}(\tilde{sk}_t, m)$;
   (e) Output $\mathsf{Combine}(vk, (\sigma'_t)_{t \in \mathcal{T}})$.
3. Whenever A outputs (m^*, σ^*), return the same pair as forgery.

Note that the reduction perfectly emulates the distribution of the shares $(sk_i^*)_{i \in [n]}$ as in the hybrid experiment. Furthermore, it is not hard to show that the answer to signature queries has also the right distribution. This is because, for each signature query with reconstruction subset $\mathcal{T}$, the reduction computes perfectly distributed tampered secret-key shares $(\tilde{sk}_t)_{t \in \mathcal{T}}$ and:

- If some of the mauled shares is $\perp$, it outputs $\perp$ and self-destructs as in the hybrid game.
- Else, if all the mauled shares are valid, it defines σ'_t to be either a signature share of the message computed with the tampered key $\tilde{sk}_t$, or a signature share of the message computed with the original key sk_t, which can be obtained from the target signature oracle. Note that the latter is indeed possible because we assumed that Π is non-interactive.

It follows that B retains the same advantage as that of A, thus finishing the proof. □

□

8 Conclusions and Open Problems

We have initiated the study of *non-malleable, threshold* secret sharing withstanding a powerful adversary that can obtain both *noisy leakage* from each of the shares *independently*, and an *arbitrary polynomial* number of reconstructed messages corresponding to shares which can be *arbitrarily* related to the original ones (as long as the shares are modified *independently*). Importantly, in our model, both the tampering functions (mauling the original target secret sharing) and the reconstruction subsets (specifying which shares contribute to the reconstructed message) can be chosen *adaptively* by the attacker. Our main result establishes the existence of such schemes in the *computational setting*, under the minimal assumption of 1-to-1 OWFs, and with information rate asymptotically approaching 1 (as the message length goes to infinity).

Our work leaves several interesting open problems. We mention some of them below.

- **Mind the gap:** As we show, continuous non-malleability is impossible to achieve in the information-theoretic setting whenever the reconstruction threshold ϱ (i.e., the minimal number of shares required to reconstruct the message) is equal to the privacy threshold τ (i.e., any collection of $\tau - 1$ shares computationally hides the message). Our schemes, however, have a minimal gap $\varrho - \tau \geq 1$. It remains open to construct CNMSS for the optimal parameters $\varrho = \tau$, possibly with information-theoretic security (even without considering leakage and adaptive concurrent reconstruction).
- **Optimal rate:** It is well known that, in the computational setting, there exist robust threshold secret sharing schemes with optimal information rate n [Kra93] (i.e., the size of each share is μ/n where μ is the message size). It remains open whether continuously non-malleable threshold secret sharing schemes with such rate exist, and under which assumptions.

- **Arbitrary access structures:** Can we construct continuously non-malleable secret sharing beyond the threshold access structure, e.g. where the sets of authorized players can be represented by an arbitrary polynomial-size monotone span program, as in [GK18b]?
- **Joint tampering:** Can we construct continuously non-malleable secret sharing where the non-malleability property holds even if joint tampering with the shares is allowed, as in [GK18a, GK18b]?
- **Applications:** Finally, it would be interesting to explore other applications of continuously non-malleable secret sharing besides tamper resistance, e.g. in the spirit of non-malleable cryptography, as in [GJK15, CMTV15, GPR16, CDTV16, GK18a].

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.

[ADL18] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. *SIAM J. Comput.*, 47(2):524–546, 2018.

[ADN+17] Divesh Aggarwal, Nico Döttling, Jesper Buus Nielsen, Maciej Obremski, and Erick Purwanto. Continuous non-malleable codes in the 8-split-state model. Cryptology ePrint Archive, Report 2017/357, 2017. https://eprint.iacr.org/2017/357.

[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://eprint.iacr.org/2018/1147.

[AGM+15] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In *TCC*, pages 375–397, 2015.

[AIS18] Prabhanjan Ananth, Yuval Ishai, and Amit Sahai. Private circuits: A modular approach. In *CRYPTO*, pages 427–455, 2018.

[BGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *STOC*, pages 1–10, 1988.

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS 1979 National Computer Conference*, 48:313–317, 1979.

[BPRW16] Allison Bishop, Valerio Pastro, Rajmohan Rajaraman, and Daniel Wichs. Essentially optimal robust secret sharing with maximal corruptions. In *EUROCRYPT*, pages 58–86, 2016.

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

[BSGV96] Carlo Blundo, Alfredo De Santis, Luisa Gargano, and Ugo Vaccaro. On the information rate of secret sharing schemes. *Theor. Comput. Sci.*, 154(2):283–306, 1996.

[CCD88] David Chaum, Claude Crépeau, and Ivan Damgård. Multiparty unconditionally secure protocols (extended abstract). In *STOC*, pages 11–19, 1988.

[CDTV16] Sandro Coretti, Yevgeniy Dodis, Björn Tackmann, and Daniele Venturi. Non-malleable encryption: Simpler, shorter, stronger. In *TCC*, pages 306–335, 2016.

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

[CG14a] Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. In *Innovations in Theoretical Computer Science*, pages 155–168, 2014.

[CG14b] Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bitwise and split-state tampering. In *TCC*, pages 440–464, 2014.

[CGL16] Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. In *STOC*, pages 285–298, 2016.

[CKR16] Nishanth Chandran, Bhavana Kanukurthi, and Srinivasan Raghuraman. Information-theoretic local non-malleable codes and their applications. In *TCC*, pages 367–392, 2016.

[CMTV15] Sandro Coretti, Ueli Maurer, Björn Tackmann, and Daniele Venturi. From single-bit to multi-bit public-key encryption via non-malleable codes. In *TCC*, pages 532–560, 2015.

[CSV93] Marco Carpentieri, Alfredo De Santis, and Ugo Vaccaro. Size of shares and probability of cheating in threshold schemes. In *EUROCRYPT*, pages 118–125, 1993.

[CZ14] Eshan Chattopadhyay and David Zuckerman. Non-malleable codes against constant split-state tampering. In *FOCS*, pages 306–315, 2014.

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

[DDWY93] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. Perfectly secure message transmission. *J. ACM*, 40(1):17–47, 1993.

[DF91] Yvo Desmedt and Yair Frankel. Shared generation of authenticators and signatures (extended abstract). In *CRYPTO*, pages 457–469, 1991.

[DHLW10] Yevgeniy Dodis, Kristiyan Haralambiev, Adriana López-Alt, and Daniel Wichs. Cryptography against continuous memory attacks. In *FOCS*, pages 511–520, 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.

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

[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.

[FNV17] Antonio Faonio, Jesper Buus Nielsen, and Daniele Venturi. Fully leakage-resilient signatures revisited: Graceful degradation, noisy leakage, and construction in the bounded-retrieval model. *Theor. Comput. Sci.*, 660:23–56, 2017.

[FRR+10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In *EUROCRYPT*, pages 135–156, 2010.

[GJK15] Vipul Goyal, Aayush Jain, and Dakshita Khurana. Witness signatures and non-malleable multi-prover zero-knowledge proofs. Cryptology ePrint Archive, Report 2015/1095, 2015. http://eprint.iacr.org/2015/1095.

[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.

[GLM+04] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In *TCC*, pages 258–277, 2004.

[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.

[Has] HashiCorp. The Vault project. https://www.vaultproject.io/. Accessed: 2018-12-22.

[ISW03] Yuval Ishai, Amit Sahai, and David A. Wagner. Private circuits: Securing hardware against probing attacks. In *CRYPTO*, pages 463–481, 2003.

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

[KOS17] Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Four-state non-malleable codes with explicit constant rate. In *TCC*, pages 344–375, 2017.

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

[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.

[NS12] Moni Naor and Gil Segev. Public-key cryptosystems resilient to key leakage. *SIAM J. Comput.*, 41(4):772–814, 2012.

[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.

[RB89] Tal Rabin and Michael Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority (extended abstract). In *STOC*, pages 73–85, 1989.

[RB07] Phillip Rogaway and Mihir Bellare. Robust computational secret sharing and a unified account of classical secret-sharing goals. In *CCS*, pages 172–184, 2007.

[Sha79] Adi Shamir. How to share a secret. *Commun. ACM*, 22(11):612–613, 1979.

[Sho04] Victor Shoup. Sequences of games: A tool for taming complexity in security proofs. Cryptology ePrint Archive, Report 2004/332, 2004. http://eprint.iacr.org/2004/332.

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