Continuously Non-Malleable Secret Sharing: Joint Tampering, Plain Model and Capacity*

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

[1] *Sapienza University of Rome*
[2] *EURECOM*

September 3, 2021

Abstract

We study non-malleable secret sharing against *joint leakage* and *joint tampering* attacks. Our main result is the first *threshold* secret sharing scheme in the *plain model* achieving resilience to noisy-leakage and continuous tampering. The above holds under (necessary) minimal computational assumptions (*i.e.*, the existence of one-to-one one-way functions), and in a model where the adversary commits to a fixed partition of all the shares into non-overlapping subsets of at most $t-1$ shares (where t is the reconstruction threshold), and subsequently jointly leaks from and tampers with the shares within each partition.

We also study the *capacity* (*i.e.*, the maximum achievable asymptotic information rate) of continuously non-malleable secret sharing against joint continuous tampering attacks. In particular, we prove that whenever the attacker can tamper jointly with $k > t/2$ shares, the capacity is at most $t-k$. The rate of our construction matches this upper bound.

An important corollary of our results is the first non-malleable secret sharing scheme against *independent tampering* attacks breaking the rate-one barrier (under the same computational assumptions as above).

Keywords: secret sharing; non-malleability; leakage resilience.

*An abridged version of this paper appears in the Proceedings of the 19th Theory of Cryptography Conference (TCC 2021). This is the full version. The first and last author acknowledge support by Sapienza University under the grant SPECTRA.

Contents

1 Introduction

A t-out-of-n secret sharing scheme [Bla79, Sha79] allows to distribute a message into n shares in such a way that: (i) given t or more shares we can reconstruct the original message; and (ii) any attacker corrupting strictly less than t share holders has no information about the message. The parameter t is called the reconstruction threshold, and a scheme with the above properties is called a *threshold* secret sharing. An important efficiency parameter of secret sharing is the so-called *information rate*, which equals the ratio between the length of the message and the maximum length of a share.

Goyal and Kumar [GK18a] introduced *non-malleable* secret sharing, which further satisfies the following guarantee: (iii) no attacker tampering with possibly all of the shares can generate a valid secret sharing of a message which is related to the original shared value. This notion was inspired by the related concept of non-malleable codes defined by Dziembowski, Pietrzak and Wichs [DPW10], and by similar notions in the setting of non-malleable cryptography [DDN91, DDN00].

Clearly, we must put some restriction on how the attacker can tamper with the shares (as if she can tamper with all of them in a joint manner she can reconstruct the message and compute a valid secret sharing of a related value). The original paper by Goyal and Kumar constructed threshold secret sharing schemes both against *independent tampering* attacks (*i.e.*, each share can be tampered arbitrarily yet independently) and *joint tampering* attacks (*i.e.*, the attacker can partition any set of t shares into two non-empty subsets and tamper jointly with the shares contained in each subset). This initial result spurred further research on the subject, yielding non-malleable secret sharing schemes with additional properties and with resilience to stronger tampering attacks. We review the state of the art for joint tampering (which is the focus of this paper) below, and in Tab. 1, and refer the reader to §1.4 for additional related work.

1.1 Non-Malleability Against Joint Tampering

In a follow-up paper, Goyal and Kumar [GK18b] constructed n-out-of-n non-malleable secret sharing in a stronger tampering model where the attacker can partition the n shares into two (possibly overlapping) subsets of its choice, and then jointly tamper with the shares in each of the subsets independently. Similarly to the construction in [GK18a], the information rate of this scheme asymptotically reaches zero (when the message length goes to infinity).

Brian, Faonio and Venturi [BFV19] showed how to compile any *leakage-resilient* secret sharing into a *continuously* non-malleable one [FV19, FMNV14] using a trusted setup (and computational assumptions). Here, leakage resilience refers to the guarantee that the secret remains hidden even given leakage from the shares. Continuous non-malleability refers to the ability of the attacker to adaptively tamper poly-many[1] times with the same target secret sharing.

When the initial secret sharing is resilient to joint-leakage attacks, the compiled scheme tolerates continuous joint-tampering and joint-leakage attacks in a model where the adversary commits to a partition $\mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_m)$ of $[n]$ into m disjoint subsets of size at most k at the beginning of the experiment, and subsequently can tamper with and leak from the shares within each subset in an adaptive fashion. The reconstruction set $\mathcal{T}$ (with cardinality $|\mathcal{T}| \geq t$) associated to each tampering query can be chosen adaptively, a feature sometimes known under the name of *adaptive concurrent reconstruction* [ADN$^+$19]. In this work, we dub secret sharing schemes that are secure in the above setting as leakage-resilient continuously non-malleable under *selective* k-joint leakage and tampering attacks. By plugging recent constructions of

[1]The only (necessary) restriction is that the experiment self-destructs after the first tampering query yielding an invalid secret sharing.

Reference	Access Structure	Non-Malleability	Leakage	Rate	Assumptions	Partitioning
[GK18a]	Threshold ($t \geq 2$)	1-Time ($k < t$)	✗	$\Theta\left(\lvert\mu\rvert^{-9}\right)$	—	Disjoint
[GK18b]	Threshold ($t = n$)	1-Time ($k < t$)	✗	$\Theta(\lvert\mu\rvert^{-6})$	—	Overlapping
[BFV19]	General	Continuous ($k \leq O(\log n)$)	Bounded	$\mathsf{poly}(\lvert\mu\rvert, n, \ell, \lambda)^{-1}$	TDPs, CRHs, CRS	Selective, Disjoint
	Threshold ($t \geq 2$)	Continuous ($k < t$)	Bounded	$\mathsf{poly}(\lvert\mu\rvert, n, \ell, \lambda)^{-1}$	TDPs, CRHs, CRS	Selective, Disjoint
	Threshold ($t = n$)	Continuous ($k \leq 0.99n$)	Bounded	$\mathsf{poly}(\lvert\mu\rvert, n, \ell, \lambda)^{-1}$	TDPs, CRHs, CRS	Selective, Disjoint
[BFO+20]	Threshold ($t \geq 2$)	p-Time ($k < t$)	✗	$\mathsf{poly}(\lvert\mu\rvert, n, p, \lambda)^{-1}$	1-to-1 OWFs	Selective, Disjoint
	Threshold ($t = n$)	p-Time ($k < t$)	✗	$\mathsf{poly}(\lvert\mu\rvert, n, p, \lambda)^{-1}$	1-to-1 OWFs	Selective, Disjoint
	General	p-Time ($k \leq O(\log n)$)	✗	$\mathsf{poly}(\lvert\mu\rvert, n, p, \lambda)^{-1}$	1-to-1 OWFs	Semi-Adaptive, Disjoint
	Threshold ($t \geq 2$)	p-Time ($k \leq O(t/\log t)$)	✗	$\mathsf{poly}(\lvert\mu\rvert, n, p, \lambda)^{-1}$	1-to-1 OWFs	Semi-Adaptive, Disjoint
	Threshold ($t = n$)	p-Time ($k \leq 0.99n$)	✗	$\mathsf{poly}(\lvert\mu\rvert, n, p, \lambda)^{-1}$	1-to-1 OWFs	Semi-Adaptive, Disjoint
[GSZ20]	Threshold ($t \geq 2$)	1-Time ($k < t$)	✗	$\mathsf{poly}(\lvert\mu\rvert, n, \ell, \lambda)^{-1}$	—	Overlapping
§A, §6.1	Threshold ($t \geq 2$)	1-Time ($k < t$)	Noisy	$\mathsf{poly}(\lvert\mu\rvert, n, \ell, \lambda)^{-1}$	—	Disjoint
§6.2	Threshold ($t \geq 2n/3$)	Continuous ($k < t$)	Noisy	$1 - \mathsf{poly}(n, \ell, \lambda) \cdot \lvert\mu\rvert^{-1}$	1-to-1 OWFs	Selective, Disjoint
§B, §6.2	Threshold ($t \geq 2n/3$)	Continuous ($k < t$)	Noisy	$t - \mathsf{poly}(n, \ell, \lambda) \cdot \lvert\mu\rvert^{-1}$	ROM/AGM	Selective, Disjoint
§6.3	Threshold ($t \geq 2n/3$)	1-Time ($k = 1$)	Noisy	$t/2 - \mathsf{poly}(n, \ell, \lambda) \cdot \lvert\mu\rvert^{-1}$	1-to-1 OWFs	—

Table 1: State-of-the-art non-malleable secret sharing schemes tolerating joint tampering and leakage attacks. The value n denotes the number of parties, $|\mu|$ is the size of the message, ℓ denotes the leakage parameter, p is the number of tampering queries, λ denotes the security parameter, t is the reconstruction threshold, and k is the maximal number of shares that can be tampered jointly. Semi-adaptive partitioning refers to the ability of the attacker to change the way the target shares are partitioned within each leakage/tampering query in a somewhat restricted manner [BFO+20]. OWFs stands for "one-way functions", TDPs for "(doubly-enhanced) trapdoor permutations", CRHs for "collision-resistant hash functions", CRS for "common reference string", ROM for "random oracle model", and AGM for "algebraic group model". For readability, in the last two rows the values for the rates are displayed as lower bounds.

leakage-resilient secret sharing under joint-leakage attacks [CGGL20, KMS19, KMZ20], we get rate-zero schemes satisfying this notion either for arbitrary access structures with $k = O(\log n)$, or for threshold access structures with $k = t - 1$ (which is optimal).

Brian *et al.* [BFO+20] showed how to compile any leakage-resilient one-time non-malleable secret sharing scheme with *statistical security* under selective k-joint leakage and tampering attacks into a p-time *computationally* non-malleable secret sharing under selective k-joint tampering attacks in the plain model (assuming one-to-one one-way functions). Here, p-time non-malleability means that the number of tolerated tampering queries is a-priori bounded (and the length of the shares depends on it). Moreover, when the initial secret sharing is secure under *adaptive* k-joint leakage and tampering attacks (*i.e.*, the attacker can change the partition adaptively within each leakage/tampering query), the compiled scheme satisfies p-time non-malleability under *semi-adaptive*[2] k-joint tampering attacks too. Combined with [CGGL20, GK18a, GK18b, KMS19, KMZ20], the results of [BFO+20] ultimately yield rate-zero schemes satisfying the latter notion either for arbitrary access structures with $k = O(\log n)$, or for threshold access structures with $k = O(t/\log t)$ (and $k = t - 1$ in case of selective partitioning).

Finally, Goyal, Srinivasan and Zhu [GSZ20] obtain rate-zero one-time non-malleable threshold secret sharing with statistical security against t-cover free tampering, which intuitively requires that every share is tampered together with at most $t - 2$ other shares (this model includes disjoint tampering as a special case).

1.2 Our Results

A major drawback of [BFO+20] is that it only satisfies computational p-time non-malleability. This is far from optimal, as the notion could in principle be obtained information theoretically. On the other hand, [BFV19] achieves continuous non-malleability under selective partitioning

[2] In this setting, once a subset of shares has been tampered with jointly, that subset is always either tampered jointly or not modified by future tampering queries.

of the shares at the price of assuming a trusted setup (and minimal, inherent, computational assumptions).

Our main contribution is a construction of leakage-resilient *continuously* non-malleable t-out-of-n secret sharing under *selective* k-joint leakage and tampering attacks in the *plain model* (assuming one-to-one one-way functions), for any $k < t$ and $t \geq 2n/3$. Furthermore, our scheme achieves the following features:

- The information rate asymptotically reaches 1, which we show to be optimal.
- Leakage resilience holds in the stronger (and more practical) model where the length of the leakage (from each subset in the fixed partition of the shares) is arbitrary, so long as it does not decrease the min-entropy of the shares by more than ℓ bits (where $\ell \geq 0$ is called the *noisy-leakage* parameter).

An interesting corollary of our results is the first non-malleable t-out-of-n secret sharing under *independent* tampering attacks in the plain model (assuming one-to-one one-way functions) breaking the rate-one barrier (for $t \geq 2n/3$). In particular, we obtain asymptotic rate $t/2$.

All previous non-malleable secret sharing schemes against joint tampering had rate zero, and the only scheme with rate one was secure in the much weaker setting of independent tampering [FV19]. In this vein, our result shows that the lower bounds on the rate of leakage-resilient and non-malleable secret sharing [BFO+20, NS20] can be circumvented in the computational setting. We stress that cryptographic assumptions are inherent for continuous non-malleability [FV19, FMNV14, SV19].

1.3 Overview of Techniques

The construction of our secret sharing schemes consists of two main steps. First, we show how to obtain leakage-resilient continuously non-malleable t-out-of-n secret sharing under selective $(t-1)$-joint leakage and tampering attacks in the plain model, with asymptotic rate zero. Second, we show how to boost the asymptotic rate to one generically.

1.3.1 Rate-Zero Construction

In order to explain our techniques, it will be useful to recall the construction of leakage-resilient continuously non-malleable t-out-of-n secret sharing under independent[3] tampering attacks in the plain model, by Brian, Faonio and Venturi [BFV19] (which in turn builds on the construction by Ostrovsky *et al.* [OPVV18]). For simplicity, let us focus on the case $t = n = 2$ (*i.e.*, so-called leakage-resilient non-malleable *split-state codes*).

Here, one takes the message μ and commits to it via a non-interactive (perfectly binding) commitment scheme using random coins ρ, yielding a commitment γ. Hence, the string $\mu||\rho$ is secret shared using a leakage-resilient one-time non-malleable 2-out-of-2 secret sharing scheme. This yields shares (σ_1, σ_2), so that the final shares become $\sigma_1^* = (\gamma, \sigma_1)$ and $\sigma_2^* = (\gamma, \sigma_2)$. In the following, we will refer to σ_1^* as the left share and to σ_2^* as the right share. The reconstruction algorithm proceeds naturally by first checking that the left and right commitment are the same value γ, and thus reconstructing the string $\mu||\rho$ from the shares (σ_1, σ_2) and outputting μ if and only if (μ, ρ) is a valid opening for the commitment.

The security analysis crucially relies on the assumption that the underlying one-time non-malleable secret sharing scheme has statistical security. In particular, the main hurdle in the proof is to reduce continuous non-malleability to one-time non-malleability. Brian *et al.* overcome this obstacle using the following strategy. First, they move to a mental[4] experiment in

[3] i.e., one-joint leakage and tampering attacks.

[4] In the hybrid experiment, one also needs to adjust the reconstruction algorithm so that an attacker cannot

which (σ_1, σ_2) is a secret sharing of $\mu||\rho'$, where ρ' is random and independent of the random coins ρ used to compute the commitment γ. Second, they reduce a distinguisher between the real and mental experiment to an (inefficient) attacker against *statistical* leakage-resilient one-time non-malleability. The key idea of this reduction is to simulate multiple tampering queries by leaking the commitments $\tilde{\gamma}_1$ and $\tilde{\gamma}_2$ contained in the tampered shares $\tilde{\sigma}_1$ and $\tilde{\sigma}_2$. If $\tilde{\gamma}_1 \neq \tilde{\gamma}_2$ the reduction outputs $\perp$ and self-destructs, and otherwise it brute forces the commitment and outputs the corresponding message.

In order for the reduction to go through, one needs to argue that it does not ask too much leakage. Here is where *noisy-leakage* resilience kicks in. Brian *et al.* assume that the underlying secret sharing satisfies an additional property known as *conditional independence*: For any message, the right (resp. left) share drops the conditional average min-entropy of the left (resp. right) share by some (possibly small) parameter $d \in \mathbb{N}$. This property is satisfied by existing leakage-resilient one-time non-malleable t-out-of-n secret sharing schemes in the independent leakage and tampering model [BFV19, OPVV18]. Now, the point is that, so long as the commitments $\tilde{\gamma}_1$ and $\tilde{\gamma}_2$ are equal, the leakage on the left (resp. right) share can be thought of as a function of the right (resp. left share), and thus the overall leakage does not drop the min-entropy by more than $d + |\gamma| + O(\log \lambda)$ where the additional loss $|\gamma|$ corresponds to the tampering query in which $\tilde{\gamma}_1 \neq \tilde{\gamma}_2$ (and the term $O(\log \lambda)$ corresponds to the index of such query). Luckily, the latter happens only once because after that a self-destruct is triggered, which ultimately allows the reduction to go through.

1st Barrier: leakage-resilient one-time non-malleability. The first (obvious) difficulty in order to generalize the above construction to the joint-tampering setting is that we need a leakage-resilient one-time non-malleable t-out-of-n secret sharing under joint leakage and tampering attacks, which was not known. We overcome this difficulty by suitably modifying a recent construction by Goyal, Srinivasan and Zhu [GSZ20], which we briefly recall below.

The sharing procedure first shares the message μ using a t-out-of-n secret sharing scheme Π. Then, given the resulting shares $(\sigma_1, \ldots, \sigma_n)$, it encodes each σ_i into a codeword $(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$ using a t-time non-malleable split-state code Π'. Finally, it uses again the t-out-of-n secret sharing scheme Π to obtain shares $(\sigma^{(1)}_{\mathsf{R},i}, \ldots, \sigma^{(n)}_{\mathsf{R},i})$ of the right part of the codeword $\sigma_{\mathsf{R},i}$ for each $i \in [n]$. This yields shares $(\sigma_{\mathsf{L},i})_{i\in[n]}$ and $(\sigma^{(j)}_{\mathsf{R},i})_{i,j\in[n]}$, which are distributed to the players by letting $\sigma^*_i = (\sigma_{\mathsf{L},i}, (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]})$ for all $i \in [n]$.

Goyal *et al.* proved that the construction is a t-out-of-n one-time non-malleable secret sharing scheme with statistical security under k-joint tampering[5] attacks for any $k < t$. The original analysis did not consider leakage resilience. However, it is not too hard to lift the proof to the setting in which the attacker is also allowed to perform noisy-leakage attacks, so long as the secret sharing scheme Π' is noisy-leakage-resilient t-time non-malleable. For completeness, we work out the details in §A of the appendix.

2nd Barrier: conditional independence. The second barrier is more subtle. One may think that after obtaining leakage-resilient one-time non-malleable t-out-of-n secret sharing under joint leakage and tampering attacks we would be done by using this scheme instead of the t-out-of-n one-time non-malleable secret sharing under independent leakage and tampering attacks in the construction by Brian, Faonio and Venturi [BFV19].

distinguish between the hybrid and the original experiment by mauling σ_1, σ_2 without changing the underlying shared value. In the description, we omit these details to simplify the exposition.

[5] Actually, Goyal *et al.* prove security in the more general setting of t-cover-free tampering, in which, intuitively, the subsets of the partition of the shares may overlap.

Unfortunately, generalizing their analysis based on conditional independence to the case of joint leakage and tampering attacks is not straightforward. Recall that the reduction cannot perform too much noisy leakage. In our case, the reduction needs to leak the tampered commitments $(\tilde{\gamma}_i)_{i\in[m]}$, *i.e.* one commitment for each set of tampered shares.[6]

For concreteness, let us focus on the leakage performed on the shares within a fixed subset $\mathcal{B}_i$ of the partition. While it is still true that, before self-destruct, the tampered commitment $\tilde{\gamma}_i$ corresponding to a tampering query $(\mathcal{T}, (f_1, \ldots, f_m))$ can be thought of as a function of the shares within $\mathcal{T}\backslash\mathcal{B}_i$, the fact that the reconstruction set $\mathcal{T}$ can change across different tampering queries would require the following flavor of conditional independence: For any message and any unauthorized subset $\mathcal{U}$, the shares within $[n] \setminus \mathcal{U}$ drop the conditional average min-entropy of the shares within $\mathcal{U}$ by some (possibly small) parameter $d \in \mathbb{N}$.

However, the leakage-resilient one-time non-malleable t-out-of-n secret sharing under joint leakage and tampering attacks we described in the previous paragraph does not satisfy such a strong flavor of conditional independence. This is because, *e.g.*, in Shamir's secret sharing with $t < n$, the conditional average min-entropy of the first share conditioned on all other shares is zero (as given any t shares we can interpolate the polynomial used to determine the shares). In §4, we show how to circumvent this problem by leaving off one level of abstraction. Namely, we analyze the compiler from [BFV19] when instantiated with (our leakage-resilient variant of) the secret sharing scheme from [GSZ20]. Intuitively, this allows us to perform an hybrid argument where at each step we reduce to leakage-resilient t-time non-malleability of the underlying 2-out-of-2 secret sharing schemes, and thus to only assume the standard flavor of conditional independence for such kind of secret sharing schemes, which is much easier to achieve.

1.3.2 Capacity of Continuously Non-Malleable Secret Sharing

The above construction still has shares of length polynomial in the number of parties, the leakage parameter, the security parameter, and the message size, thus yielding information rate asymptotically reaching 0. Motivated by this limitation, we study the *capacity* (*i.e.*, the best achievable rate) of continuously non-malleable threshold secret sharing against joint tampering. As our main negative result, in §5.1, we establish that whenever the attacker can tamper jointly with $k > t/2$ shares, the capacity is at most $t - k$.

The latter can be seen as follows. Let Π be any continuously non-malleable threshold secret sharing scheme against joint tampering with at most k shares. Consider the non-interactive commitment scheme whose commit procedure does a secret sharing of the message μ obtaining $(\sigma_1, \ldots, \sigma_n)$ using a continuous non-malleable secret sharing scheme, and finally outputs $\gamma = (\sigma_1, \ldots, \sigma_{t-k})$. If we can show that this commitment is perfectly binding then $|\mu| \leq |\gamma| = (t-k)\cdot s$ (where s is the size of a single share), and thus the rate of Π must be at most $t - k$. Assume the commitment scheme is not perfectly binding, namely, there exist two distinct messages $\mu^{(0)}$ and $\mu^{(1)}$, along with openings ρ_0 and ρ_1, such that $\gamma = (\sigma_1, \ldots, \sigma_{t-k})$ is consistent with both $(\mu^{(0)}, \rho_0)$ and $(\mu^{(1)}, \rho_1)$.

We show how to construct an efficient adversary breaking continuous non-malleability of Π. Let $\sigma^* = (\sigma_1^*, \ldots, \sigma_n^*)$ be the target secret sharing. The adversary computes offline $\sigma^{(0)} = (\sigma_1^{(0)}, \ldots, \sigma_n^{(0)})$ and $\sigma^{(1)} = (\sigma_1^{(1)}, \ldots, \sigma_n^{(1)})$ by secret sharing $\mu^{(0)}$ with coins ρ_0 and $\mu^{(1)}$ with coins ρ_1. Note that, by construction, the first $t - k$ shares of $\sigma^{(0)}$ and $\sigma^{(1)}$ are identical. Hence, the attacker tampers repeatedly with σ^* as described below:

[6] Note that in case the tampered commitments within one of the subsets $\mathcal{B}_i$ are not equal, the reduction can immediately self-destruct.

- It fixes the partition $\mathcal{B}$ of $[n]$ to be $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2, \mathcal{B}_3)$, such that $\mathcal{B}_1 = [t-k]$ and $\mathcal{B}_2 = [t] \setminus [t-k]$, and $\mathcal{B}_3$ is any k-sized partition of $[n] \setminus [t]$. The fact that $k > t/2$ ensures that $\mathcal{B}$ is a k-sized partition of $[n]$.
- It defines the tampering query f that replaces the first $t-k$ shares of σ^* with the corresponding shares of $\sigma^{(0)}$ (which are the same of $\sigma^{(1)}$), and the shares of σ^* within $\mathcal{B}_2$ with the corresponding shares of either $\sigma^{(0)}$ or $\sigma^{(1)}$ depending on whether the i-th bit of $\sigma^*_{\mathcal{B}_2}$ is either zero or one. The shares within $\mathcal{B}_3$ are unchanged.
- It submits $f = (f_1, f_2, f_3)$ to the tampering oracle along with the reconstruction set $\mathcal{T} = [t]$. Note that it is irrelevant how the shares within $\mathcal{B}_3$ were modified, as those are not included in $[t]$.

Using the above tampering query, the attacker learns the i-th bit of $\sigma^*_{\mathcal{B}_2}$. After all the shares $\sigma^*_{\mathcal{B}_2}$ are obtained, it is trivial to break non-malleability by hard-wiring those shares in the tampering function that is allowed to modify the shares within $\mathcal{B}_1$ (as $\mathcal{B}_1 \cup \mathcal{B}_2 = [t]$, which allows to reconstruct the target message).

1.3.3 Rate-One Construction (and More)

The above upper bound shows that the best possible rate of continuously non-malleable secret sharing against $(t-1)$-joint tampering attacks is 1. As our last contribution, in §5.2, we show that such a rate is achievable under the same computational assumptions needed for our rate-zero construction. We do so by revisiting a paradigm originally due to Krawczyk [Kra94] for boosting the rate of classical threshold secret sharing.

Let Π be a threshold secret sharing scheme with rate zero. The main idea is to use Π to share the private key κ of a symmetric encryption scheme, obtaining shares $(\kappa_1, \ldots, \kappa_n)$; hence, we encrypt the message μ and use an information dispersal in order to distribute the ciphertext γ (along with the shares of the key) to the parties. Namely, by denoting with γ_i the i-th share of the information dispersal, the final share of party i is going to be $\sigma_i = (\kappa_i, \gamma_i)$. Krawczyk proved that computational privacy of this construction follows easily from the privacy property of the underlying secret sharing scheme, along with semantic security of encryption. Moreover, let t^* be the reconstruction threshold of the information dispersal, by setting $t^* = t$, the rate of the scheme asymptotically reaches t (thus share size is smaller than the message size) the reason is that the size of the shares of the key do not depend on the size of the message. Such a rate is known to be optimal.

Unfortunately, our capacity upper bound immediately implies that the above construction cannot yield a continuously non-malleable secret sharing scheme against $(t-1)$-joint leakage and tampering attacks when $t^* = t$. In fact the best possible rate is one, which corresponds to setting the reconstruction threshold of the information dispersal to $t^* = 1$, essentially meaning that the same ciphertext must be repeated in every share, *i.e.* $\sigma_i = (\kappa_i, \gamma)$. The main step of the proof is to transition to a mental experiment in which the shares $(\sigma_1, \ldots, \sigma_n)$ are computed by sharing an unrelated key $\hat{\kappa}$, and reduce a distinguisher between this experiment and the original game to an adversary attacking leakage-resilient continuous non-malleability of the underlying secret sharing scheme. In particular, the reduction needs to obtain the tampered ciphertexts $(\tilde{\gamma}_i)_{i \in [m]}$, *i.e.* one ciphertext for each set of tampered shares,[7] so that it can either decrypt the ciphertext $\tilde{\gamma} := \tilde{\gamma}_1 = \cdots = \tilde{\gamma}_m$ with the tampered key $\tilde{\kappa}$ obtained from the challenger, or self-destruct in case the ciphertexts within the reconstruction set $\mathcal{T}$ specified by the distinguisher are not all equal.

One possibility would be to obtain each ciphertext $\tilde{\gamma}_i$ via leakage queries, and then to argue

[7]Note that in case the tampered ciphertexts within one of the subsets $\mathcal{B}_i$ are not equal, the reduction can immediately self-destruct.

that this does not reduce the conditional average min-entropy by too much since, before self-destruct, the ciphertext $\tilde{\gamma}_i$ can be thought of as a function of the other shares. However, the possibility to change the reconstruction set $\mathcal{T}$ adaptively within each tampering query, would require us to assume the strong flavor of conditional independence discussed in §1.3.1 (which we do not know how to achieve). Instead, we use a different technique, and obtain the tampered ciphertexts via multiple tampering queries (and thus with no leakage). In particular, given a tampering query from the adversary, our reduction sends $|\gamma|+1$ different tampering queries. The first $|\gamma|$ queries extract the tampered ciphertext $\tilde{\gamma}$ one bit at a time, while the last tampering query extracts the secret key used to encrypt the message. To perform the first $|\gamma|$ queries we fix two valid secret sharing $(\sigma_1^{(0)}, \ldots, \sigma_n^{(0)})$ and $(\sigma_1^{(1)}, \ldots, \sigma_n^{(1)})$ for two distinct messages $\mu^{(0)}$ and $\mu^{(1)}$. The i-th tampering query coordinates its outputs using the i-th bits of the tampered ciphertexts. If the tampered ciphertexts are all the same then the shares produced by the i-th tampering function are either $(\sigma_1^{(0)}, \ldots, \sigma_n^{(0)})$ or $(\sigma_1^{(1)}, \ldots, \sigma_n^{(1)})$ (depending on the i-th bit of $\tilde{\gamma}$). On the other hand, if the tampered ciphertexts are not all the same, let j, j' be the indexes such that the ciphertexts $\tilde{\gamma}_j$ and $\tilde{\gamma}_{j'}$ differ on the i-th bit, then the tampering function outputs $(\sigma_k^{(0)})_{k \in \mathcal{B}_j}$ and $(\sigma_k^{(1)})_{k \in \mathcal{B}_{j'}}$ which triggers a self-destruct.

Finally, we show how to bypass the limitations imposed by our capacity upper bound by extending the ideas behind our rate compiler in two directions:

- First, we analyze the rate compiler assuming the reconstruction threshold of the information dispersal is any value $t^* \leq t-1$ and the adversary is limited to what we call *t^*-intersecting tampering*: Each tampering query $(\mathcal{T}, f)$ output by the attacker is such that, for all subsets $\mathcal{B}_i$ of the partition $\mathcal{B}$, either $\mathcal{B}_i \cap \mathcal{T} = \emptyset$ or $|\mathcal{B}_i \cap \mathcal{T}| \geq t^*$. Note that this yields asymptotic rate t^* for the final secret sharing scheme (without contradicting our capacity upper bound which does not consider t^*-intersecting tampering). An important consequence of this generalization is that it yields the first non-malleable secret sharing scheme against *independent* tampering attacks with positive rate $t/2$. We achieve this by setting $t^* = t/2$, and by reducing an attacker for independent tampering to a t^*-intersecting-tampering attacker that partitions the shares into two blocks of $t/2$ shares each.
- Second, in §B of the appendix, we show that optimal asymptotic rate t can be obtained both in the random oracle model (ROM) and in the algebraic group model (AGM). More in detail, we consider the same rate compiler but where the sharing procedure additionally appends to each of the shares a *cryptographic hash* h of the ciphertext γ. The reconstruction procedure checks that the hash values are consistent (*i.e.*, they are all the same and equal to the hash of γ). In the ROM, we model the hash function as a random oracle, while in the AGM we instantiate it using the so-called Pedersen's hash function. In the reductions, we show that one can extract the tampered ciphertext $\tilde{\gamma}$ from the tampered hash $\tilde{h}$ corresponding to each tampering query, without the need to rely on leakage resilience of the underlying t-out-of-n continuously non-malleable secret sharing scheme.

  These results do not contradict our capacity upper bound which is for the plain model only. Informally, this is true because both in the ROM and in the AGM we can obtain extractable commitment schemes that are succinct (*i.e.*, where the size of a commitment is shorter than the size of the message).

1.3.4 Concrete Instantiations

Finally, in §6, we show how to instantiate the building blocks required for all of our constructions. We construct a leakage-resilient t-time non-malleable split-state code by generalizing the black-box transformation of Ball *et al.* [BGW19] to the setting of noisy-leakage and multiple-tampering attacks. This construction satisfies the conditional independence property that is needed for the analysis of our secret sharing scheme from §4.

Putting everything together, for any $t \geq 2$ (resp. $t \geq 2n/3$) we obtain the first statistically-secure (resp. computationally-secure) t-out-of-n noisy-leakage-resilient one-time (resp. continuously) non-malleable secret sharing under selective $(t-1)$-joint leakage and tampering attacks with asymptotic rate zero (resp. one), as also highlighted in Table 1.

1.4 Related Work

Non-malleable secret sharing with security against one-time joint-tampering attacks further exists for certain restricted tampering classes including polynomials of bounded degree (see Ball *et al.* [BCL+20]) and affine tampering (see Lin *et al.* [LCG+19]), and for ramp secret sharing (see Chattopadhyay and Li [CL18]).

A series of papers focuses on constructing non-malleable secret sharing in the weaker setting of independent tampering attacks [ADN+19, BS19, BFV19, FV19, GK18a, GK18b, SV19]. In particular, Faonio and Venturi [FV19], as well as Brian *et al.* [BFV19], previously analyzed a simplified version of the rate compiler of Krawczyk [Kra94] and the non-malleable code construction by Ostrovsky *et al.* [OPVV18] (generalized to threshold secret sharing) in the setting of both independent and joint leakage and tampering attacks. However, their analysis requires a non-standard[8] flavor of noisy-leakage resilience for the underlying rate-zero secret sharing scheme which we show to be not necessary in this work.

2 Standard Definitions

For a string $x \in \{0,1\}^*$, we denote its length by $|x|$; if $x, y \in \{0,1\}^*$ are two strings, we denote by $x||y$ the concatenation of x and y. If $\mathcal{X}$ is a set, $|\mathcal{X}|$ represents the number of elements in $\mathcal{X}$. We denote by $[n]$ the set $\{1,\ldots,n\}$. For a set of indices $\mathcal{I} = (i_1,\ldots,i_t)$ and a vector $x = (x_1,\ldots,x_n)$, we write $x_{\mathcal{I}}$ to denote the vector $(x_{i_1},\ldots,x_{i_t})$. 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 ρ) and output y; the value y is a random variable and $\mathsf{A}(x;\rho)$ denotes a run of A on input x and randomness ρ. An algorithm A is *probabilistic polynomial-time* (PPT for short) if A is randomized and for any input $x, \rho \in \{0,1\}^*$, the computation of $\mathsf{A}(x;\rho)$ 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 *polynomial* (in the security parameter) if $p(\lambda) \in \Theta(\lambda^c)$ for some constant $c > 0$; we sometimes write $\mathsf{poly}(\lambda)$ for an unspecified polynomial. A function $\nu : \mathbb{N} \to [0,1]$ is *negligible* (in the security parameter) 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 sometimes write $\mathsf{negl}(\lambda)$ to denote an unspecified negligible function. We assume that the security parameter is given as input (in unary) to all algorithms.

Random variables. For a random variable $\mathbf{X}$, we write $\Pr[\mathbf{X} = x]$ 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

[8] They require that the leakage on the i-th share does not drop the conditional average min-entropy of the share i *conditioned on all other shares* $j \neq i$ by too much. This additional requirement makes their rate compiler incompatible with the non-malleable secret sharing scheme by Brian, Faonio, Obremski, Simkin and Venturi [BFO+20] which does not satisfy this property.

distance between two random variables $\mathbf{X}$ and $\mathbf{Y}$ over the same set $\mathcal{X}$ is defined as

$$\Delta(\mathbf{X}, \mathbf{Y}) = \frac{1}{2} \sum_{x \in \mathcal{X}} |\Pr[\mathbf{X} = x] - \Pr[\mathbf{Y} = x]| \,.$$

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, $\mathbf{X} \overset{\mathsf{s}}{\approx} \mathbf{Y}$ to denote that they are *statistically close*, *i.e.* $\Delta(\mathbf{X}_\lambda, \mathbf{Y}_\lambda) \leq \mathsf{negl}(\lambda)$, and $\mathbf{X} \overset{\mathsf{c}}{\approx} \mathbf{Y}$ to denote that they are *computationally indistinguishable*, *i.e.* for every PPT distinguisher D:

$$|\Pr[\mathsf{D}(\mathbf{X}_\lambda) = 1] - \Pr[\mathsf{D}(\mathbf{Y}_\lambda) = 1]| \leq \mathsf{negl}(\lambda).$$

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 that A outputs a decision bit. Given two experiments $\mathbf{G}_\mathsf{A}(\lambda, 0)$ and $\mathbf{G}_\mathsf{A}(\lambda, 1)$, we write $\{\mathbf{G}_\mathsf{A}(\lambda, 0)\}_{\lambda \in \mathbb{N}} \overset{\mathsf{c}}{\approx} \{\mathbf{G}_\mathsf{A}(\lambda, 1)\}_{\lambda \in \mathbb{N}}$ as a shorthand for

$$|\Pr[\mathbf{G}_\mathsf{A}(\lambda, 0) = 1] - \Pr[\mathbf{G}_\mathsf{A}(\lambda, 1) = 1]| \leq \mathsf{negl}(\lambda).$$

The above naturally generalizes to statistical distance, which we denote by $\Delta(\mathbf{G}_\mathsf{A}(\lambda, 0), \mathbf{G}_\mathsf{A}(\lambda, 1))$, in case of *unbounded* adversaries.

Average min-entropy. The min-entropy of a random variable $\mathbf{X}$ with domain $\mathcal{X}$ is $\mathbb{H}_\infty(\mathbf{X}) := -\log \max_{x \in \mathcal{X}} \Pr[\mathbf{X} = x]$, 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 $\widetilde{\mathbb{H}}_\infty(\mathbf{X} \mid \mathbf{Y}) := -\log \mathbb{E}_y \left[2^{-\mathbb{H}_\infty(\mathbf{X} \mid \mathbf{Y} = y)}\right]$ [DORS03]. The lemma below is sometimes known as the "chain rule" for conditional average min-entropy.

Lemma 1 ([DORS03], Lemma 2.2)**.** *Let $\mathbf{X}, \mathbf{Y}, \mathbf{Z}$ be random variables. If $\mathbf{Y}$ has at most 2^ℓ possible values, then $\widetilde{\mathbb{H}}_\infty(\mathbf{X} \mid \mathbf{Y}, \mathbf{Z}) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{X}, \mathbf{Y} \mid \mathbf{Z}) - \ell \geq \widetilde{\mathbb{H}}_\infty(\mathbf{X} \mid \mathbf{Z}) - \ell$. In particular, $\widetilde{\mathbb{H}}_\infty(\mathbf{X} \mid \mathbf{Y}) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{X}, \mathbf{Y}) - \ell \geq \widetilde{\mathbb{H}}_\infty(\mathbf{X}) - \ell$.*

2.1 Non-Interactive Commitment Schemes

A non-interactive commitment scheme Com is a randomized algorithm taking as input a message $\mu \in \mathcal{M}$ and random coins $\rho \in \mathcal{R}$ and outputting a value $\gamma = \mathsf{Com}(\mu; \rho)$ called *commitment*. The pair (μ, ρ) is called *opening*.

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

Definition 1 (Binding)**.** We say that a non-interactive commitment scheme Com is *computationally binding* if for all PPT adversaries A the following is negligible:

$$\Pr\left[\mu' \neq \mu \wedge \mathsf{Com}(\mu'; \rho') = \mathsf{Com}(\mu; \rho) \;\middle|\; (\mu, \rho, \mu', \rho') \leftarrow\!\!\$\, \mathsf{A}(1^\lambda)\right].$$

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

Game $\mathbf{SKE}^{\mathsf{ind\text{-}cca}}_{\Sigma,\mathsf{A}}(\lambda, b)$:

$\kappa \leftarrow\!\!\$\; \mathcal{K}$

$(\mu_0, \mu_1, \alpha) \leftarrow\!\!\$\; \mathsf{A}_1^{\mathcal{O}_{\mathsf{enc}}(\kappa,\cdot),\mathcal{O}_{\mathsf{dec}}(\kappa,\cdot)}(1^\lambda)$

$\hat{\gamma} \leftarrow\!\!\$\; \mathsf{Enc}(\kappa, \mu_b)$

Return $\mathsf{A}_2^{\mathcal{O}_{\mathsf{enc}}(\kappa,\cdot),\mathcal{O}_{\mathsf{dec}}(\kappa,\cdot)}(\alpha, \hat{\gamma})$

Oracle $\mathcal{O}_{\mathsf{enc}}(\kappa, \mu)$:

Return $\mathsf{Enc}(\kappa, \mu)$

Oracle $\mathcal{O}_{\mathsf{dec}}(\kappa, \gamma)$:

If $\gamma = \hat{\gamma}$

  Return $\perp$

Else

  Return $\mathsf{Dec}(\kappa, \gamma)$

Figure 1: Experiment defining security of SKE.

Definition 2 (Hiding)**.** We say that a non-interactive commitment scheme Com is *computationally hiding* if for all messages $\mu_0, \mu_1 \in \mathcal{M}$ it holds that

$$\left\{\mathsf{Com}(1^\lambda, \mu_0)\right\}_{\lambda\in\mathbb{N}} \overset{\mathsf{c}}{\approx} \left\{\mathsf{Com}(1^\lambda, \mu_1)\right\}_{\lambda\in\mathbb{N}}.$$

In case the above ensembles are *statistically close* (resp. identically distributed), we say that Com is *statistically* (resp. *perfectly*) hiding.

2.2 Symmetric Encryption

A secret-key encryption (SKE) scheme is a tuple $\Sigma = (\mathsf{Enc}, \mathsf{Dec})$ of polynomial-time algorithms specified as follows.

- Enc is a randomized algorithm that takes as input a key $\kappa \in \mathcal{K}$ and a message $\mu \in \mathcal{M}$ and outputs a ciphertext $\gamma \in \mathcal{C}$, where $\mathcal{M}$ and $\mathcal{C}$ are the *message space* and the *ciphertext space* respectively.
- Dec is a deterministic algorithm that takes as input the key $\kappa \in \mathcal{K}$ and a ciphertext $\gamma \in \mathcal{C}$ and outputs a message $\mu \in \mathcal{M} \cup \{\perp\}$, where $\perp$ denotes an invalid ciphertext.

We say that Σ satisfies correctness if, for all $\kappa \in \mathcal{K}$ and all messages $\mu \in \mathcal{M}$, we have that $\mathsf{Dec}(\kappa, \mathsf{Enc}(\kappa, \mu)) = \mu$ with probability 1 over the randomness of Enc. As for security, we will need SKE schemes satisfying the standard notion of indistinguishability against chosen-ciphertext attacks (IND-CCA). Informally, this property states that it is hard to distinguish the encryption of any two messages even if the adversary has encryption/decryption capabilities under the target key. Formally, we have the following definition.

Definition 3 (Security of SKE)**.** We say that $\Sigma = (\mathsf{Enc}, \mathsf{Dec})$ is an IND-CCA secure SKE scheme if the following holds for the experiment in Fig. 1: For all PPT adversaries A,

$$\left\{\mathbf{SKE}^{\mathsf{ind\text{-}cca}}_{\Sigma,\mathsf{A}}(\lambda, 0)\right\}_{\lambda\in\mathbb{N}} \overset{\mathsf{c}}{\approx} \left\{\mathbf{SKE}^{\mathsf{ind\text{-}cca}}_{\Sigma,\mathsf{A}}(\lambda, 1)\right\}_{\lambda\in\mathbb{N}}.$$

2.3 Information Dispersal

Information dispersals are similar to secret sharing schemes but they do not guarantee privacy. Formally, let $n, t \in \mathbb{N}$, with $t \leq n$. A t-out-of-n information dispersal is a pair of (deterministic) polynomial-time algorithms $(\mathsf{IDisp}, \mathsf{IRec})$ defined as follows:

- IDisp takes as input a message $\mu \in \mathcal{M}$ and outputs n shares $\mu_1, \ldots, \mu_n$, where each $\mu_i \in \mathcal{M}_i$.
- IRec takes as input a certain subset of shares and outputs a value in $\mathcal{M} \cup \{\perp\}$.

We require the following correctness property: For all $\mu \in \mathcal{M}$ and all $\mathcal{I}$ with $|\mathcal{I}| \geq t$, it holds that $\mathsf{IRec}((\mathsf{IDisp}(\mu))_{\mathcal{I}}) = \mu$.

3 Secret Sharing Schemes

A n-party secret sharing scheme Π, with *message space* $\mathcal{M}$ and *share space* $\mathcal{S} = \mathcal{S}_1 \times \ldots \times \mathcal{S}_n$, consists of polynomial-time algorithms $(\mathsf{Share}, \mathsf{Rec})$ specified as follows:

- Share is a randomized algorithm that takes as input a message $\mu \in \mathcal{M}$ and outputs n shares $\sigma_1, \ldots, \sigma_n$, with $\sigma_i \in \mathcal{S}_i$.
- Rec is a deterministic algorithm that takes as input a certain subset of shares and outputs a value in $\mathcal{M} \cup \{\perp\}$.

The subset of parties allowed to reconstruct the secret by pulling their shares together form the so-called *access structure*. We consider the t-out-of-n access structure where any subset of shares of cardinality bigger or equal to t can reconstruct (we call such subsets *authorized*), while any subset of shares of cardinality less than t cannot (we call such subsets *unauthorized*). Subsets of cardinality exactly t are called *minimal authorized* sets.

Definition 4 (Threshold secret sharing scheme)**.** Let $n, t \in \mathbb{N}$ and $t \leq n$, we say that $\Pi = (\mathsf{Share}, \mathsf{Rec})$ is a t-out-of-n *secret sharing* scheme if the following two properties are satisfied.

- **Correctness:** for all $\lambda \in \mathbb{N}$, all $\mu \in \mathcal{M}$ and all $\mathcal{I}$ with $|\mathcal{I}| \geq t$, we have that $\mathsf{Rec}((\mathsf{Share}(1^\lambda, \mu))_{\mathcal{I}}) = \mu$ with overwhelming probability over the randomness of the sharing algorithm.
- **Perfect Privacy:** for all pairs of messages $\mu_0, \mu_1 \in \mathcal{M}$ and for any $\mathcal{U}$ with $|\mathcal{U}| < t$, we have that
$$\{(\mathsf{Share}(1^\lambda, \mu_0))_{\mathcal{U}}\}_{\lambda \in \mathbb{N}} \equiv \{(\mathsf{Share}(1^\lambda, \mu_1))_{\mathcal{U}}\}_{\lambda \in \mathbb{N}},$$
If the above ensembles are statistically (resp. computationally) close, we speak of *statistical* (resp. *computational*) privacy.

Finally, when considering the length of the shares, we define the *information rate* of a secret sharing scheme as the ratio between the length of the secret message and the maximal length of a share.

Definition 5 (Asymptotic rate)**.** Let Π be an n-party secret sharing scheme with message space $\mathcal{M} = \{0,1\}^*$ and share space $\mathcal{S}_1 \times \ldots \times \mathcal{S}_n$. Let $s(|\mu|, \lambda) = \max_{i \in [n]} \log |\mathcal{S}_i(|\mu|, \lambda)|$ where $\mathcal{S}_1(|\mu|, \lambda) \times \ldots \times \mathcal{S}_n(|\mu|, \lambda)$ is the share space for messages μ of length $|\mu|$ and security parameter λ. The asymptotic information rate of Π is defined to be

$$\varrho = \inf_{\lambda \in \mathbb{N}} \lim_{|\mu| \to \infty} \frac{|\mu|}{s(|\mu|, \lambda)}.$$

3.1 Tampering and Leakage Model

In our model the attacker partitions all of the shares into m (non-overlapping) blocks with size at most k, covering the entire set $[n]$. This is formalized through the notion of a k-sized partition.

Definition 6 (k-sized partition)**.** Let $k, m \in \mathbb{N}$. We call $\mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_m)$ a k-sized partition of $[n]$ if:

(i) $\bigcup_{i \in [m]} \mathcal{B}_i = [n]$;

(ii) $\forall i_1, i_2 \in [m]$ such that $i_1 \neq i_2$, $\mathcal{B}_{i_1} \cap \mathcal{B}_{i_2} = \emptyset$;

(iii) $\forall i \in [m], |\mathcal{B}_i| \leq k$.

Fix $\mu \in \mathcal{M}$ and let $\mathcal{B}$ be a k-sized partition of $[n]$. To define our security model, we consider an adversary A interacting with a target secret sharing $\sigma = (\sigma_1, \ldots, \sigma_n)$ of μ as follows:

$\mathbf{LR\text{-}CNMSS}^{\mu_0,\mu_1}_{\Pi,\mathsf{A},\mathcal{B}}(\lambda, b)$:

$\sigma \leftarrow\$ \mathsf{Share}(\mu_b)$
$\texttt{stop} \leftarrow \texttt{false}$
Return $\mathsf{A}^{\mathcal{O}_{\mathsf{leak}}(\sigma,\cdot),\mathcal{O}^{\mu_0,\mu_1}_{\mathsf{tamp}}(\sigma,\cdot,\cdot)}$

Oracle $\mathcal{O}_{\mathsf{leak}}(\sigma, (g_1, \ldots, g_m))$:

If $\texttt{stop} = \texttt{true}$
  Return $\perp$
Else
  $\forall i \in [m]$, $\Lambda_i = g_i(\sigma_{\mathcal{B}_i})$
  Return $\Lambda = \Lambda_1 || \ldots || \Lambda_m$

Oracle $\mathcal{O}_{\mathsf{tamp}}(\sigma, \mathcal{T}, (f_1, \ldots, f_m))$:

If $\texttt{stop} = \texttt{true}$
  Return $\perp$
Else
  $\forall i \in [m]$, $\tilde{\sigma}_{\mathcal{B}_i} = f_i(\sigma_{\mathcal{B}_i})$
  $\tilde{\mu} = \mathsf{Rec}(\tilde{\sigma}_{\mathcal{T}})$
  If $\tilde{\mu} \in \{\mu_0, \mu_1\}$
    Return $\diamond$
  If $\tilde{\mu} = \perp$
    Return $\perp$, and $\texttt{stop} \leftarrow \texttt{true}$
  Else
    Return $\tilde{\mu}$

Figure 2: Experiment defining leakage-resilient continuously non-malleable secret sharing against joint tampering. The tampering and leakage oracles are implicitly parameterized by set $\mathcal{B}$, messages μ_0, μ_1 and flag $\texttt{stop}$.

- **Leakage queries.** For each $i \in [m]$, the attacker can leak jointly from the shares $\sigma_{\mathcal{B}_i}$. This can be done repeatedly and in an adaptive fashion, so long as the leakage does not decrease the min-entropy of the shares by too much. Formally, for any $\mu \in \mathcal{M}$, for each $i \in [m]$ and for $\ell \geq 0$, we require that
$$\widetilde{\mathbb{H}}_\infty((\mathbf{\Sigma}_j)_{j \in \mathcal{B}_i} \mid \mathbf{\Lambda}_i) \geq \widetilde{\mathbb{H}}_\infty((\mathbf{\Sigma}_j)_{j \in \mathcal{B}_i}) - \ell, \tag{1}$$
where $(\mathbf{\Sigma}_1, \ldots, \mathbf{\Sigma}_n)$ is the r.v. corresponding to $\mathsf{Share}(\mu)$, and $\mathbf{\Lambda}_i$ is the r.v. corresponding to the total leakage performed within $\mathcal{B}_i$ (over all queries). An adversary obeying this restriction is called *ℓ-leakage admissible.*
- **Tampering queries.** For each $i \in [m]$, the attacker can tamper jointly with the shares $\sigma_{\mathcal{B}_i}$. Each such query yields tampered shares $\tilde{\sigma}_1, \ldots, \tilde{\sigma}_n$, for which the adversary is allowed to choose a *different* reconstruction set $\mathcal{T} \subseteq [n]$, with $|\mathcal{T}| \geq t$, and see the corresponding reconstructed message. This can be done repeatedly and in an adaptive fashion, the only restriction being that after the first tampering query yielding an invalid message, the answer to all future tampering (and leakage) queries is automatically set to $\perp$ (the so-called *self-destruct* feature). This restriction is well-known to be necessary when an arbitrary polynomial number of tampering queries is allowed [FV19, SV19].

Now we are ready to give the formal notion of security. Intuitively, leakage-resilient continuous non-malleability states that, given two[9] messages $\mu_0, \mu_1 \in \mathcal{M}$, no admissible adversary, as defined above, can distinguish whether it is interacting with a secret sharing of μ_0 or of μ_1.

Definition 7 (Leakage-resilient continuous non-malleability [BFV19])**.** Let $n, t, k \in \mathbb{N}$ and $\ell \geq 0$ be parameters. A t-out-of-n secret sharing scheme Π is ℓ-noisy-leakage-resilient continuously non-malleable under selective k-joint leakage and tampering attacks ((k, ℓ)-LR-CNMSS, for short), if for all messages $\mu_0, \mu_1 \in \mathcal{M}$, all k-sized partitions of $[n]$, and all PPT ℓ-leakage admissible attackers A, the following holds for the experiment of Fig. 2:
$$\left\{\mathbf{LR\text{-}CNMSS}^{\mu_0,\mu_1}_{\Pi,\mathsf{A},\mathcal{B}}(\lambda, 0)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathbf{LR\text{-}CNMSS}^{\mu_0,\mu_1}_{\Pi,\mathsf{A},\mathcal{B}}(\lambda, 1)\right\}_{\lambda\in\mathbb{N}}. \tag{2}$$

[9] Goyal and Kumar [GK18a] originally gave a simulation-based definition of non-malleability (for the case of one-time tampering). It is folklore that this flavor of non-malleability can be shown to be equivalent to the indistinguishability-based notion we define (even in the setting of continuous tampering), so long as the message length is super-logarithmic in the security parameter.

When no leakage is allowed (i.e. $\ell = 0$), we simply say that Π is a k-CNMSS.

3.2 Related Notions

By adapting the above definition, we obtain several notions as special cases, as detailed below.

Leakage-resilient one-time non-malleability. Let $\mathbf{LR\text{-}NMSS}^{\mu_0,\mu_1}_{\Pi,\mathsf{A},\mathcal{B}}(\lambda, b)$ be the experiment that is defined identically to $\mathbf{LR\text{-}CNMSS}^{\mu_0,\mu_1}_{\Pi,\mathsf{A},\mathcal{B}}(\lambda, b)$, except that the attacker is allowed a *single* tampering query and all the leakage happens before such query. In this case, Definition 7 can be achieved information theoretically (*i.e.*, for all unbounded adversaries). In particular, Eq. (2) now becomes

$$\Delta\left(\mathbf{LR\text{-}NMSS}^{\mu_0,\mu_1}_{\Pi,\mathsf{A},\mathcal{B}}(\lambda, 0); \mathbf{LR\text{-}NMSS}^{\mu_0,\mu_1}_{\Pi,\mathsf{A},\mathcal{B}}(\lambda, 1)\right) \leq \epsilon \tag{3}$$

for some $\epsilon \in [0, 1)$, and we speak of ℓ-noisy-leakage-resilient one-time statistically ϵ-non-malleable secret sharing under selective k-joint leakage and tampering attacks ((k, ℓ, ϵ)-LR-NMSS, for short).

Asymmetric p-time non-malleable codes. When the number of parties is $n = 2$, *i.e.* in case Π is a 2-out-of-2 secret sharing, we obtain the notion of leakage-resilient split-state continuously non-malleable codes [FMNV14, OPVV18] as a special case. Here, it will be useful to consider asymmetric shares and possibly to tolerate different amounts of leakage from each side; towards this, when we explicitly need the size of the shares, we speak of $(s_{\mathsf{L}}, s_{\mathsf{R}})$-asymmetric codes, where $s_{\mathsf{L}} = \log|\mathcal{S}_{\mathsf{L}}|$ is the size of the left share and $s_{\mathsf{R}} = \log|\mathcal{S}_{\mathsf{R}}|$ is the size of the right share. Moreover, it will suffice for us to consider an attack scenario where the adversary performs all the leakage *before* tampering, and can only send p tampering queries (where p is fixed *a priori*). Notice that, when the number of tampering queries is bounded, then we can obtain security even without assuming self-destruct (*i.e.*, the self-destruct flag `stop` is never set to true in the experiment of Fig. 2).

An adversary A is called $(\ell_{\mathsf{L}}, \ell_{\mathsf{R}})$-leakage and p-time tampering admissible, so long as it makes at most p tampering queries and performs all leakage queries before the first tampering query, with the restriction that:

$$\widetilde{\mathbb{H}}_\infty(\mathbf{\Sigma}_{\mathsf{L}} \mid \mathbf{\Lambda}_{\mathsf{L}}) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{\Sigma}_{\mathsf{L}}) - \ell_{\mathsf{L}},$$
$$\widetilde{\mathbb{H}}_\infty(\mathbf{\Sigma}_{\mathsf{R}} \mid \mathbf{\Lambda}_{\mathsf{R}}) \geq \widetilde{\mathbb{H}}_\infty(\mathbf{\Sigma}_{\mathsf{R}}) - \ell_{\mathsf{R}},$$

where $\mathbf{\Sigma} = (\mathbf{\Sigma}_{\mathsf{L}}, \mathbf{\Sigma}_{\mathsf{R}})$ is the r.v. corresponding ot the target secret sharing, and $\mathbf{\Lambda}_{\mathsf{L}}, \mathbf{\Lambda}_{\mathsf{R}}$ are the r.v. corresponding to the total leakage performed on $\mathbf{\Sigma}_{\mathsf{L}}, \mathbf{\Sigma}_{\mathsf{R}}$ (over all queries). In this case, we say that Π is an asymmetric $(\ell_{\mathsf{L}}, \ell_{\mathsf{R}})$-leakage-resilient p-time ϵ-non-malleable split-state code (asymmetric $(\ell_{\mathsf{L}}, \ell_{\mathsf{R}}, p, \epsilon)$-LR-NMC, for short). We denote the corresponding experiment as $\mathbf{LR\text{-}NMC}^{\mu_0,\mu_1}_{\Pi,\mathsf{A}}(\lambda, b)$.

When no leakage is allowed (*i.e.*, $\ell_{\mathsf{L}}, \ell_{\mathsf{R}} = 0$), we simply speak of asymmetric p-time ϵ-non-malleable split-state codes (asymmetric (p, ϵ)-NMC for short); similarly, when no tampering is allowed (*i.e.* $p = 0$), we speak of asymmetric $(\ell_{\mathsf{L}}, \ell_{\mathsf{R}}, \epsilon)$-leakage-resilient split-state code (asymmetric $(\ell_{\mathsf{L}}, \ell_{\mathsf{R}})$-LRC for short). Finally, in the latter case, we also need the existence of an efficiently computable algorithm $\overline{\mathsf{Share}}$ such that, for all $\sigma_{\mathsf{R}} \in \mathcal{S}_{\mathsf{R}}$ and $\mu \in \mathcal{M}$, it holds that $\mathsf{Rec}(\overline{\mathsf{Share}}(\mu, \sigma_{\mathsf{R}}), \sigma_{\mathsf{R}}) = \mu$ and moreover the distributions of the left shares sampled from $\overline{\mathsf{Share}}$ is equivalent to the distribution of the left shares of Share conditioned on the right share being σ_{R} and the message being μ. In other words, given as input the message μ and a right share

σ_{R}, the algorithm $\overline{\mathsf{Share}}$ produces a left share σ_{L} such that $(\sigma_{\mathsf{L}}, \sigma_{\mathsf{R}})$ is a valid and properly distributed encoding of μ. We refer the reader to §6 for concrete examples of leakage-resilient split-state codes meeting the above property.

4 Rate-Zero Continuously Non-Malleable Secret Sharing

In this section, we give a construction of a leakage-resilient continuously non-malleable secret sharing scheme against selective joint tampering. We refer the reader to §1.3 for an overview of our scheme (and its security) and here directly provide a formal treatment.

Let $\Pi = (\mathsf{Share}, \mathsf{Rec})$ be the t-out-of-n Shamir's secret sharing scheme. For simplicity we assume that Π can support messages of variable length, namely the sharing procedure chooses a field that is large enough to encode the input message μ for n parties (for simplicity, we assume that $|\mu| \geq \log n$), and we denote such field as $\mathbb{F}(|\mu|)$, or simply $\mathbb{F}$ when the message is clear from the context. A share σ_i of Π is a tuple (i, x) where $i \in [n]$ and $x \in \mathbb{F}$ is a field element; in particular, if p is the polynomial chosen by the Share algorithm, for all $i \in [n]$, $\sigma_i = (i, p(i))$. Let $\mathcal{S}_i(|\mu|) := \{(i, x) : x \in \mathbb{F}(|\mu|)\}$, clearly a secret sharing of μ has support $\mathcal{S}_1(|\mu|) \times \cdots \times \mathcal{S}_n(|\mu|)$. Consider the function idx that, upon input a tuple $\sigma = (i^*, x)$, outputs the first component $\mathsf{idx}(\sigma) = i^*$; in particular, for a share σ_i generated by the sharing function Share, it holds that $\mathsf{idx}(\sigma_i) = i$. Finally, let $\Pi' = (\mathsf{Share}', \mathsf{Rec}')$ be a split-state code with codeword space $\mathcal{S}_{\mathsf{L}} \times \mathcal{S}_{\mathsf{R}}$ and Com be a non-interactive commitment scheme. Consider the following derived scheme $\Pi^* = (\mathsf{Share}^*, \mathsf{Rec}^*)$.

- **Algorithm** Share^***:** upon input μ, first sample randomness $\rho \leftarrow\!\!\$\, \mathcal{R}$ and compute $\gamma \leftarrow \mathsf{Com}(\mu; \rho)$ and $(\sigma_1, \ldots, \sigma_n) \leftarrow\!\!\$\, \mathsf{Share}(\mu || \rho)$. Then, for each $i \in [n]$, compute $(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \leftarrow\!\!\$\, \mathsf{Share}'(\sigma_i)$ and $(\sigma^{(1)}_{\mathsf{R},i}, \ldots, \sigma^{(n)}_{\mathsf{R},i}) \leftarrow\!\!\$\, \mathsf{Share}(\sigma_{\mathsf{R},i})$. Finally, set $\sigma^*_i = (\gamma, \sigma_{\mathsf{L},i}, (\sigma^{(i)}_{\mathsf{R},j})_{j \in [n]})$ for all $i \in [n]$ and output $(\sigma^*_1, \ldots, \sigma^*_n)$.
- **Algorithm** Rec^***:** upon input shares $(\sigma^*_i)_{i \in \mathcal{I}}$, parse $\sigma_i = (\gamma_i, \sigma_{\mathsf{L},i}, (\sigma^{(i)}_{\mathsf{R},j})_{j \in [n]})$ for all $i \in \mathcal{I}$. Check that all the commitments $(\gamma_i)_{i \in \mathcal{I}}$ are the same; if not output $\perp$, and else let γ be the commitment contained in each share. Compute $\sigma_{\mathsf{R},i} = \mathsf{Rec}((\sigma^{(j)}_{\mathsf{R},i})_{j \in \mathcal{I}})$ and $\sigma_i = \mathsf{Rec}'(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$; check that there exist no distinct $i_1, i_2 \in \mathcal{I}$ such that $\mathsf{idx}(\sigma_{i_1}) = \mathsf{idx}(\sigma_{i_2})$ (and output $\perp$ otherwise). Finally, reconstruct $\mu || \rho = \mathsf{Rec}((\sigma_i)_{i \in \mathcal{I}})$ and output μ if $\gamma = \mathsf{Com}(\mu; \rho)$ and $\perp$ otherwise.

We are now ready to state the following theorem.

Theorem 1. *Let $n, t \in \mathbb{N}$, with $t > 2n/3$, and $\ell_{\mathsf{L}}, \ell_{\mathsf{R}} \geq 0$. Assume that Com is a perfectly binding and computationally hiding commitment and Π' is an asymmetric $(\ell_{\mathsf{L}}, \ell_{\mathsf{R}}, \mathsf{negl}(\lambda), t)$-LR-NMC satisfying the following properties:*

(i) There exists $\sigma^*_{\mathsf{L}} \in \mathcal{S}_{\mathsf{L}}$ such that, for any μ, there exists $\sigma_{\mathsf{R}} \in \mathcal{S}_{\mathsf{R}}$ such that $\mathsf{Rec}'(\sigma^*_{\mathsf{L}}, \sigma_{\mathsf{R}}) = \mu$.

(ii) There exists $d \geq 0$ such that, for any μ, it holds that $\widetilde{\mathbb{H}}_\infty(\mathbf{\Sigma}_{\mathsf{L}} \,|\, \mathbf{\Sigma}_{\mathsf{R}}) \geq \mathbb{H}_\infty(\mathbf{\Sigma}_{\mathsf{L}}) - d$ and $\widetilde{\mathbb{H}}_\infty(\mathbf{\Sigma}_{\mathsf{R}} \,|\, \mathbf{\Sigma}_{\mathsf{L}}) \geq \mathbb{H}_\infty(\mathbf{\Sigma}_{\mathsf{R}}) - d$, where $(\mathbf{\Sigma}_{\mathsf{L}}, \mathbf{\Sigma}_{\mathsf{R}})$ is the r.v. corresponding to $\mathsf{Share}'(\mu)$.

Then, the above secret sharing scheme Π^* is a t-out-of-n $(t-1, \ell^*)$-LR-CNMSS so long as:

$$\ell_{\mathsf{R}} \geq (t-1) \cdot \ell^* + |\mu| + |\gamma| + d + 1 + \log(\lambda)$$
$$\ell_{\mathsf{L}} \geq \ell^* + n \cdot (t-1) \cdot s_{\mathsf{R}} + |\gamma| + d + 1 + \log(\lambda),$$

where $|\mu| \in \mathbb{N}$ is the length of the message, $|\gamma|$ is the length of a commitment and $s_{\mathsf{R}} = \log |\mathcal{S}_{\mathsf{R}}|$ is the size of a right share under Π'.

The privacy property of Π^* follows readily by privacy of Π and the computational hiding property of Com. In what follows, we focus on the proof of leakage-resilient continuous non-malleability. Wlog., we are going to assume that each reconstruction set $\mathcal{T}$ queried by the

$\mathbf{LR\text{-}CNMSS}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, b)$ $\boxed{\mathbf{Hyb}^{\mu_0,\mu_1}_r(\lambda, b)}$:

$\rho \leftarrow\!\!\$\ \mathcal{R}$
$\gamma := \mathsf{Com}(\mu_b; \rho)$
$(\sigma_1, \ldots, \sigma_n) \leftarrow\!\!\$\ \mathsf{Share}(\mu_b || \rho)$
$\boxed{(\sigma'_1, \ldots, \sigma'_n) \leftarrow\!\!\$\ \times_{i\in[n]} \mathcal{S}_i(|\mu_b| + |\rho|)}$
$\boxed{\forall i > r, \sigma'_i := \sigma_i}$
$\forall i \in [n]$:
  $(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \leftarrow\!\!\$\ \mathsf{Share}'(\sigma_i)$
  $\boxed{(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \leftarrow\!\!\$\ \mathsf{Share}'(\sigma'_i)}$
  $(\sigma^{(1)}_{\mathsf{R},i}, \ldots, \sigma^{(n)}_{\mathsf{R},i}) \leftarrow\!\!\$\ \mathsf{Share}(\sigma_{\mathsf{R},i})$
  $\sigma^*_i := (\gamma, \sigma_{\mathsf{L},i}, (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]})$
$\sigma^* := (\sigma^*_1, \ldots, \sigma^*_n)$
$\mathtt{stop} \leftarrow \mathtt{false}$
Return $\mathsf{A}^{\mathcal{O}_{\mathsf{tamp}}(\sigma^*,\cdot),\mathcal{O}_{\mathsf{leak}}(\sigma^*,\cdot)}(1^\lambda)$

Algorithm $\mathsf{Split}((\sigma^*_i)_{i\in\mathcal{T}})$:

$\sigma_{\mathsf{R},i} = \mathsf{Rec}((\sigma^{(j)}_{\mathsf{R},i})_{j\in\mathcal{T}})$
$\sigma_i = \mathsf{Rec}'(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$
Output $(\sigma_i)_{i\in\mathcal{T}}$

Oracle $\mathcal{O}_{\mathsf{tamp}}(\sigma^*, \mathcal{T}, (f_1, \ldots, f_m))$:

If $\mathtt{stop} = \mathtt{true}$, return $\perp$
$\forall i \in [m] : \tilde{\sigma}^*_{\mathcal{B}_i} := f_i(\sigma^*_{\mathcal{B}_i})$
$\tilde{\sigma}^* = (\sigma^*_1, \ldots, \sigma^*_n)$
$\forall i \in \mathcal{T}, \tilde{\sigma}^*_i = (\tilde{\gamma}_i, \tilde{\sigma}_{\mathsf{L},i}, (\tilde{\sigma}^{(i)}_{\mathsf{R},j})_{j\in[n]})$
If $\exists i_1, i_2 \in \mathcal{T} : \tilde{\gamma}_{i_1} \neq \tilde{\gamma}_{i_2}$
  $\mathtt{stop} \leftarrow \mathtt{true}$ and return $\perp$
Else, let $\tilde{\gamma} := \tilde{\gamma}_i$
$(\tilde{\sigma}_i)_{i\in\mathcal{T}} = \mathsf{Split}((\tilde{\sigma}^*_i)_{i\in\mathcal{T}})$
If $\exists i_1, i_2 \in \mathcal{T} : \mathsf{idx}(\tilde{\sigma}_{i_1}) = \mathsf{idx}(\tilde{\sigma}_{i_2})$
  $\mathtt{stop} \leftarrow \mathtt{true}$ and return $\perp$
$\boxed{\forall i_1, i_2 \in \mathcal{T} : \tilde{\sigma}_{i_1} = \sigma'_{i_2}}$
  $\boxed{\text{Let } \tilde{\sigma}_{i_1} := \sigma_{i_2}}$
$\tilde{\mu} || \tilde{\rho} = \mathsf{Rec}((\tilde{\sigma}_i)_{i\in\mathcal{T}})$
If $\tilde{\gamma} \neq \mathsf{Com}(\tilde{\mu}; \tilde{\rho})$,
  $\mathtt{stop} \leftarrow \mathtt{true}$ and return $\perp$
If $\tilde{\mu} \in \{\mu_0, \mu_1\}$, return $\diamond$
Return $\tilde{\mu}$

Oracle $\mathcal{O}_{\mathsf{leak}}(\sigma^*, (g_1, \ldots, g_m))$:

If $\mathtt{stop} = \mathtt{true}$, return $\perp$
Else, return $g_1(\sigma^*_{\mathcal{B}_1}), \ldots, g_m(\sigma^*_{\mathcal{B}_m})$

Figure 3: Experiments in the proof of Theorem 1. The instructions boxed in red are the modifications introduced by the hybrid experiment. For compactness, we denote by Split the algorithm that reconstructs the shares σ_i from the shares $(\sigma_{\mathsf{L},i}, (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]})$.

adversary is minimal.[10] Furthermore, we will make the simplifying assumption that the partition $\mathcal{B}$ fixed by the attacker only contains two subsets, i.e. $\mathcal{B}_1$ and $\mathcal{B}_2$. Note that this restriction is wlog. whenever $t > 2n/3$: in fact, for any partition $\mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_m)$, it is always possible to find a set of indices $\mathcal{I} \subseteq [m]$ such that $t/2 \leq |\bigcup_{i\in\mathcal{I}} \mathcal{B}_i| < t$ and then consider the two subsets to be $\hat{\mathcal{B}}_1 = \bigcup_{i\in\mathcal{I}} \mathcal{B}_i$ (which contains less than t elements by construction) and $\hat{\mathcal{B}}_2 = \bigcup_{i\notin\mathcal{I}} \mathcal{B}_i$ (which contains $n - |\bigcup_{i\in\mathcal{I}} \mathcal{B}_i| < 3t/2 - t/2 = t$ elements).

Remark 1. *Note that if we restrict the adversary to only choose partitions of two subsets, i.e.* $\mathcal{B}_1, \mathcal{B}_2 \subseteq [n]$ *s.t.* $\mathcal{B}_1 \cap \mathcal{B}_2 = \emptyset$ *and* $\mathcal{B}_1 \cup \mathcal{B}_2 = [n]$*, then it only suffices to require* $t \geq n/2+1$*. This is because we can put* $\hat{\mathcal{B}}_1 := \mathcal{B}_1$ *and* $\hat{\mathcal{B}}_2 := \mathcal{B}_2$*, while the restriction on* t *comes from the fact that both* $\mathcal{B}_1$ *and* $\mathcal{B}_2$ *must be unauthorized, i.e.* $|\mathcal{B}_1|, |\mathcal{B}_2| \leq t-1$*, and therefore* $n = |\mathcal{B}_1| + |\mathcal{B}_2| \leq 2t-2$*, that is,* $t \geq n/2+1$*.*

For $r \in [n]$, consider the auxiliary hybrid experiments $\mathbf{Hyb}_r(\lambda, b)$ described in Fig. 3 along with the original experiment in order to highlight the main differences. In particular, in $\mathbf{Hyb}_r(\lambda, b)$, we replace the first r shares $(\sigma_1, \ldots, \sigma_r)$ from the first application of Π with random and independent values $(\sigma'_1, \ldots, \sigma'_r)$, letting the remaining shares $\sigma'_{r+1}, \ldots, \sigma'_n$ the same

[10] It is always possible to modify the reconstruction algorithm Rec so that, upon input more than t shares, say $\sigma_{i_1}, \ldots, \sigma_{i_t}, \ldots$, with $i_1 < i_2 < \ldots < i_t < \ldots$, it only considers the first t shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$ in order to reconstruct the message.

as the original experiment. Note that, when $r = 0$, we do not replace any share, hence, for all $b \in \{0,1\}$, $\mathbf{Hyb}_0(\lambda, b) \equiv \mathbf{LR\text{-}CNMSS}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, b)$. For all $r \in [n]$, we will prove by induction over the number of tampering queries that the experiments $\mathbf{Hyb}_{r-1}(\lambda, b)$ and $\mathbf{Hyb}_r(\lambda, b)$ are statistically close. Towards this, for all $r \in [n] \cup \{0\}$, let us denote by $\mathbf{Hyb}_r(\lambda, b, p)$ the experiment $\mathbf{Hyb}_r(\lambda, b)$ where the adversary A is limited to ask exactly p tampering queries.

4.1 Induction Basis

The lemma below constitutes the basis of the induction.

Lemma 2. *For all $b \in \{0,1\}$, and all $r \in [n]$, it holds that*

$$\left\{\mathbf{Hyb}_{r-1}(\lambda, b, 1)\right\}_{\lambda\in\mathbb{N}} \overset{\mathsf{s}}{\approx} \left\{\mathbf{Hyb}_r(\lambda, b, 1)\right\}_{\lambda\in\mathbb{N}}.$$

Proof. The difference between the two hybrids is that in $\mathbf{Hyb}_r(\lambda, b, 1)$ the share σ'_r is uniformly random, whereas in $\mathbf{Hyb}_{r-1}(\lambda, b, 1)$ the share σ'_r is set to be σ_r (as defined in the original experiment). For any $j \in [n]$, let $\xi(j) \in \{1,2\}$ be the index such that $j \in \mathcal{B}_{\xi(j)}$. The proof proceeds by reduction to leakage-resilient t-time non-malleability of Π'. In more detail, for a fixed choice of $b \in \{0,1\}$ and $r \in [n]$, let A be an adversary telling apart the two hybrids with probability at least $1/\mathsf{poly}(\lambda)$. Consider the following (possibly inefficient) adversary A' attacking Π'.

1. **Setup.** Set the challenge messages to be σ_r and σ'_r sampled as in $\mathbf{Hyb}_r(\lambda, b, 1)$, and let γ be the commitment corresponding to the message μ_b.
2. **Shared randomness.** For every $i \in [n] \setminus \{r\}$, sample $\sigma_{\mathsf{L},i}, (\sigma^{(j)}_{\mathsf{R},i})_{j\in[n]}$ according to $\mathbf{Hyb}_r(\lambda, b)$. Then, let $i_\mathsf{L} = \xi(r)$ and $i_\mathsf{R} = 3 - i_\mathsf{L} \in \{1,2\}$. Let $\mathcal{J}$ be any set such that $\mathcal{B}_{i_\mathsf{L}} \subseteq \mathcal{J} \subseteq [n]$ and $|\mathcal{J}| = t-1$. For all $j \in \mathcal{J}$, sample the shares $\sigma^{(j)}_{\mathsf{R},r}$ uniformly at random. Finally, sample the left share σ^*_L given by property (i) of Π'.
   After this step, the reduction A' knows σ^*_L and the following values:
$$\forall i \in [n] \setminus \mathcal{J}: \qquad \gamma, \qquad \sigma_{\mathsf{L},i}, \qquad (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]\setminus\{r\}} \tag{4}$$
$$\forall i \in \mathcal{J} \setminus \{r\}: \qquad \gamma, \qquad \sigma_{\mathsf{L},i}, \qquad (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]} \tag{5}$$
$$\text{For } i = r: \qquad \gamma, \qquad (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]} \tag{6}$$
3. **Leakage queries.** Upon receiving a leakage query $g = (g_1, g_2)$ from A, construct the following leakage functions.
   (a) Let g_L be the leakage function which takes as input the value $\sigma_{\mathsf{L},r}$, plugs it in Eq. (6) and appends the values of Eq. (5) to obtain $(\sigma^*_i)_{i\in\mathcal{B}_{i_\mathsf{L}}}$ (recall that $\mathcal{B}_{i_\mathsf{L}} \subseteq \mathcal{J}$), and finally outputs $\Lambda_{i_\mathsf{L}} = g_{i_\mathsf{L}}((\sigma^*_i)_{i\in\mathcal{B}_{i_\mathsf{L}}})$.
   (b) Let g_R be the leakage function which takes as input the value $\sigma_{\mathsf{R},r}$, computes the values $(\sigma^{(i)}_{\mathsf{R},r})_{i\in[n]\setminus\mathcal{J}}$ using $\sigma_{\mathsf{R},r}$ and the values $(\sigma^{(i)}_{\mathsf{R},r})_{i\in\mathcal{J}}$ and plugs them in Eq. (4) in order to obtain $(\sigma^*_i)_{i\in[n]\setminus\mathcal{J}}$; then, appends the values of Eq. (5) to obtain $(\sigma^*_i)_{i\in\mathcal{B}_{i_\mathsf{R}}}$, and finally outputs $\Lambda_{i_\mathsf{R}} = g_{i_\mathsf{R}}((\sigma^*_i)_{i\in\mathcal{B}_{i_\mathsf{R}}})$.

   Send $(g_\mathsf{L}, g_\mathsf{R})$ to the leakage oracle and forward the answer $\Lambda_1||\Lambda_2$ to A.
4. **Tampering query.** Upon receiving the tampering query $(\mathcal{T}, f = (f_1, f_2))$ from A, construct the following leakage and tampering functions.
   (a) Let $\hat{g}_\mathsf{L}$ be the leakage function which takes as input the value $\sigma_{\mathsf{L},r}$, plugs it in Eq. (6) and appends the values of Eq. (5) to obtain $(\sigma^*_i)_{i\in\mathcal{B}_{i_\mathsf{L}}}$, computes the tampered shares $(\tilde{\sigma}^*_i)_{i\in\mathcal{B}_{i_\mathsf{L}}} = f_{i_\mathsf{L}}((\sigma^*_i)_{i\in\mathcal{B}_{i_\mathsf{L}}})$, checks if all the tampered commitments within $\mathcal{T} \cap \mathcal{B}_{i_\mathsf{L}}$ agree on a single value $\tilde{\gamma}_\mathsf{L}$ (and outputs $\perp$ if not), and finally outputs the values $\tilde{\gamma}_\mathsf{L}, (\tilde{\sigma}^{(i)}_{\mathsf{R},j})_{i\in\mathcal{B}_{i_\mathsf{L}}, j\in\mathcal{T}}$.

(b) Let $\hat{g}_{\mathsf{R}}$ be the leakage function which takes as input the value $\sigma_{\mathsf{R},r}$, computes the values $(\sigma^{(i)}_{\mathsf{R},r})_{i\in[n]\setminus\mathcal{J}}$ using $\sigma_{\mathsf{R},r}$ and the values $(\sigma^{(i)}_{\mathsf{R},r})_{i\in\mathcal{J}}$ and plugs them in Eq. (4) in order to obtain $(\sigma^*_i)_{i\in[n]\setminus\mathcal{J}}$; then, appends the values of Eq. (5) to obtain $(\sigma^*_i)_{i\in\mathcal{B}_{i_{\mathsf{R}}}}$, applies $f_{i_{\mathsf{R}}}$ to $(\sigma^*_i)_{i\in\mathcal{B}_{i_{\mathsf{R}}}}$, thus obtaining $(\tilde{\sigma}^*_i)_{i\in\mathcal{B}_{i_{\mathsf{R}}}}$, checks if all the tampered commitments within $\mathcal{T}\cap\mathcal{B}_{i_{\mathsf{R}}}$ agree on a single value $\tilde{\gamma}_{\mathsf{R}}$ (and outputs $\perp$ if not), and finally outputs $\tilde{\gamma}_{\mathsf{R}}$.

(c) For all $i\in\mathcal{T}$, let $f_{\mathsf{L},i}$ be the function which takes as input the value $\sigma_{\mathsf{L},r}$, obtains $(\sigma^*_j)_{j\in\mathcal{B}_{i_{\mathsf{L}}}}$ by appending the values of Eq. (5) and plugging $\sigma_{\mathsf{L},r}$ into Eq. (6), and then computes the tampered shares $(\tilde{\sigma}^*_j)_{j\in\mathcal{B}_{i_{\mathsf{L}}}} = f_{i_{\mathsf{L}}}((\sigma^*_j)_{j\in\mathcal{B}_{i_{\mathsf{L}}}})$ and outputs $\tilde{\sigma}_{\mathsf{L},i}$ if $i\in\mathcal{B}_{i_{\mathsf{L}}}$ and the special share σ^*_{L} otherwise.

(d) For all $i\in\mathcal{T}$, let $f_{\mathsf{R},i}$ be the function which takes as input the value $\sigma_{\mathsf{R},r}$, computes the values $(\sigma^{(i)}_{\mathsf{R},r})_{i\in[n]\setminus\mathcal{J}}$ using $\sigma_{\mathsf{R},r}$ and the values $(\sigma^{(i)}_{\mathsf{R},r})_{i\in\mathcal{J}}$ and plugs them in Eq. (4) in order to obtain $(\sigma^*_i)_{i\in[n]\setminus\mathcal{J}}$; then, appends the values of Eq. (5) to obtain $(\sigma^*_i)_{i\in\mathcal{B}_{i_{\mathsf{R}}}}$, applies $f_{i_{\mathsf{R}}}$ to $(\sigma^*_i)_{i\in\mathcal{B}_{i_{\mathsf{R}}}}$, thus obtaining $(\tilde{\sigma}^*_i)_{i\in\mathcal{B}_{i_{\mathsf{R}}}}$, uses these values along with the values $(\tilde{\sigma}^{(i)}_{\mathsf{R},j})_{i\in\mathcal{B}_{i_{\mathsf{L}}},j\in\mathcal{T}}$ obtained by $\hat{g}_{\mathsf{L}}$ in order to reconstruct $\tilde{\sigma}_{\mathsf{R},i}$ for all $i\in\mathcal{T}$, and finally outputs $\tilde{\sigma}_{\mathsf{R},i}$ if $i\in\mathcal{B}_{i_{\mathsf{L}}}$ and a share $\sigma^*_{\mathsf{R},i}$ such that $\mathsf{Rec}'(\sigma^*_{\mathsf{L}},\sigma^*_{\mathsf{R},i}) = \mathsf{Rec}'(\tilde{\sigma}_{\mathsf{L},i},\tilde{\sigma}_{\mathsf{R},i})$ otherwise.

Send the leakage query $(\hat{g}_{\mathsf{L}},\hat{g}_{\mathsf{R}})$, thus obtaining $((\tilde{\gamma}_{\mathsf{L}},(\tilde{\sigma}^{(i)}_{\mathsf{R},j})_{i\in\mathcal{B}_{i^*},j\in[n]}),\tilde{\gamma}_{\mathsf{R}})$, return $\perp$ to A if $\tilde{\gamma}_{\mathsf{L}}\neq\tilde{\gamma}_{\mathsf{R}}$ and, otherwise, call $\tilde{\gamma}$ the tampered commitment obtained from such a query. Next, for all $i\in\mathcal{T}$, send the tampering query $(f_{\mathsf{L},i},f_{\mathsf{R},i})$, thus obtaining the tampered share $\tilde{\sigma}_i$ (or $\perp$, in which case return $\perp$ to A), and replace $\tilde{\sigma}_i$ with σ_r if $\tilde{\sigma}_i=\diamond$ or replace $\tilde{\sigma}_i$ with σ_j if there exists $j\in[n]$ such that $\tilde{\sigma}_i=\sigma'_j$. Finally, check that there exist no distinct $i_1,i_2\in\mathcal{T}$ s.t. $\mathsf{idx}(\sigma_{i_1})=\mathsf{idx}(\sigma_{i_2})$ (and output $\perp$ otherwise), reconstruct $\tilde{\mu}||\tilde{\rho}=\mathsf{Rec}((\tilde{\sigma}_i)_{i\in\mathcal{T}})$, check that $\tilde{\gamma}=\mathsf{Com}(\tilde{\mu};\tilde{\rho})$ (and return $\perp$ otherwise), replace $\tilde{\mu}$ with $\diamond$ if $\tilde{\mu}\in\{\mu_0,\mu_1\}$ and return $\tilde{\mu}$ to A.

5. **Guess.** Return the same distinguishing bit as that of A.

For the analysis, call $\mathbf{Bad}_i$ the event that one tampering query modifies the shares so that the tampered value $(\tilde{\sigma}_{\mathsf{L},i},\tilde{\sigma}_{\mathsf{R},i})$ is a valid encoding of σ'_r (*i.e.*, the adversary purposely replaces $(\sigma_{\mathsf{L},i},\sigma_{\mathsf{R},i})$ with a valid encoding of σ'_r). Clearly, the probability of the event $\mathbf{Bad}_i$ in the hybrid experiment $\mathbf{Hyb}_{r-1}$ is $O(2^{-\lambda})$ as provoking the event corresponds to guessing the value σ'_r which is uniformly random over $\mathcal{S}_r(|\mu_b|+|\rho|)$. Furthermore, the reduction perfectly simulates $\mathbf{Hyb}_{r-1}(\lambda,b,1)$ if the target codeword encodes σ_r and conditioning on $\mathbf{Bad}=\bigcup_i\mathbf{Bad}_i$ not happening. On the other hand, if the target codeword encodes σ'_r, the reduction perfectly simulates $\mathbf{Hyb}_r(\lambda,b,1)$. In particular, the latter holds because: (i) By perfect privacy of Shamir's secret sharing the distribution of the shares $(\sigma^*_i)_{i\in\mathcal{B}_{i_{\mathsf{L}}}}$ and $(\sigma^*_i)_{i\in\mathcal{B}_{i_{\mathsf{R}}}}$ computed inside the leakage and tampering oracles is identical to that of the target secret sharing of either $\mathbf{Hyb}_{r-1}(\lambda,b,1)$ or $\mathbf{Hyb}_r(\lambda,b,1)$; (ii) The auxiliary information leaked by the functions $(\hat{g}_{\mathsf{L}},\hat{g}_{\mathsf{R}})$, along with the answer to the tampering queries $(f_{\mathsf{L},i},f_{\mathsf{R},i})_{i\in[t]}$, yield a perfect simulation of A's tampering query.

Hence, to conclude the proof, it only remains to show that the constraints on the leakage hold. The amount of leakage performed by the reduction is exactly the one performed by A, plus the leakage used to obtain the tampered commitments $\tilde{\gamma}_{\mathsf{L}},\tilde{\gamma}_{\mathsf{R}}$ and the tampered shares $(\tilde{\sigma}^{(i)}_{\mathsf{R},j})_{i\in\mathcal{B}_{i^*},j\in[n]}$, therefore we need:

$$\ell_{\mathsf{L}} \geq \ell^* + t\cdot(t-1)\cdot s_{\mathsf{R}} + |\gamma| \qquad \text{and} \qquad \ell_{\mathsf{R}} \geq \ell^* + |\gamma|.$$

The lemma follows. □

4.2 Inductive Step

The lemma below constitutes the inductive step.

Lemma 3. *Fix any $p \in \mathsf{poly}(\lambda)$ and assume that for all $b \in \{0,1\}$, and all $r \in [n]$, it holds:*

$$\left\{\mathbf{Hyb}_{r-1}(\lambda, b, p)\right\}_{\lambda\in\mathbb{N}} \overset{\mathsf{s}}{\approx} \left\{\mathbf{Hyb}_{r}(\lambda, b, p)\right\}_{\lambda\in\mathbb{N}}.$$

Then,

$$\left\{\mathbf{Hyb}_{r-1}(\lambda, b, p+1)\right\}_{\lambda\in\mathbb{N}} \overset{\mathsf{s}}{\approx} \left\{\mathbf{Hyb}_{r}(\lambda, b, p+1)\right\}_{\lambda\in\mathbb{N}}.$$

Proof. Again, the difference between the two hybrids is that in $\mathbf{Hyb}_r(\lambda, b, 1)$ the share σ'_r is uniformly random, whereas in $\mathbf{Hyb}_{r-1}(\lambda, b, 1)$ the share σ'_r is set to be σ_r. For any $j \in [n]$, let $\xi(j) \in \{1,2\}$ be the index such that $j \in \mathcal{B}_{\xi(j)}$. The proof proceeds by reduction to leakage-resilient t-time non-malleability of Π'. In more detail, for a fixed choice of $b \in \{0,1\}$ and $r \in [n]$, let A be a (possibly inefficient) adversary telling apart the two hybrids with probability at least $1/\mathsf{poly}(\lambda)$. Consider the following (possibly inefficient) adversary A' attacking Π':

1. **Setup.** As in the proof of Lemma 2.
2. **Shared randomness.** As in the proof of Lemma 2. Recall that, after this step, the reduction A' knows σ^*_{L} and the following values:

$$\forall i \in [n] \setminus \mathcal{J}: \qquad \gamma, \qquad \sigma_{\mathsf{L},i}, \qquad (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]\setminus\{r\}} \tag{7}$$

$$\forall i \in \mathcal{J} \setminus \{r\}: \qquad \gamma, \qquad \sigma_{\mathsf{L},i}, \qquad (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]} \tag{8}$$

$$\text{For } i = r: \qquad \gamma, \qquad (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]} \tag{9}$$

3. **Leakage queries.** As in the proof of Lemma 2. Call Λ the final transcript corresponding to all leakage queries.
4. **First p tampering queries.** Upon receiving a tampering query $(\mathcal{T}^{(q)}, (f_1^{(q)}, f_2^{(q)}))$ from A, construct the following leakage functions.
   (a) Let $\hat{g}^{(q)}_{\mathsf{L}}$ be the leakage function which takes as input the value $\sigma_{\mathsf{L},r}$, plugs it in Eq. (9) and appends the values of Eq. (8) to obtain $(\sigma^*_i)_{i\in\mathcal{B}_{i_\mathsf{L}}}$, computes the tampered shares $(\tilde{\sigma}^*_i)_{i\in\mathcal{B}_{i_\mathsf{L}}} = f_{i_\mathsf{L}}((\sigma^*_i)_{i\in\mathcal{B}_{i_\mathsf{L}}})$, checks if all the tampered commitments within $\mathcal{T}^{(q)} \cap \mathcal{B}_{i_\mathsf{L}}$ agree on a single value $\tilde{\gamma}_\mathsf{L}$ (and outputs $\perp$ if not), and finally outputs the value $\tilde{\gamma}_\mathsf{L}$.
   (b) Let $\hat{g}^{(q)}_{\mathsf{R}}$ be the leakage function which takes as input the value $\sigma_{\mathsf{R},r}$, computes the values $(\sigma^{(i)}_{\mathsf{R},r})_{i\in[n]\setminus\mathcal{J}}$ using $\sigma_{\mathsf{R},r}$ and the values $(\sigma^{(i)}_{\mathsf{R},r})_{i\in\mathcal{J}}$ and plugs them in Eq. (7) in order to obtain $(\sigma^*_i)_{i\in[n]\setminus\mathcal{J}}$; then, appends the values of Eq. (8) to obtain $(\sigma^*_i)_{i\in\mathcal{B}_{i_\mathsf{R}}}$, applies f_{i_R} to $(\sigma^*_i)_{i\in\mathcal{B}_{i_\mathsf{R}}}$, thus obtaining $(\tilde{\sigma}^*_i)_{i\in\mathcal{B}_{i_\mathsf{R}}}$, checks if all the tampered commitments within $\mathcal{T}^{(q)} \cap \mathcal{B}_{i_\mathsf{R}}$ agree on a single value $\tilde{\gamma}_\mathsf{R}$ (and outputs $\perp$ if not), and finally outputs the value $\tilde{\gamma}_\mathsf{R}$.

   Then, send $(\hat{g}^{(q)}_{\mathsf{L}}, \hat{g}^{(q)}_{\mathsf{R}})$ to the leakage oracle, thus obtaining $(\tilde{\gamma}_\mathsf{L}, \tilde{\gamma}_\mathsf{R})$, return $\perp$ and self-destruct if either $\tilde{\gamma}_\mathsf{L} \neq \tilde{\gamma}_\mathsf{R}$ or one of the commitments equals $\perp$, and otherwise call $\tilde{\gamma}^{(q)}$ the tampered commitment. Find, by brute force, an opening $(\tilde{\mu}^{(q)}, \tilde{\rho}^{(q)})$ for $\tilde{\gamma}^{(q)}$, replace $\tilde{\mu}^{(q)}$ with $\diamond$ if $\tilde{\mu}^{(q)} \in \{\mu_0, \mu_1\}$, and finally return $\tilde{\mu}^{(q)}$ to A (or output $\perp$ and self-destruct if no opening is found).
5. **Last tampering query.** Upon receiving the last tampering query $(\mathcal{T}^{(p+1)}, (f_1^{(p+1)}, f_2^{(p+1)}))$ from A, construct the following leakage and tampering functions.
   (a) Wlog., assume that the output of A is equal to 0 whenever it believes that the target secret sharing is distributed as in $\mathbf{Hyb}_{r-1}(\lambda, b, p+1)$.

(b) Check that the simulation up to the first p tampering queries did not cause any inconsistency, due to the fact that the outcome of the q-th tampering query would have been $\perp$ because $(\tilde{\sigma}_i^*)_{i\in\mathcal{T}^{(q)}}$ was not a valid secret sharing. Namely, let $\hat{h}$ be the leakage function that hard-wires a description of A, the values $(\gamma, \sigma_r, \sigma_r')$, a description of the final tampering query $(\mathcal{T}^{(p+1)}, (f_1^{(p+1)}, f_2^{(p+1)}))$, the answer to all the previous tampering queries $\tilde{\mu}^{(1)}, \ldots, \tilde{\mu}^{(p)}$ and the answer to all the previous leakage queries Λ and proceeds as follows.

i. Upon input $\sigma_{\mathsf{L},r}$, construct a set $\hat{\mathcal{S}}_{\mathsf{R}}$ containing all the possible right shares $\hat{\sigma}_{\mathsf{R}}$ such that $(\sigma_{\mathsf{L},r}, \hat{\sigma}_{\mathsf{R}})$ is either a secret sharing of σ_r or a secret sharing of σ_r' and, additionally, $(\sigma_{\mathsf{L},r}, \hat{\sigma}_{\mathsf{R}})$ is also compatible with the transcript of all the previous leakage queries being Λ and the answer to all the previous tampering queries being $\tilde{\mu}^{(1)}, \ldots, \tilde{\mu}^{(p)}$.

ii. For each $\hat{\sigma}_{\mathsf{R}} \in \hat{\mathcal{S}}_{\mathsf{R}}$, let $(\hat{\sigma}_{\mathsf{R},r}^{(j)})_{j\in[n]}$ be the secret sharing of $\hat{\sigma}_{\mathsf{R}}$ such that $(\hat{\sigma}_{\mathsf{R},r}^{(j)})_{j\in\mathcal{J}} = (\sigma_{\mathsf{R},r}^{(j)})_{j\in\mathcal{J}}$ and let $(\hat{\sigma}_{\mathsf{R},i}^{(j)})_{j\in[n]} = (\sigma_{\mathsf{R},i}^{(j)})_{j\in[n]}$ for all $i \neq r$, and $\hat{\sigma}_i^* = (\gamma, \sigma_{\mathsf{L},i}, (\hat{\sigma}_{\mathsf{R},j}^{(i)})_{j\in[n]})$ for all $i \in [n]$. Finally, let $\sigma^* = (\sigma_1^*, \ldots, \sigma_n^*)$.

iii. For each $\hat{\sigma}_{\mathsf{R}} \in \hat{\mathcal{S}}_{\mathsf{R}}$, run A answering to all of its leakage and tampering queries until the last tampering query arrives. Then, compute the result $\tilde{\mu}^*$ of such query using σ^*, forward $\tilde{\mu}^*$ to A and obtain the distinguishing bit b'.

iv. Output a bit $\hat{b}$ such that $\hat{b} = 1$ if and only if, in the previous step, A outputs $b' = 0$ more often when σ^* is distributed as in $\mathbf{Hyb}_{r-1}(\lambda, b, p+1)$.

(c) Let $\hat{g}_{\mathsf{L}}^{(p+1)}, \hat{g}_{\mathsf{R}}^{(p+1)}$ be the same as the leakage functions $\hat{g}_{\mathsf{L}}, \hat{g}_{\mathsf{R}}$ in the proof of Lemma 2.

(d) For all $i \in \mathcal{T}$, let $\hat{f}_{\mathsf{L},i}^{(p+1)}, \hat{f}_{\mathsf{R},i}^{(p+1)}$ be the same as the tampering functions $\hat{f}_{\mathsf{L}}, \hat{f}_{\mathsf{R}}$ in the proof of Lemma 2.

Then, send $(\hat{h}, \varepsilon)$ to the leakage oracle, obtaining a bit $\hat{b}$. Also, send $(\hat{g}_{\mathsf{L}}^{(p+1)}, \hat{g}_{\mathsf{R}}^{(p+1)})$ to the leakage oracle and, for all $i \in \mathcal{T}$, send $(\hat{f}_{\mathsf{L},i}^{(p+1)}, \hat{f}_{\mathsf{R},i}^{(p+1)})$ to the tampering oracle, and compute the tampered message $\tilde{\mu}^{(p+1)} \in \mathcal{M} \cup \{\diamond, \perp\}$ as the value $\tilde{\mu}$ in the proof of Lemma 2.

6. **Guess.** If $\hat{b} = 0$ output a random guess, and otherwise return $\tilde{\mu}^{(p+1)}$ to A and output the same distinguishing bit as that of A.

The reduction runs in exponential time. We now show that its distinguishing advantage is negligibly close to that of A. Indeed:

$$
\begin{aligned}
&\left|\Pr\left[\mathbf{LR\text{-}NMC}_{\Pi',\mathsf{A}'}^{\sigma_r,\sigma_r'}(\lambda,0)=1\right] - \Pr\left[\mathbf{LR\text{-}NMC}_{\Pi',\mathsf{A}'}^{\sigma_r,\sigma_r'}(\lambda,1)=1\right]\right| \\
&= \Bigg|\Pr\left[\mathbf{LR\text{-}NMC}_{\Pi',\mathsf{A}'}^{\sigma_r,\sigma_r'}(\lambda,0)=1 \wedge \hat{b}=1\right] \\
&\qquad\qquad - \Pr\left[\mathbf{LR\text{-}NMC}_{\Pi',\mathsf{A}'}^{\sigma_r,\sigma_r'}(\lambda,1)=1 \wedge \hat{b}=1\right]\Bigg| && (10)\\
&\geq \frac{1}{\mathsf{poly}(\lambda)}\Bigg|\Pr\left[\mathbf{LR\text{-}NMC}_{\Pi',\mathsf{A}'}^{\sigma_r,\sigma_r'}(\lambda,0)=1 \middle| \hat{b}=1\right] \\
&\qquad\qquad - \Pr\left[\mathbf{LR\text{-}NMC}_{\Pi',\mathsf{A}'}^{\sigma_r,\sigma_r'}(\lambda,1)=1 \middle| \hat{b}=1\right]\Bigg| && (11)\\
&\geq \frac{1}{\mathsf{poly}(\lambda)}\left(\frac{1}{\mathsf{poly}(\lambda)} - \mathsf{negl}(\lambda)\right). && (12)
\end{aligned}
$$

In the derivation above, Eq. (10) follows because when $\hat{b} = 0$, the reduction A' returns a random guess and thus its distinguishing advantage is zero, Eq. (11) holds as the induction hypothesis implies that $\hat{b} = 1$ with non-negligible probability, otherwise A generates an invalid secret sharing $(\tilde{\sigma}_i^*)_{i\in\mathcal{T}}$ within the first p tampering queries with overwhelming probability, which in turn means that A can distinguish using less than $p+1$ tampering queries. Finally, Eq. (12) holds thanks to an analysis similar to the proof of Lemma 2 shows that the view of A is perfectly simulated (except with negligible probability) conditioned on $\hat{b}=1$, and thus in this case A' retains essentially the same advantage as that of A.

In order to conclude the proof, it remains to show that A' is ℓ^*-leakage admissible, for ℓ^* as in the statement of the theorem. Note that the adversary A' makes leakage queries by forwarding the leakage queries of A (in step 3), for obtaining the tampered commitments (in step 4a and step 4b) and the auxiliary information needed to answer the last tampering query (in step 5c), and finally for running the consistency check (in step 5b). However, the leakage performed in step 5b and step 5c is executed only once and for a total of at most $t \cdot (t-1) \cdot s_{\mathsf{R}} + 1$ bits. Let $q^* \in \mathbb{N}$ be the index of the tampering query, if any, where the commitments retrieved with the leakage functions $\hat{g}_{\mathsf{L}}^{(q)}, \hat{g}_{\mathsf{R}}^{(q)}$ happen to be either different or $\perp$, and set $q^* = p+1$ in case that never happens; note that q^* is a random variable, which we denote by $\mathbf{q}^*$. Clearly, such leakage queries are executed exactly $\min\{q^*, p\}$ times. For each $i \in \{\mathsf{L}, \mathsf{R}\}$, denote by $\mathbf{\Lambda}_i'$ the random variable corresponding to the leakage performed by the reduction $\hat{\mathsf{A}}$ on the share $\sigma_{i,r}$ of the target secret sharing $(\mathbf{\Sigma}_{\mathsf{L},r}, \mathbf{\Sigma}_{\mathsf{R},r})$. We can write:

$$
\begin{aligned}
&\widetilde{\mathbb{H}}_\infty\left(\mathbf{\Sigma}_{\mathsf{L},r} \middle| \mathbf{\Lambda}_{\mathsf{L}}'\right) \\
&\geq \widetilde{\mathbb{H}}_\infty\left(\mathbf{\Sigma}_{\mathsf{L},r} \middle| \mathbf{\Sigma}_{\mathsf{R},r}, \mathbf{\Lambda}_{\mathsf{L}}'\right) &(13)\\
&\geq \widetilde{\mathbb{H}}_\infty\left(\mathbf{\Sigma}_{\mathsf{L},r} \middle| \mathbf{\Sigma}_{\mathsf{R},r}, \hat{g}_{\mathsf{L}}^{(1)}(\mathbf{\Sigma}_{\mathsf{L},r}), \ldots, \hat{g}_{\mathsf{L}}^{(\mathbf{q}^*)}(\mathbf{\Sigma}_{\mathsf{L},r}), \hat{g}_{\mathsf{L}}^{(p+1)}(\mathbf{\Sigma}_{\mathsf{L},r}), \hat{h}(\mathbf{\Sigma}_{\mathsf{L},r})\right) - \ell^* &(14)\\
&\geq \widetilde{\mathbb{H}}_\infty\left(\mathbf{\Sigma}_{\mathsf{L},r} \middle| \mathbf{\Sigma}_{\mathsf{R},r}, \hat{g}_{\mathsf{L}}^{(1)}(\mathbf{\Sigma}_{\mathsf{L},r}), \ldots, \hat{g}_{\mathsf{L}}^{(\mathbf{q}^*)}(\mathbf{\Sigma}_{\mathsf{L},r}), \hat{g}_{\mathsf{L}}^{(p+1)}(\mathbf{\Sigma}_{\mathsf{L},r})\right) - \ell^* - 1 &(15)\\
&\geq \widetilde{\mathbb{H}}_\infty\left(\mathbf{\Sigma}_{\mathsf{L},r} \middle| \mathbf{\Sigma}_{\mathsf{R},r}, \mathbf{q}^*, \hat{g}_{\mathsf{L}}^{(\mathbf{q}^*)}(\mathbf{\Sigma}_{\mathsf{L},r}), \hat{g}_{\mathsf{L}}^{(p+1)}(\mathbf{\Sigma}_{\mathsf{L},r})\right) - \ell^* - 1 &(16)\\
&\geq \widetilde{\mathbb{H}}_\infty\left(\mathbf{\Sigma}_{\mathsf{L},r} | \mathbf{\Sigma}_{\mathsf{R},r}\right) - |\gamma| - O(\log(\lambda)) - t \cdot (t-1) \cdot s_{\mathsf{R}} - \ell^* - 1 &(17)\\
&\geq \widetilde{\mathbb{H}}_\infty\left(\mathbf{\Sigma}_{\mathsf{L},r}\right) - d - |\gamma| - O(\log(\lambda)) - t \cdot (t-1) \cdot s_{\mathsf{R}} - \ell^* - 1. &(18)
\end{aligned}
$$

In the above derivation, Eq. (13) follows by the fact that further conditioning can only reduce the conditional average min-entropy and Eq. (14) follows by definition of the leakage $\mathbf{\Lambda}_{\mathsf{L}}'$ and by the fact that A is ℓ^*-leakage admissible. Eq. (15) follows by the Chain Rule as the function $\hat{h}$ outputs a single bit. Eq. (16) follows by the fact that, for each $q < q^*$, the commitments leaked from the left part are identical to the ones leaked from the right part and thus can be computed as a deterministic function of $\mathbf{\Sigma}_{\mathsf{R},r}$ and q^*. Eq. (17) follows again by the Chain Rule, since $|q^*| = O(\log\lambda)$ and either $\mathbf{q}^* = p+1$ (in which case the min-entropy drop due to the leakage $\hat{g}_{\mathsf{L}}^{(\mathbf{q}^*)}(\mathbf{\Sigma}_{\mathsf{L},r}), \hat{g}_{\mathsf{L}}^{(p+1)}(\mathbf{\Sigma}_{\mathsf{L},r})$ is bounded by the size $|\gamma|$ of a commitment plus $t \cdot (t-1) \cdot s_{\mathsf{R}}$ bits for simulating the last tampering query) or $\mathbf{q}^* \leq p$ (in which case the reduction self-destructs and only the value $\hat{g}_{\mathsf{L}}^{(\mathbf{q}^*)}(\mathbf{\Sigma}_{\mathsf{L},r})$ is leaked, causing a loss of at most $|\gamma|$ bits in the min-entropy bound). Finally, Eq. (18) follows by property (ii) of the secret sharing scheme Π'.

An almost identical analysis holds for the leakage from the right share, with the only difference that we do not leak any auxiliary information from $\mathbf{\Sigma}_{\mathsf{R},r}$ and therefore the min-entropy drop only amounts to $d + |\gamma| + O(\log(\lambda)) + \ell^* + 1$. The lemma follows. □

4.3 Putting it Together

By Lemma 2 and Lemma 3, we get that, for all $b \in \{0,1\}$ and all $r \in [n]$,

$$\left\{\mathbf{Hyb}_{r-1}(\lambda, b)\right\}_{\lambda\in\mathbb{N}} \overset{\mathsf{s}}{\approx} \left\{\mathbf{Hyb}_{r}(\lambda, b)\right\}_{\lambda\in\mathbb{N}}.$$

Hence, by repeatedly applying the triangular inequality, we have obtained

$$\{\mathbf{Hyb}_0(\lambda, b)\}_{\lambda\in\mathbb{N}} \overset{\mathsf{s}}{\approx} \{\mathbf{Hyb}_n(\lambda, b)\}_{\lambda\in\mathbb{N}}.$$

The lemma below concludes the proof of the theorem.

Lemma 4. $\{\mathbf{Hyb}_n(\lambda, 0)\}_{\lambda\in\mathbb{N}} \overset{\mathsf{c}}{\approx} \{\mathbf{Hyb}_n(\lambda, 1)\}_{\lambda\in\mathbb{N}}$.

Proof. By reduction to the computational hiding property of the commitment. Suppose that there exists a PPT adversary A able to distinguish between $\mathbf{Hyb}_n(\lambda, 0)$ and $\mathbf{Hyb}_n(\lambda, 1)$ with probability at least $1/\mathsf{poly}(\lambda)$. Consider the following reduction A' trying to distinguish if γ is a commitment to μ_0 or to μ_1.

- Sample n uniformly random shares $\sigma_1, \ldots, \sigma_n$ and, for all $i \in [n]$, compute $(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \leftarrow\!\$\ \mathsf{Share}'(\sigma_i)$ and $(\sigma_{\mathsf{R},i}^{(1)}, \ldots, \sigma_{\mathsf{R},i}^{(n)}) \leftarrow\!\$\ \mathsf{Share}(\sigma_{\mathsf{R},i})$.
- Set $(\sigma_1^*, \ldots, \sigma_n^*)$, where $\sigma_i^* = (\gamma, \sigma_{\mathsf{L},i}, (\sigma_{\mathsf{R},j}^{(i)})_{j\in[n]})$ for all $i \in [n]$.
- Answer leakage and tampering queries from A as described in the last hybrid experiment.
- Output the same distinguishing bit as A does.

For the analysis, note that A' is efficient and perfectly simulates $\mathbf{Hyb}_n(\lambda, 0)$ whenever γ is a commitment to μ_0, and perfectly simulates $\mathbf{Hyb}_n(\lambda, 1)$ whenever γ is a commitment to μ_1. Hence, A' has non-negligible distinguishing advantage. This concludes the proof. □

5 Rate Compilers and Capacity Upper Bounds

In this section, we first establish an upper bound on the capacity of continuously non-malleable threshold secret sharing against joint tampering. We focus on secret sharing scheme that are not leakage resilient. Indeed, an upper bound on the capacity of this weaker primitive implies an upper bound on the capacity of leakage-resilient continuous non-malleable secret sharing schemes. Additionally, we exhibit a compiler for boosting the rate of our construction from the previous section so that it achieves the best possible rate in the plain model. For completeness, in Appendix B, we show that our upper bound on the capacity can be overcome both in the random oracle model (ROM) and in the algebraic generic group model (AGM).

5.1 Capacity Upper Bounds

We show the following upper bound on the maximal achievable rate of any continuously non-malleable secret sharing scheme against k-joint tampering, for $k > t/2$. Recall that computational assumptions are inherent for continuous non-malleability, and thus our negative results hold even in the computational setting.

Theorem 2. *Let Π be a t-out-of-n k-CNMSS scheme. If $k > t/2$, then Π cannot achieve better asymptotic rate than $\varrho \leq t - k$.*

Proof. We prove the slightly stronger statement that the capacity upper bound holds even if the attacker always uses the same reconstruction set $\mathcal{T}$ across all tampering queries. For simplicity, we assume that the share space of Π is $\mathcal{S} = \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ with $|\mathcal{S}_i| = |\mathcal{S}_j|$ for all $i, j \in [n]$. (A generalization is immediate.) Consider the following commitment scheme:

- The commitment procedure Com, upon input a message μ and random coins ρ, samples shares $(\sigma_1, \ldots, \sigma_n) := \mathsf{Share}(\mu; \rho)$ and outputs $\gamma = (\sigma_1, \ldots, \sigma_{t-k})$.
- The opening procedure, upon input an opening μ, ρ and a commitment γ, recomputes the shares $\sigma_1, \ldots, \sigma_n$ and checks that γ equals the first $t - k$ shares.

We now prove that the above defined commitment scheme is perfectly binding. Note that the latter implies that $|\mu| \le |\gamma|$ because Com must be an injective function. Thus, by letting $s = \log|\mathcal{S}_1|$, the rate satisfies $\varrho = |\mu|/s \le |\gamma|/s \le t-k$ (as desired).

Towards a contradiction, assume that Com is not perfectly binding. Namely, there exist a commitment γ and two openings $(\mu^{(0)}, \rho_0)$ and $(\mu^{(1)}, \rho_1)$ such that both openings are valid for γ and $\mu^{(0)} \neq \mu^{(1)}$. Consider the following PPT attacker against continuous non-malleability, with the values $\mu^{(0)}, \rho_0, \mu^{(1)}, \rho_1$ hard-coded in:

1. Let μ_0^* and μ_1^* be any two distinct messages, and denote by $(\sigma_1, \ldots, \sigma_n)$ the target secret sharing of μ_b^* in the experiment defining continuous non-malleability. For better readability, set $\ell = |(\sigma_{t-k+1}||\cdots||\sigma_t)|$.
2. Compute the shares $(\sigma_0^{(0)}, \ldots, \sigma_n^{(0)}) := \mathsf{Share}(\mu^{(0)}; \rho_0)$ and $(\sigma_0^{(1)}, \ldots, \sigma_n^{(1)}) := \mathsf{Share}(\mu^{(1)}; \rho_1)$. By validity of the openings, we have that $\sigma_i^{(0)} = \sigma_i^{(1)}$ for all $i \in [t-k]$.
3. Set $\mathcal{T} := [t]$, $\mathcal{B}_1 := [t-k]$, and $\mathcal{B}_2 := [t] \setminus [t-k]$.
4. For each $j \in [\ell]$, the j-th tampering query is defined to be $(\mathcal{T}, (f_1, f_2^{(j)}))$ where the tampering functions are specified as follows:
   - $f_1((\sigma_i)_{i\in\mathcal{B}_1}) := (\sigma_i^{(0)})_{i\in\mathcal{B}_1}$.
   - $f_2^{(j)}((\sigma_i)_{i\in\mathcal{B}_2})$ is the function that outputs $(\sigma_i^{(0)})_{i\in\mathcal{B}_2}$ if and only if the j-th bit of the string $(\sigma_i)_{i\in\mathcal{B}_2}$ equals 0 (and outputs $(\sigma_i^{(1)})_{i\in\mathcal{B}_2}$ otherwise).

   Let $\tilde{\mu}$ be the output of the j-th tampering query. Set $\alpha_j := 0$ if and only if $\tilde{\mu} = \mu^{(0)}$ (and $\alpha_j := 1$ otherwise).
5. Parse the string $\alpha_1, \ldots, \alpha_\ell$ as $\sigma_{t-k+1}, \ldots, \sigma_t$. Forward the query $(\mathcal{T}, (f_1', f_2'))$ to the tampering oracle, where f_1' takes as input $(\sigma_i)_{i\in\mathcal{B}_1}$, reconstructs the message μ_b^* (using the values $\sigma_{t-k+1}, \ldots, \sigma_t$), and finally either does nothing (say, if the reconstructed message is μ_0^*) or outputs garbage.
6. Output $b' = 0$ if and only if the output of the last tampering query is $\diamond$ (and otherwise output $b' = 1$).

Note that $t-k < k$ as $k > t/2$, and thus $\mathcal{B} = (\mathcal{B}_1, \mathcal{B}_2)$ is a k-sized partition of $\mathcal{T} = [t]$. Moreover, the above reduction clearly breaks continuous non-malleability of Π with overwhelming probability. This concludes the proof. $\square$

5.2 Rate Compiler (Plain Model)

Let $(\mathsf{IDisp}, \mathsf{IRec})$ be an information dispersal, $\Pi = (\mathsf{Share}, \mathsf{Rec})$ be a secret sharing scheme and $\Sigma = (\mathsf{Enc}, \mathsf{Dec})$ be a secret-key encryption scheme. Consider the following derived secret sharing scheme $\Pi^* = (\mathsf{Share}^*, \mathsf{Rec}^*)$.

- **Algorithm Share^*:** upon input a message μ, sample a random key $\kappa \leftarrow\$ \mathcal{K}$ and compute $\gamma \leftarrow\$ \mathsf{Enc}(\kappa, \mu)$, $(\gamma_1, \ldots, \gamma_n) = \mathsf{IDisp}(\gamma)$ and $(\kappa_1, \ldots, \kappa_n) \leftarrow\$ \mathsf{Share}(\kappa)$; finally, for all $i \in [n]$, let $\sigma_i = (\kappa_i, \gamma_i)$ and output $(\sigma_1, \ldots, \sigma_n)$.
- **Algorithm Rec^*:** upon input a set $(\sigma_i)_{i\in\mathcal{I}}$ of at least t shares parse $\sigma_i = (\kappa_i, \gamma_i)$ for all $i \in \mathcal{I}$, reconstruct $\kappa = \mathsf{Rec}((\kappa_i)_{i\in\mathcal{I}})$ and $\gamma = \mathsf{IRec}((\gamma_i)_{i\in\mathcal{I}})$, check that $\mathsf{IDisp}(\gamma)_\mathcal{I} = (\gamma_i)_{i\in\mathcal{I}}$ (and return $\perp$ if not), and finally output $\mu = \mathsf{Dec}(\kappa, \gamma)$.

The construction above was first proposed and analyzed by Krawczyk [Kra94] in the setting of plain threshold secret sharing. The theorem below states its security in the setting of continuous joint tampering and leakage attacks.

Theorem 3. *Let $n, t, t^*, \ell \in \mathbb{N}$ be parameters such that $t^* \le t-1$. Assume that:*

- $(\mathsf{IDisp}, \mathsf{IRec})$ *is a t^*-out-of-n information dispersal;*
- $(\mathsf{Share}, \mathsf{Rec})$ *is a t-out-of-n $(\ell, t-1)$-LR-CNMSS;*

$\mathbf{LR\text{-}CNMSS}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, b)$ $\boxed{\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, b)}$:

$\kappa \leftarrow\!\!\$\; \mathcal{K}$
$\boxed{\hat{\kappa} \leftarrow\!\!\$\; \mathcal{K}}$
$(\kappa_1, \ldots, \kappa_n) \leftarrow\!\!\$\; \mathsf{Share}(\kappa)$
$\boxed{(\kappa_1, \ldots, \kappa_n) \leftarrow\!\!\$\; \mathsf{Share}(\hat{\kappa})}$
$\gamma \leftarrow\!\!\$\; \mathsf{Enc}(\kappa, \mu_b)$
$(\gamma_1, \ldots, \gamma_n) = \mathsf{IDisp}(\gamma)$
$\forall i \in [n]$:
  $\sigma_i := (\kappa_i, \gamma_i)$
$\sigma := (\sigma_1, \ldots, \sigma_n)$
$\mathtt{stop} \leftarrow \mathtt{false}$
Return $\mathsf{A}^{\mathcal{O}_{\mathsf{tamp}}(\sigma,\cdot),\mathcal{O}_{\mathsf{leak}}(\sigma,\cdot)}(1^\lambda)$

Oracle $\mathcal{O}_{\mathsf{leak}}(\sigma, (g_1, \ldots, g_m))$:
If $\mathtt{stop} = \mathtt{true}$, return $\perp$
Else, return $g_1(\sigma_{\mathcal{B}_1}), \ldots, g_m(\sigma_{\mathcal{B}_m})$

Oracle $\mathcal{O}_{\mathsf{tamp}}(\sigma, \mathcal{T}, (f_1, \ldots, f_m))$:
If $\mathtt{stop} = \mathtt{true}$, return $\perp$
$\forall i \in [m] : \tilde{\sigma}_{\mathcal{B}_i} := f_i(\sigma_{\mathcal{B}_i})$
$\tilde{\sigma} = (\sigma_1, \ldots, \sigma_n)$
$\forall i \in \mathcal{T}, \tilde{\sigma}_i = (\tilde{\kappa}_i, \tilde{\gamma}_i)$
$\tilde{\gamma} = \mathsf{IRec}((\tilde{\gamma}_i)_{i \in \mathcal{T}})$
If $\mathsf{IDisp}(\tilde{\gamma})_{\mathcal{T}} \neq (\tilde{\gamma}_i)_{i \in \mathcal{T}}$
  $\mathtt{stop} \leftarrow \mathtt{true}$ and return $\perp$
$\tilde{\kappa} = \mathsf{Rec}((\tilde{\kappa}_i)_{i \in \mathcal{T}})$
If $\tilde{\kappa} = \perp$, $\mathtt{stop} \leftarrow \mathtt{true}$ and return $\perp$
$\boxed{\text{If } \tilde{\kappa} = \hat{\kappa},\ \tilde{\kappa} \leftarrow \kappa}$
$\tilde{\mu} = \mathsf{Dec}(\tilde{\kappa}, \tilde{\gamma})$
If $\tilde{\mu} = \perp$, $\mathtt{stop} \leftarrow \mathtt{true}$ and return $\perp$
If $\tilde{\mu} \in \{\mu_0, \mu_1\}$, return $\diamond$
Else return $\tilde{\mu}$

Figure 4: Experiments in the proof of Theorem 3. The instructions boxed in red are the modifications introduced by the hybrid experiment.

- $(\mathsf{Enc}, \mathsf{Dec})$ *is an IND-CCA secure secret-key encryption scheme.*

Then, Π^* is a t-out-of-n $(\ell, t-1)$-LR-CNMSS under the following restriction: Each tampering query $(\mathcal{T}, f)$ output by the attacker is such that, for all subsets $\mathcal{B}_i$ of the partition $\mathcal{B}$, either $\mathcal{B}_i \cap \mathcal{T} = \emptyset$ or $|\mathcal{B}_i \cap \mathcal{T}| \geq t^*$. Moreover, the asymptotic rate of Π^* is $\varrho = t^*$.

Proof. The proof proceeds by a hybrid argument. In particular, we argue that the original experiment is computationally close to a mental experiment in which we replace the secret sharing of the key κ with a secret sharing of an unrelated random key $\hat{\kappa}$. The mental experiments is depicted in Figure 4 along with the original experiment in order to highlight the main differences. The lemma below states that the two experiments are computationally indistinguishable.

Lemma 5. *For all $\mu_0, \mu_1 \in \mathcal{M}$, all $(t-1)$-sized partitions $\mathcal{B}$ of $[n]$, and all $b \in \{0, 1\}$, it holds that:*

$$\left\{\mathbf{LR\text{-}CNMSS}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A}}(\lambda, b)\right\}_{\lambda \in \mathbb{N}} \overset{c}{\approx} \left\{\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, b)\right\}_{\lambda \in \mathbb{N}}.$$

Proof. Before going through the proof, we introduce the following two procedures which are parameterized by two messages μ_0, μ_1, secret sharing scheme Π, a partition $\mathcal{B}$ of $[n]$ and a reconstruction set $\mathcal{T}$, and which involve an adversary A with black-box access to the tampering oracle $\mathcal{O}_{\mathsf{tamp}}(\sigma, \cdot, \cdot)$.

- **Procedure** $\mathsf{LeakTamper}^{\mu_0,\mu_1}_{\Pi,\mathcal{B},\mathcal{T}}(g_1, \ldots, g_m)$**:** fix any three pairwise distinct messages $\mu^{(0)}$, $\mu^{(1)}, \mu^{(\mathsf{eof})} \in \mathcal{M} \setminus \{\mu_0, \mu_1\}$, let $(\sigma_1^{(b)}, \ldots, \sigma_n^{(b)}) \leftarrow\!\!\$\; \mathsf{Share}(\mu^{(b)})$ for all $b \in \{0, 1, \mathsf{eof}\}$, set $\Lambda^* = \varepsilon$ and $i^* = 1$ and perform the following loop.
  - For all $i \in [m]$, construct a tampering function $\hat{f}_i$ that, upon input $(\sigma_j)_{j \in \mathcal{B}_i}$, computes $\Lambda = g_i((\sigma_j)_{j \in \mathcal{B}_i})$ and outputs either $(\sigma_j^{(b)})_{j \in \mathcal{B}_i}$, where b is the i^*-th bit of Λ, or $(\sigma_j^{(\mathsf{eof})})_{j \in \mathcal{B}_i}$ if $i^* > |\Lambda|$ (unless $\Lambda = \perp$, in which case the output is set to $\perp$).
  - Send $(\mathcal{T}, (\hat{f}_1, \ldots, \hat{f}_m))$ to the tampering oracle, thus obtaining either of $\mu^{(0)}, \mu^{(1)}$, $\mu^{(\mathsf{eof})}$ or $\perp$ (if the outputs of g_i differ for some $i_1, i_2 \in \mathcal{T}$).

– If the oracle replies with $\perp$, output $\perp$ and terminate. Else, if the output is $\mu^{(\mathsf{eof})}$, stop the loop and output Λ^*. Else, let $\Lambda^* \leftarrow \Lambda^* || b$, where $\mu^{(b)}$ is the received message, set $i^* \leftarrow i^* + 1$ and continue the loop.

- **Procedure** $\mathsf{CheckEq}_{\Pi,\mathcal{B},\mathcal{T}}(\hat{\Lambda}, g_1, \ldots, g_m)$**:** for all $i \in [m]$, construct a tampering function $\hat{f}_i$ that, upon input $(\sigma_j)_{j\in\mathcal{B}_i}$, computes $\Lambda = g_i((\sigma_j)_{j\in\mathcal{B}_i})$ and outputs $(\sigma_j)_{j\in\mathcal{B}_i}$ if $\Lambda = \hat{\Lambda}$ and $\perp$ otherwise; then, send $(\mathcal{T}, (\hat{f}_1, \ldots, \hat{f}_m))$ to the tampering oracle and obtain a message that is either $\diamond$ or $\perp$.

When they are clear from the context, we will omit the values μ_0, μ_1, as well as $\Pi, \mathcal{B}$, thus writing only $\mathsf{LeakTamper}_{\mathcal{T}}(g_1, \ldots, g_m)$ and $\mathsf{CheckEq}_{\mathcal{T}}(\hat{\Lambda}, g_1, \ldots, g_m)$.

To prove the lemma, we proceed by reduction to leakage-resilient continuous non-malleability of the underlying secret sharing scheme. Fix any $b \in \{0, 1\}$, and suppose that there exist two messages $\mu_0, \mu_1 \in \mathcal{M}$, a $(t-1)$-sized partition $\mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_m)$ of $[n]$, and a PPT adversary A able to distinguish between the two experiments with non-negligible probability. Consider the following reduction $\hat{\mathsf{A}}$ attacking leakage-resilient continuous non-malleability of Π.

1. **Setup.** Set the challenge messages to be κ and $\hat{\kappa}$ sampled as in the hybrid experiment, then compute $\gamma \leftarrow\!\!\$\ \mathsf{Enc}(\kappa, \mu_b)$ and $(\gamma_1, \ldots, \gamma_n) = \mathsf{IDisp}(\gamma)$. In what follows, all tampering and leakage functions constructed by the reduction implicitly hard-wire the shares $(\gamma_j)_{j\in\mathcal{B}_i}$ and (a description of) the query they refer to.
2. **Leakage queries.** Upon input a leakage query $(g_1, \ldots, g_m)$, for all $i \in [m]$, construct the leakage function $\hat{g}_i$ which takes as input the shares $(\kappa_j)_{j\in\mathcal{B}_i}$, runs $\Lambda_i = g_i((\sigma_j)_{j\in\mathcal{B}_i})$ for $\sigma_j := (\kappa_j, \gamma_j)$, and outputs Λ_i; then, send the leakage query $(\hat{g}_1, \ldots, \hat{g}_m)$ to the target leakage oracle and forward the answer to A.
3. **Tampering queries.** Upon input a tampering query $(\mathcal{T}, (f_1, \ldots, f_m))$, execute the following steps.
   (a) *Obtain the tampered ciphertext.* For all $i \in [m]$, construct the function $\hat{h}_i$ which takes as input the shares $(\kappa_j)_{j\in\mathcal{B}_i}$, computes the tampering $(\tilde{\kappa}_j, \tilde{\gamma}_j)_{j\in\mathcal{B}_i} = f_i((\kappa_j, \gamma_j)_{j\in\mathcal{B}_i})$ and outputs either ε if $\mathcal{B}_i \cap \mathcal{T} = \emptyset$, or $\tilde{\gamma} = \mathsf{IRec}((\tilde{\gamma}_j)_{j\in\mathcal{B}_i\cap\mathcal{T}})$ otherwise; note that this function is well defined thanks to the additional restriction on tampering queries stated in the theorem (in particular, in case $\mathcal{B}_i \cap \mathcal{T} \neq \emptyset$, this set contains at least t^* elements, which is enough for running IRec). Run procedure $\mathsf{LeakTamper}_{\mathcal{T}}(\hat{h}_1, \ldots, \hat{h}_m)$ to obtain either the tampered ciphertext $\tilde{\gamma}$ or $\perp$.
   (b) *Check that the ciphertext is consistent.* For all $i \in [m]$, construct the function $\hat{h}'_i$ which also hard-wires the value $\tilde{\gamma}$, takes as input the shares $(\kappa_j)_{j\in\mathcal{B}_i}$, computes $(\hat{\gamma}_1, \ldots \hat{\gamma}_n) = \mathsf{IDisp}(\tilde{\gamma})$ and $(\tilde{\kappa}_j, \tilde{\gamma}_j)_{j\in\mathcal{B}_i} = f_i((\kappa_j, \gamma_j)_{j\in\mathcal{B}_i})$ and, for all $j \in \mathcal{B}_i \cap \mathcal{T}$, checks that $\tilde{\gamma}_j = \hat{\gamma}_j$, outputting 0 if the check fails (and 1 otherwise). Run procedure $\mathsf{CheckEq}_{\mathcal{T}}(1, \hat{h}'_1, \ldots, \hat{h}'_m)$ to obtain either $\diamond$ (in case the tampered ciphertext $\tilde{\gamma}$ is consistent with the tampered shares of the information dispersal) or $\perp$.
   (c) *Obtain the tampered key.* For all $i \in [m]$, construct the tampering function $\hat{f}_i$ which takes as input all the shares $(\kappa_j)_{j\in\mathcal{B}_i}$, computes the tampered shares $(\tilde{\kappa}_j, \tilde{\gamma}_j)_{j\in\mathcal{B}_i} = f_i((\kappa_j, \gamma_j)_{j\in\mathcal{B}_i})$ and outputs $(\tilde{\kappa}_j)_{j\in\mathcal{B}_i}$. Send $(\mathcal{T}, (\hat{f}_1, \ldots, \hat{f}_m))$ to the tampering oracle, obtaining a tampered key $\tilde{\kappa} \in \mathcal{K} \cup \{\diamond, \perp\}$.
   (d) *Obtain the tampered message.* If any of the previous steps resulted in a $\perp$, output $\perp$ and self-destruct; otherwise, compute $\tilde{\mu} = \mathsf{Dec}(\tilde{\kappa}, \tilde{\gamma})$, replacing $\tilde{\kappa}$ with κ if $\tilde{\kappa} = \diamond$, and return $\tilde{\mu}$ to the adversary (and self-destruct if $\tilde{\mu} = \perp$).
4. **Guess.** Output the same distinguishing bit as A does.

For the analysis, recall that A has the restriction on the tampering queries and, in particular, for each tampering query $(\mathcal{T}, f)$ and each $i \in [m]$ such that $\mathcal{B}_i \cap \mathcal{T} \neq \emptyset$, we have that $|\mathcal{B}_i \cap \mathcal{T}| \geq t^*$, hence the ciphertext can be reconstructed from all the shares in $\mathcal{B}_i \cap \mathcal{T}$. In particular, this allows the reduction to obtain a candidate tampered ciphertext from each subset of tampered shares; moreover, once obtained a possible tampered ciphertext, the reduction checks that all the shares of the ciphertext are consistent and finally obtains the tampered key with which obtains the tampered message.

Consider the following event **Bad**, defined over the probability space of the original experiment: The event becomes true if the attacker ever submits a query to the tampering oracle that triggers the condition $\tilde{\kappa} = \hat{\kappa}$, where the key $\hat{\kappa}$ is sampled at the beginning of the experiment. It is easy to see that, conditioning on $\overline{\mathbf{Bad}}$, the above reduction perfectly emulates the view of the adversary in experiment $\mathbf{LR\text{-}CNMSS}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A}}(\lambda, b)$ when $(\kappa_1, \ldots, \kappa_n)$ is a secret sharing of κ, and perfectly emulates the view of the adversary in $\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A}}(\lambda, b)$ when $(\kappa_1, \ldots, \kappa_n)$ is a secret sharing of $\hat{\kappa}$. Since, using a union bound, $\Pr[\mathbf{Bad}] \leq \mathsf{poly}(\lambda) \cdot 2^{-\lambda} = \mathsf{negl}(\lambda)$, by a standard argument the two experiments are computationally indistinguishable. The lemma follows. □

The lemma below concludes the proof of continuous non-malleability in Theorem 3.

Lemma 6. *For all $\mu_0, \mu_1 \in \mathcal{M}$, and all $(t-1)$-sized partitions $\mathcal{B}$ of $[n]$, it holds that:*

$$\left\{\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, 0)\right\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \left\{\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, 1)\right\}_{\lambda\in\mathbb{N}}.$$

Proof. By reduction to IND-CCA security of the symmetric encryption scheme. Suppose that there exist two messages $\mu_0, \mu_1 \in \mathcal{M}$, a $(t-1)$-sized partition $\mathcal{B}$ of $[n]$, and a PPT adversary A that is able to distinguish between the two experiments with non-negligible probability. Consider the following reduction A' attacking IND-CCA security of Σ.

1. **Setup.** Set the challenge messages to be μ_0 and μ_1, obtain the challenge ciphertext γ, sample a key $\hat{\kappa} \leftarrow\!\!\$\, \mathcal{K}$ and compute $(\gamma_1, \ldots, \gamma_n) = \mathsf{IDisp}(\hat{\gamma})$, and $(\kappa_1, \ldots, \kappa_n) \leftarrow\!\!\$\, \mathsf{Share}(\hat{\kappa})$. Finally, for all $i \in [n]$, construct the share $\sigma_i^* := (\kappa_i, \gamma_i)$.
2. **Leakage queries.** Answer leakage queries as in the hybrid experiment.
3. **Tampering queries.** Upon input a tampering query $(\mathcal{T}, (f_1, \ldots, f_m))$, for all $i \in [m]$, compute $(\tilde{\sigma}_j)_{j\in\mathcal{B}_i} = f_i((\sigma_j)_{j\in\mathcal{B}_i})$, perform the consistency checks on the tampered ciphertext $\tilde{\gamma}$ (and output $\perp$ if any of these checks fails), and then reconstruct the tampered key $\tilde{\kappa} \in \mathcal{K}$. If $\tilde{\kappa} = \hat{\kappa}$, obtain the tampered message $\tilde{\mu} \in \mathcal{M} \cup \{\perp\}$ by sending $\tilde{\gamma}$ to the decryption oracle; otherwise, compute $\tilde{\mu} = \mathsf{Dec}(\tilde{\kappa}, \tilde{\gamma})$. If $\tilde{\mu} \in \{\mu_0, \mu_1\}$, set $\tilde{\mu} = \diamond$. Finally, return $\tilde{\mu}$ to A (and self-destruct if $\tilde{\mu} = \perp$).
4. **Guess.** Output the same distinguishing bit as A does.

For the analysis, note that the reduction is perfect and, in particular, for $b \in \{0, 1\}$, it perfectly simulates $\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, b)$ whenever the challenge ciphertext γ is an encryption of μ_b. This concludes the proof. □

It only remains to discuss the rate of the construction. Towards this, note that the length of the key κ for the SKE scheme Σ, and thus the size of the shares of the secret sharing scheme Π, only depends on the security parameter λ, the number of parties n and the tolerated leakage ℓ (but not on the length $|\mu|$ of the message); call $s_0(\lambda, n, \ell)$ the length of this portion of the final shares (namely, κ_i). On the other hand, it is possible to achieve length of the ciphertext $|\gamma| = |\mu| + O(\lambda)$, hence the length of each share γ_i of the information dispersal amounts to $|\gamma_i| = \frac{|\gamma|}{t^*} = \frac{|\mu|+O(\lambda)}{t^*}$. Putting it together, we have obtained:

$$s(\lambda, n, \ell, |\mu|) = s_0(\lambda, n, \ell) + \frac{|\mu| + O(\lambda)}{t^*},$$

that translates into

$$\varrho = \inf_{\lambda\in\mathbb{N}} \lim_{|\mu|\to\infty} \frac{|\mu|}{s(\lambda, n, \ell, |\mu|)}$$

$$
\begin{aligned}
&= \inf_{\lambda \in \mathbb{N}} \lim_{|\mu| \to \infty} \frac{t^* \cdot |\mu|}{|\mu| + t^* \cdot \mathsf{poly}(\lambda, n, \ell)} \\
&= t^*.
\end{aligned}
$$

This completes the proof of Theorem 3. $\square$

Rate optimality. We stress that when $k = t - 1$, Theorem 2 says that the capacity of continuously non-malleable secret sharing against joint tampering with at most $t - 1$ shares is 1. This is not in contrast with the fact that our rate compiler from Theorem 3 achieves rate larger than 1, as the latter only holds under an additional restriction on the way the attacker can manipulate the shares. Nevertheless, it is possible to adapt the proof of Theorem 2 in order to show that our rate compiler achieves the best possible rate whenever $t^* < t/2$.

Theorem 4. *Let Π be a t-out-of-n $(t-1)$-CNMSS scheme under the restriction that each tampering query $(\mathcal{T}, f)$ output by the attacker must be such that, for all subsets $\mathcal{B}_i$ of the partition $\mathcal{B}$, either $\mathcal{B}_i \cap \mathcal{T} = \emptyset$ or $|\mathcal{B}_i \cap \mathcal{T}| \geq t^*$. If $t^* \leq t/2$, then Π cannot achieve better rate than $\varrho \leq t^*$.*

Proof. The proof is almost identical to that of Theorem 2, and thus we only highlight the main differences. We change the definition of Com so that it now outputs the value $\gamma = (\sigma_1, \ldots, \sigma_{t^*})$, and we adjust the opening procedure accordingly. Hence, the goal is to prove, again, that Com is perfectly binding, so that the rate of Π must satisfy $\varrho \leq t^*$.

The reduction is identical to that in the proof of Theorem 2, except that now we define $\ell := |\sigma_{t^*+1}|| \cdots ||\sigma_t|$ and moreover the adversary attacking continuous non-malleability sets $\mathcal{B}_1 := [t^*]$ and $\mathcal{B}_2 := [t] \setminus [t^*]$ in step 3, and parses the string $\alpha_1, \ldots, \alpha_\ell$ as $\sigma_{t^*+1}, \ldots, \sigma_t$ in step 5. Note that $|\mathcal{B}_1| = t^*$ and $|\mathcal{B}_2| = t - t^* \geq 2t^* - t^* = t^*$. Since $t^* \leq t - 1$, the adversary is admissible which concludes the proof. $\square$

Remark 2. *More generally, Theorem 4 holds for t-out-of-n k-CNMSS so long as $k \geq t - t^*$.*

6 Instantiations

In this section, we show how to instantiate the building blocks required by the abstract constructions of Theorem 1 and Theorem 3.

6.1 Leakage-Resilient p-time Non-Malleable Code

Here, we explain how to obtain noisy-leakage-resilient p-time non-malleable asymmetric split-state codes with the additional properties stated in Theorem 1. Our construction exploits leakage-resilient asymmetric split-state codes as defined in §3.2, as recently introduced by Ball, Guo, and Wichs [BGW19] and generalized to the noisy-leakage setting by Brian, Faonio and Venturi [BFV19].

Let $\Pi = (\mathsf{Share}, \mathsf{Rec})$, $\Pi_\mathsf{L} = (\mathsf{Share}_\mathsf{L}, \mathsf{Rec}_\mathsf{L})$ and $\Pi_\mathsf{R} = (\mathsf{Share}_\mathsf{R}, \mathsf{Rec}_\mathsf{R})$ be split-state codes. Consider the following split-state code $\Pi^* = (\mathsf{Share}^*, \mathsf{Rec}^*)$.

- **Algorithm** Share^***:** upon input a message μ, compute $(\sigma_\mathsf{L}, \sigma_\mathsf{R}) \leftarrow\$ \mathsf{Share}(\mu)$, and $(\sigma_{\mathsf{L},\mathsf{L}}, \sigma_{\mathsf{L},\mathsf{R}}) \leftarrow\$ \mathsf{Share}_\mathsf{L}(\sigma_\mathsf{L})$ and $(\sigma_{\mathsf{R},\mathsf{L}}, \sigma_{\mathsf{R},\mathsf{R}}) \leftarrow\$ \mathsf{Share}_\mathsf{R}(\sigma_\mathsf{R})$. Set $\sigma^*_\mathsf{L} = (\sigma_{\mathsf{L},\mathsf{L}}, \sigma_{\mathsf{R},\mathsf{R}})$ and $\sigma^*_\mathsf{R} = (\sigma_{\mathsf{R},\mathsf{L}}, \sigma_{\mathsf{L},\mathsf{R}})$, and output $\sigma^*_\mathsf{L}, \sigma^*_\mathsf{R}$.
- **Algorithm** Rec^***:** upon input two shares $(\sigma^*_\mathsf{L}, \sigma^*_\mathsf{R})$, parse $\sigma^*_\mathsf{L} = (\sigma_{\mathsf{L},\mathsf{L}}, \sigma_{\mathsf{R},\mathsf{R}})$ and $\sigma^*_\mathsf{R} = (\sigma_{\mathsf{R},\mathsf{L}}, \sigma_{\mathsf{L},\mathsf{R}})$, compute the shares $\sigma_\mathsf{L} = \mathsf{Rec}_\mathsf{L}(\sigma_{\mathsf{L},\mathsf{L}}, \sigma_{\mathsf{L},\mathsf{R}})$ and $\sigma_\mathsf{R} = \mathsf{Rec}_\mathsf{R}(\sigma_{\mathsf{R},\mathsf{L}}, \sigma_{\mathsf{R},\mathsf{R}})$, and output $\mu = \mathsf{Rec}(\sigma_\mathsf{L}, \sigma_\mathsf{R})$.

Theorem 5. *For all $i, j \in \{\mathsf{L}, \mathsf{R}\}$, let $p, s_i, s_{i,j} \in \mathbb{N}$, $\ell_i, \ell_{i,j} \geq 0$, and $\epsilon, \epsilon_i \in [0,1]$ be parameters such that:*

- $s_{\mathsf{R}} < s_{\mathsf{L}}$;
- $s_{\mathsf{L,R}} < s_{\mathsf{L,L}}$, $\ell_{\mathsf{L,L}} \geq \ell_{\mathsf{L}} + p \cdot s_{\mathsf{R,R}}$ *and* $\ell_{\mathsf{L,R}} \geq \ell_{\mathsf{R}}$;
- $s_{\mathsf{R,R}} < s_{\mathsf{R,L}}$, $\ell_{\mathsf{R,L}} \geq \ell_{\mathsf{R}} + p \cdot s_{\mathsf{L,R}}$ *and* $\ell_{\mathsf{R,R}} \geq \ell_{\mathsf{L}}$.

Assume that:

- Π *is an* $(s_{\mathsf{L}}, s_{\mathsf{R}})$*-asymmetric* (p, ϵ)*-NMC;*
- Π_{L} *is an* $(s_{\mathsf{L,L}}, s_{\mathsf{L,R}})$*-asymmetric* $(\ell_{\mathsf{L,L}}, \ell_{\mathsf{L,R}}, \epsilon_{\mathsf{L}})$*-LRC;*
- Π_{R} *is an* $(s_{\mathsf{R,L}}, s_{\mathsf{R,R}})$*-asymmetric* $(\ell_{\mathsf{R,L}}, \ell_{\mathsf{R,R}}, \epsilon_{\mathsf{R}})$*-LRC.*

Then, Π^* *is an* $(s^*_{\mathsf{L}}, s^*_{\mathsf{R}})$*-asymmetric* $(\ell_{\mathsf{L}}, \ell_{\mathsf{R}}, p, \epsilon + 2(\epsilon_{\mathsf{L}} + \epsilon_{\mathsf{R}}))$*-LR-NMC, where* $s^*_{\mathsf{L}} = s_{\mathsf{L,L}} + s_{\mathsf{R,R}}$ *and* $s^*_{\mathsf{R}} = s_{\mathsf{L,R}} + s_{\mathsf{R,L}}$.

The proof to the above theorem goes along the same lines of the proof of Theorem 7 in [BFV19] for the case of 2-out-of-2 secret sharing. The only difference is that Π is a p-time NMC instead of a one-time NMC, and we use different parameters for Π_{L} and Π_{R}. In particular, all the hybrid experiments are the same as in [BFV19] with the only difference that we have to leak $2p$ tampered values (namely, $\tilde{\sigma}^{(j)}_{\mathsf{R,R}}, \tilde{\sigma}^{(j)}_{\mathsf{L,R}}$ for $j \in [p]$) instead of only two; however, our choice of the leakage parameters allows us to do so, since

$$\ell_{\mathsf{L,L}} \geq \ell_{\mathsf{L}} + p \cdot s_{\mathsf{R,R}} \qquad \text{and} \qquad \ell_{\mathsf{R,L}} \geq \ell_{\mathsf{R}} + p \cdot s_{\mathsf{L,R}}.$$

Proof. We consider the same hybrids as in the proof of [BFV19, Theorem 7] and focus here only on the reduction to non-malleability, thus showing that the last hybrid experiment with bit $b = 0$ is statistically close to the same experiment with bit $b = 1$. Following the proof in [BFV19], we consider the hybrid experiments below:

- **Hybrid 1.** Same as the security experiment but before applying the tampering queries we re-sample the share $\sigma'_{\mathsf{L,L}}$ (resp. $\sigma'_{\mathsf{R,L}}$) in such a way that: (1) the reconstruction with $\sigma'_{\mathsf{L,L}}, \sigma_{\mathsf{L,R}}$ (resp. $\sigma'_{\mathsf{R,L}}, \sigma_{\mathsf{R,R}}$) leads to the value σ_{L} (resp. σ_{R}), (2) it is consistent with the leakage performed, (3) for any j the j-th tampering query applied to $(\sigma'_{\mathsf{L,L}}, \sigma_{\mathsf{R,R}})$ (resp. $(\sigma'_{\mathsf{R,L}}, \sigma_{\mathsf{L,R}})$) produces the same tampered value $\tilde{\sigma}^{(j)}_{\mathsf{R,R}}$ (resp. $\tilde{\sigma}^{(j)}_{\mathsf{L,R}}$). This hybrid is equivalent to the security experiment because we re-sample the values from the same distributions where the original values are defined.
- **Hybrid 2.** Apply all the leakage queries to the shares $(\hat{\sigma}_{\mathsf{L,R}}, \sigma_{\mathsf{R,R}})$ and $(\hat{\sigma}_{\mathsf{R,L}}, \sigma_{\mathsf{L,R}})$ where $\hat{\sigma}_{\mathsf{L,L}}, \sigma_{\mathsf{L,R}}$ and $\hat{\sigma}_{\mathsf{R,L}}, \sigma_{\mathsf{L,R}}$ are valid shares of dummy messages of appropriate size. Here, we can reduce by two consecutive steps to the noisy-leakage resilience of Π_{L} and Π_{R}.

Thus, in the last hybrid experiment, the leakage is computed using fake shares $\hat{\sigma}_{\mathsf{L,L}}$ and $\hat{\sigma}_{\mathsf{R,L}}$, and so are the tampered values $\tilde{\sigma}^{(j)}_{\mathsf{R,R}}, \tilde{\sigma}^{(j)}_{\mathsf{L,R}}$ for $j \in [p]$. Consider the following reduction to an adversary A distinguishing between the last hybrid experiment with $b = 0$ and $b = 1$.

1. Sample $\sigma_{\mathsf{L,R}}$, and $\sigma_{\mathsf{R,R}}$ at random from their respective domains.
2. Sample $\hat{\sigma}_{\mathsf{L,L}} \leftarrow\!\!\$\ \overline{\mathsf{Share}}_{\mathsf{L}}(0^{s_{\mathsf{L}}}, \sigma_{\mathsf{L,R}})$ and $\hat{\sigma}_{\mathsf{R,L}} \leftarrow\!\!\$\ \overline{\mathsf{Share}}_{\mathsf{R}}(0^{s_{\mathsf{R}}}, \sigma_{\mathsf{R,R}})$ and redirect all the leakage queries to the codeword $(\hat{\sigma}_{\mathsf{L,L}}, \sigma_{\mathsf{R,R}})$ and $(\hat{\sigma}_{\mathsf{R,L}}, \sigma_{\mathsf{L,R}})$.
3. Upon receiving tampering queries $f^{(1)}, \dots, f^{(p)}$, do the following.
   (a) For all $j \in [p]$, compute $(\tilde{\sigma}^{(j)}_{\mathsf{L,L}}, \tilde{\sigma}^{(j)}_{\mathsf{R,R}}) = f^{(j)}_{\mathsf{L}}((\hat{\sigma}_{\mathsf{L,L}}, \sigma_{\mathsf{R,R}}))$ and $(\tilde{\sigma}^{(j)}_{\mathsf{R,L}}, \tilde{\sigma}^{(j)}_{\mathsf{L,R}}) = f^{(j)}_{\mathsf{R}}((\hat{\sigma}_{\mathsf{R,L}}, \sigma_{\mathsf{L,R}}))$;
   (b) Sample random strings $\rho_{\mathsf{L}}, \rho_{\mathsf{R}}$.
   (c) For all $j \in [p]$, send the tampering query $\bar{f}^{(j)} = (\bar{f}^{(j)}_{\mathsf{L}}, \bar{f}^{(j)}_{\mathsf{R}})$, where $\bar{f}^{(j)}_{\mathsf{L}}$ (resp. $\bar{f}^{(j)}_{\mathsf{R}}$), upon input σ_{L} (resp. σ_{R}) proceeds as follows:
      i. using randomness ρ_{L} (resp., ρ_{R}), sample the share $\sigma'_{\mathsf{L,L}}$ (resp. $\sigma'_{\mathsf{R,L}}$) such that (1) $\mathsf{Rec}_{\mathsf{L}}(\sigma'_{\mathsf{L,L}}, \sigma_{\mathsf{L,R}}) = \sigma_{\mathsf{L}}$ (resp. $\mathsf{Rec}_{\mathsf{R}}(\sigma'_{\mathsf{R,L}}, \sigma_{\mathsf{R,R}}) = \sigma_{\mathsf{R}}$) and (2) $\sigma'_{\mathsf{L,L}}$ (resp. $\sigma'_{\mathsf{R,L}}$) is

consistent with all the leakage done by A and with the tampered values $\tilde{\sigma}^{(1)}_{\mathsf{R},\mathsf{R}}, \ldots,$ $\tilde{\sigma}^{(p)}_{\mathsf{R},\mathsf{R}}$ (resp. $\tilde{\sigma}^{(1)}_{\mathsf{L},\mathsf{R}}, \ldots, \tilde{\sigma}^{(p)}_{\mathsf{L},\mathsf{R}}$);

ii. apply the tampering query $f^{(j)}_{\mathsf{L}}$ (resp. $f^{(j)}_{\mathsf{R}}$) to $(\sigma'_{\mathsf{L},\mathsf{L}}, \sigma_{\mathsf{R},\mathsf{R}})$ (resp. $\sigma'_{\mathsf{R},\mathsf{L}}, \sigma_{\mathsf{L},\mathsf{R}}$) obtaining the tampered shares $(\tilde{\sigma}'_{\mathsf{L},\mathsf{L}}, \tilde{\sigma}^{(j)}_{\mathsf{R},\mathsf{R}})$ (resp. $(\tilde{\sigma}'_{\mathsf{R},\mathsf{L}}, \tilde{\sigma}^{(j)}_{\mathsf{L},\mathsf{R}})$);

iii. output $\tilde{\sigma}_{\mathsf{L}} = \mathsf{Rec}_{\mathsf{L}}(\tilde{\sigma}'_{\mathsf{L},\mathsf{L}}, \tilde{\sigma}^{(j)}_{\mathsf{R},\mathsf{R}})$ (resp. $\tilde{\sigma}_{\mathsf{R}} = \mathsf{Rec}_{\mathsf{R}}(\tilde{\sigma}'_{\mathsf{R},\mathsf{L}}, \tilde{\sigma}^{(j)}_{\mathsf{L},\mathsf{R}})$).

For the analysis, note that the reduction is perfect and, in particular, samples a new valid codeword that is consistent with the view of the adversary A and encodes the message μ_b as in the real experiment. This finishes the proof. □

Finally, we show that the scheme Π^* of Theorem 5 is able to achieve the properties (i)-(ii) needed to instantiate Theorem 1. The lemma below states that if the underlying NMC Π satisfies the additional property (i), so does the scheme Π^*.

Lemma 7. *Suppose that there exists σ_{L} such that, for all $\mu \in \mathcal{M}$, there exists σ_{R} such that $\mathsf{Rec}(\sigma_{\mathsf{L}}, \sigma_{\mathsf{R}}) = \mu$. Then, there exists σ^*_{L} such that, for all $\mu \in \mathcal{M}$, there exists σ^*_{R} such that $\mathsf{Rec}^*(\sigma^*_{\mathsf{L}}, \sigma^*_{\mathsf{R}}) = \mu$.*

Proof. Let σ_{L} be such that, for all $\mu \in \mathcal{M}$, there exists σ_{R} such that $\mathsf{Rec}(\sigma_{\mathsf{L}}, \sigma_{\mathsf{R}}) = \mu$. Then, we can fix $\sigma_{\mathsf{R},\mathsf{R}}$ and $\sigma_{\mathsf{L},\mathsf{R}}$ and compute $\sigma_{\mathsf{L},\mathsf{L}} \leftarrow\$ \overline{\mathsf{Share}}_{\mathsf{L}}(\sigma_{\mathsf{L}}, \sigma_{\mathsf{L},\mathsf{R}})$. The new left share will be $\sigma^*_{\mathsf{L}} = (\sigma_{\mathsf{L},\mathsf{L}}, \sigma_{\mathsf{R},\mathsf{R}})$ and, once fixed σ^*_{L} and $\mu \in \mathcal{M}$, in order to obtain the right share it suffices to compute $\sigma_{\mathsf{R},\mathsf{L}} \leftarrow\$ \overline{\mathsf{Share}}_{\mathsf{R}}(\sigma_{\mathsf{R}}, \sigma_{\mathsf{R},\mathsf{R}})$ and set $\sigma^*_{\mathsf{R}} = (\sigma_{\mathsf{R},\mathsf{L}}, \sigma_{\mathsf{L},\mathsf{R}})$. □

The property (ii) is a bit more delicate because, even if $\Pi_{\mathsf{L}}, \Pi_{\mathsf{R}}$ achieve it, the random variables $(\mathbf{\Sigma}_{\mathsf{L},\mathsf{L}}, \mathbf{\Sigma}_{\mathsf{R},\mathsf{R}})$ and $(\mathbf{\Sigma}_{\mathsf{R},\mathsf{L}}, \mathbf{\Sigma}_{\mathsf{L},\mathsf{R}})$ are defined by $(\mathbf{\Sigma}_{\mathsf{L},\mathsf{L}}, \mathbf{\Sigma}_{\mathsf{L},\mathsf{R}}) = \mathsf{Share}_{\mathsf{L}}(\mathbf{\Sigma}_{\mathsf{L}})$ and $(\mathbf{\Sigma}_{\mathsf{R},\mathsf{L}}, \mathbf{\Sigma}_{\mathsf{R},\mathsf{R}}) = \mathsf{Share}_{\mathsf{R}}(\mathbf{\Sigma}_{\mathsf{R}})$, and $\mathbf{\Sigma}_{\mathsf{L}}$ and $\mathbf{\Sigma}_{\mathsf{R}}$ are related distributions. Instead, here we use a non-blackbox approach and prove that the asymmetric code given by Appendix A of [BFV19], which we describe below, allows Π^* to achieve property (ii).

Let Ext be a seeded extractor with d-bits source, r-bit seed and m-bit output and let $\mathsf{2Ext}$ be a two-source extractor with s_2-bits sources and r-bit output. Consider the following secret sharing scheme Π_{LRC} with message space $\mathcal{M} = \{0,1\}^m$ and shares space $\mathcal{S} = \{0,1\}^{s_1} \times \{0,1\}^{s_2}$:

- **Algorithm Share:** upon input the message μ, randomly sample $\sigma_2 \leftarrow\$ \{0,1\}^{s_2}, x \leftarrow\$ \{0,1\}^d, y \leftarrow\$ \{0,1\}^{s_2}$, compute $\rho := \mathsf{2Ext}(\sigma_2, y)$ and $z := \mathsf{Ext}(x, \rho) \oplus \mu$ and finally output (σ_1, σ_2), where $\sigma_1 = (x, y, z)$.
- **Algorithm Rec:** upon input the shares (σ_1, σ_2), parse $\sigma_1 = (x, y, z)$ and output $\mu := z \oplus \mathsf{Ext}(x, \mathsf{2Ext}(\sigma_1, y))$.

For all $\epsilon, \ell_1, \ell_2 \geq 0$ there exists an appropriate choice of the parameters d and r such that the above is an $(\ell_1, \ell_2, \epsilon)$-LRC (see [BGW19, BFV19] for the details) and, moreover, the above admits the following alternative sharing algorithm $\overline{\mathsf{Share}}$:

- **Algorithm $\overline{\mathsf{Share}}$:** upon input the message μ and the value σ_2, randomly sample $x \leftarrow\$ \{0,1\}^d, y \leftarrow\$ \{0,1\}^{s_2}$, compute $\rho := \mathsf{2Ext}(\sigma_2, y)$ and $z := \mathsf{Ext}(x, \rho) \oplus \mu$ and finally output (σ_1, σ_2), where $\sigma_1 = (x, y, z)$.

The following lemma proves that the above scheme allows our construction Π^* to achieve property (ii) of Theorem 1.

Lemma 8. *Instantiating Π_{L} and Π_{R} with the asymmetric LRC Π_{LRC}, for all $\mu \in \mathcal{M}$ it holds that*

$$\widetilde{\mathbb{H}}_\infty(\mathbf{\Sigma}^*_{\mathsf{L}} \mid \mathbf{\Sigma}^*_{\mathsf{R}}) \geq \mathbb{H}_\infty(\mathbf{\Sigma}^*_{\mathsf{L}}) - d \qquad \widetilde{\mathbb{H}}_\infty(\mathbf{\Sigma}^*_{\mathsf{R}} \mid \mathbf{\Sigma}^*_{\mathsf{L}}) \geq \mathbb{H}_\infty(\mathbf{\Sigma}^*_{\mathsf{R}}) - d,$$

where $d = s_{\mathsf{L}} + s_{\mathsf{R}}$ and $(\mathbf{\Sigma}^*_{\mathsf{L}}, \mathbf{\Sigma}^*_{\mathsf{R}}) = \mathsf{Share}^*(\mu)$ is the distribution of the shares of μ using the scheme Π^*.

Proof. For any message $\mu \in \mathcal{M}$, let $(\mathbf{\Sigma}_\mathsf{L}, \mathbf{\Sigma}_\mathsf{R}) = \mathsf{Share}(\mu)$ be the random variable relative to a NMC of μ and let $(\mathbf{\Sigma}_{\mathsf{L,L}}, \mathbf{\Sigma}_{\mathsf{L,R}}) = \mathsf{Share}_\mathsf{L}(\mathbf{\Sigma}_\mathsf{L}), (\mathbf{\Sigma}_{\mathsf{R,L}}, \mathbf{\Sigma}_{\mathsf{R,R}}) = \mathsf{Share}_\mathsf{R}(\mathbf{\Sigma}_\mathsf{R})$ the respective random variables relative to the left and right leakage-resilient encodings; in particular, let $(\mathbf{X}_\mathsf{L}, \mathbf{Y}_\mathsf{L}, \mathbf{Z}_\mathsf{L}) = \mathbf{\Sigma}_{\mathsf{L,L}}$ and $(\mathbf{X}_\mathsf{R}, \mathbf{Y}_\mathsf{R}, \mathbf{Z}_\mathsf{R}) = \mathbf{\Sigma}_{\mathsf{R,L}}$ the random variables relative to the values x, y, z of the asymmetric LRC Π_{LRC} in the left and in the right instantiation respectively. Moreover, let $d_\mathsf{L}, d_\mathsf{R}$ the parameter d in Π_L and Π_R respectively. Then,

$$
\begin{aligned}
\widetilde{\mathbb{H}}_\infty(\mathbf{\Sigma}^*_\mathsf{L} \,|\, \mathbf{\Sigma}^*_\mathsf{R}) &= \widetilde{\mathbb{H}}_\infty\left((\mathbf{X}_\mathsf{L}, \mathbf{Y}_\mathsf{L}, \mathbf{Z}_\mathsf{L}), \mathbf{\Sigma}_{\mathsf{R,R}} \,|\, (\mathbf{X}_\mathsf{R}, \mathbf{Y}_\mathsf{R}, \mathbf{Z}_\mathsf{R}), \mathbf{\Sigma}_{\mathsf{L,R}}\right) \\
&\geq \widetilde{\mathbb{H}}_\infty\left(\mathbf{X}_\mathsf{L}, \mathbf{Y}_\mathsf{L}, \mathbf{Z}_\mathsf{L}, \mathbf{\Sigma}_{\mathsf{R,R}} \,|\, \mathbf{X}_\mathsf{R}, \mathbf{Y}_\mathsf{R}, \mathbf{\Sigma}_{\mathsf{L,R}}\right) - |\mathbf{Z}_\mathsf{R}| \qquad (19)\\
&\geq \widetilde{\mathbb{H}}_\infty\left(\mathbf{X}_\mathsf{L}, \mathbf{Y}_\mathsf{L}, \mathbf{\Sigma}_{\mathsf{R,R}} \,|\, \mathbf{X}_\mathsf{R}, \mathbf{Y}_\mathsf{R}, \mathbf{\Sigma}_{\mathsf{L,R}}\right) - |\mathbf{Z}_\mathsf{R}| \qquad (20)\\
&\geq \mathbb{H}_\infty\left(\mathbf{X}_\mathsf{L}, \mathbf{Y}_\mathsf{L}, \mathbf{\Sigma}_{\mathsf{R,R}}\right) - |\mathbf{Z}_\mathsf{R}| \qquad (21)\\
&= s_{\mathsf{L,L}} - s_\mathsf{L} + s_{\mathsf{R,R}} - s_\mathsf{R} \qquad (22)\\
&= \mathbb{H}_\infty(\mathbf{\Sigma}^*_\mathsf{L}) - s_\mathsf{L} - s_\mathsf{R}, \qquad (23)
\end{aligned}
$$

where Eq. (19) follows from the application of Lemma 1, in Eq. (20) we simply removed the random variable $\mathbf{Z}_\mathsf{L}$, Eq. (21) holds because now the random variables $\mathbf{X}_\mathsf{L}, \mathbf{Y}_\mathsf{L}, \mathbf{\Sigma}_{\mathsf{R,R}}$ are independent of $\mathbf{X}_\mathsf{R}, \mathbf{Y}_\mathsf{R}, \mathbf{\Sigma}_{\mathsf{L,R}}$, Eq. (22) follows from the fact that $x_\mathsf{L}, y_\mathsf{L}, \sigma_{\mathsf{R,R}}$ are randomly sampled and that $|(x_\mathsf{L}, y_\mathsf{L})| = \sigma_{\mathsf{L,L}} - |z_\mathsf{L}|$, where $|z_\mathsf{L}| = \sigma_\mathsf{L}$ and Eq. (23) follows from the fact that $\mathbf{\Sigma}^*_\mathsf{L} = (\mathbf{\Sigma}_{\mathsf{L,L}}, \mathbf{\Sigma}_{\mathsf{R,R}})$ is uniformly distributed over $\{0,1\}^{s_{\mathsf{L,L}}} \times \{0,1\}^{s_{\mathsf{R,R}}}$. A similar analysis shows that

$$
\begin{aligned}
\widetilde{\mathbb{H}}_\infty(\mathbf{\Sigma}^*_\mathsf{R} \,|\, \mathbf{\Sigma}^*_\mathsf{L}) &= \widetilde{\mathbb{H}}_\infty\left((\mathbf{X}_\mathsf{R}, \mathbf{Y}_\mathsf{R}, \mathbf{Z}_\mathsf{R}), \mathbf{\Sigma}_{\mathsf{L,R}} \,|\, (\mathbf{X}_\mathsf{L}, \mathbf{Y}_\mathsf{L}, \mathbf{Z}_\mathsf{L}), \mathbf{\Sigma}_{\mathsf{R,R}}\right) \\
&\geq \widetilde{\mathbb{H}}_\infty\left(\mathbf{X}_\mathsf{R}, \mathbf{Y}_\mathsf{R}, \mathbf{Z}_\mathsf{R}, \mathbf{\Sigma}_{\mathsf{L,R}} \,|\, \mathbf{X}_\mathsf{L}, \mathbf{Y}_\mathsf{L}, \mathbf{\Sigma}_{\mathsf{R,R}}\right) - |\mathbf{Z}_\mathsf{L}| \\
&\geq \widetilde{\mathbb{H}}_\infty\left(\mathbf{X}_\mathsf{R}, \mathbf{Y}_\mathsf{R}, \mathbf{\Sigma}_{\mathsf{L,R}} \,|\, \mathbf{X}_\mathsf{L}, \mathbf{Y}_\mathsf{L}, \mathbf{\Sigma}_{\mathsf{R,R}}\right) - |\mathbf{Z}_\mathsf{L}| \\
&\geq \mathbb{H}_\infty\left(\mathbf{X}_\mathsf{R}, \mathbf{Y}_\mathsf{R}, \mathbf{\Sigma}_{\mathsf{L,R}}\right) - |\mathbf{Z}_\mathsf{L}| \\
&= s_{\mathsf{R,L}} - s_\mathsf{R} + s_{\mathsf{L,R}} - s_\mathsf{L} \\
&= \mathbb{H}_\infty(\mathbf{\Sigma}^*_\mathsf{R}) - s_\mathsf{R} - s_\mathsf{L}.
\end{aligned}
$$

□

Corollary 5.7 of [GSZ20] shows that, for all $n_1, n_2 \in \mathbb{N}$ and all polynomials p', there exists a two-source p-time ϵ-non-malleable extractor for sources of full-entropy of size n_1, n_2, where $p = n_2^{\Omega(1)}$, $n_1 = 4n_2 + p'(n_2)$ and $\epsilon = 2^{-n_2^{\Omega(1)}}$. This scheme has efficient pre-image sampleability and further satisfies the additional property described in the hypothesis of Lemma 7. By the known connection between (leakage-resilient) non-malleable extractors with efficient pre-image sampleability and (leakage-resilient) non-malleable codes, we obtain a $(p, \epsilon \cdot 2^{p|\mu|+1})$-NMC. Additionally, we note that by our setting of the parameters in Theorem 5 we can have $\ell_\mathsf{L} \geq s^*_\mathsf{R}$ so long as the underlying schemes Π_L and Π_R allow to arbitrarily set the parameters of leakage and of the codeword size of the left shares and right shares, which is the case thanks to Theorem 6 of [BFV19].

Hence, together with Lemma 7 and Lemma 8, we have obtained the following corollary:

Corollary 1. *For any $s_\mathsf{L}, s_\mathsf{R}, \ell_\mathsf{L}, \ell_\mathsf{R}, p \in \mathbb{N}, \epsilon \in [0,1]$, there is a construction of an $(s_\mathsf{L}, s_\mathsf{R})$-asymmetric $(\ell_\mathsf{L}, \ell_\mathsf{R})$-noisy leakage-resilient p-time ϵ-non-malleable code satisfying the additional properties stated in Theorem 1.*

6.2 Leakage-Resilient Continuously Non-Malleable Secret Sharing

By instantiating Theorem 1, we obtain the following.

Corollary 2. *Assuming the existence of one-to-one one-way functions, for any $n, t, \ell \in \mathbb{N}$ with $t > 2n/3$, there is a construction of a t-out-of-n secret sharing scheme satisfying noisy-leakage resilient continuous non-malleability under selective k-joint leakage and tampering attacks, where $k = t - 1$.*

Proof. The proof follows by instantiating the inner non-malleable code using Corollary 1 and recalling that perfectly binding and computationally hiding commitment schemes can be instantiated from one-to-one one-way functions [GMW87]. □

Furthermore, by instantiating Theorem 3 with $t^* = 1$, we obtain the following.

Corollary 3. *Assuming the existence of one-to-one one-way functions, for any $n, t, \ell \in \mathbb{N}$ with $t > 2n/3$, there is a construction of a t-out-of-n secret sharing scheme satisfying noisy leakage-resilient continuous non-malleability under selective k-joint leakage and tampering attacks; moreover, the scheme achieves asymptotic rate 1, which is optimal.*

Proof. It is well known that IND-CCA secure SKE schemes can be constructed in a black-box way from any OWF, whereas the information dispersal can be instantiated using linear algebra over finite fields [Rab89]; as for the continuously non-malleable secret sharing scheme we can take the one given by Corollary 2. Finally, when applying Theorem 3 with $t^* = 1$, the restriction on the tampering queries disappears (any subset either contains at least $t^* = 1$ share in $\mathcal{T}$ or does not contain any share in $\mathcal{T}$) and we obtain the standard definition of continuous non-malleability against $(t-1)$-joint tampering attacks; since $t^* = 1$, the asymptotic rate[11] of the construction is one, which, by Theorem 2, is optimal. □

6.3 Breaking the Rate-One Barrier

Finally, Theorem 3 also allows to obtain the first non-malleable secret sharing scheme against independent tampering attacks with rate larger than one.

Corollary 4. *Assuming the existence of one-to-one one-way functions, for any $n, t \in \mathbb{N}$ with $t > 2n/3$, there is a construction of a t-out-of-n secret sharing scheme satisfying one-time non-malleability under independent tampering attacks; moreover, the scheme achieves asymptotic rate $t/2$.*

Proof. The construction is the same of Theorem 3 with $t^* = t/2$,[12] therefore the concrete instantiation follows by Corollary 3.

The proof of security follows by a simple reduction to non-malleability against joint tampering. In particular, assume that there exists an adversary A which is able to break one-time non-malleability by submitting an independent tampering query $(\mathcal{T}, f)$ to the tampering oracle. Then, it is possible to construct a reduction $\hat{\mathsf{A}}$ which partitions $\mathcal{T}$ into two subsets $\mathcal{B}_1, \mathcal{B}_2$ of $t/2$ shares each, runs A, forwards the tampering query $(\mathcal{T}, f)$ to the tampering oracle (recall that any independent tampering query is also a k-joint tampering query for all $k \geq 1$), and finally returns the tampered message $\tilde{\mu}$ to A and outputs the same distinguishing bit of A. Clearly, the attacker $\hat{\mathsf{A}}$ perfectly simulates the view of A, and moreover the condition $|\mathcal{B}_1 \cup \mathcal{T}| = |\mathcal{B}_2 \cup \mathcal{T}| = t/2$ is satisfied. □

Remark 3. *Corollary 4 can be trivially extended to include noisy-leakage resilience. Moreover, it can also be extended to continuous non-malleability if we assume that the reconstruction set $\mathcal{T}$ is the same across all tampering queries.*

[11] For the information dispersal, it suffices to define $\mathsf{IDisp}(\mu) := (\mu, \ldots, \mu)$ (*i.e.*, the same message repeated n times) and $\mathsf{IRec}(\mu) := \mu$.

[12] For the sake of simplicity, assume t even. When t is odd, we obtain $t^* = (t-1)/2$.

Acknowledgments

We thank Mark Simkin and Maciej Obremski for their valuable comments.

References

[ADN+19] Divesh Aggarwal, Ivan Damgård, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, João Ribeiro, and Mark Simkin. Stronger leakage-resilient and non-malleable secret sharing schemes for general access structures. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 510–539. Springer, Heidelberg, August 2019. 1, 8

[BCL+20] Marshall Ball, Eshan Chattopadhyay, Jyun-Jie Liao, Tal Malkin, and Li-Yang Tan. Non-malleability against polynomial tampering. Cryptology ePrint Archive, Report 2020/147, 2020. https://ia.cr/2020/147. 8

[BFO+20] Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, and Daniele Venturi. Non-malleable secret sharing against bounded joint-tampering attacks in the plain model. In Daniele Micciancio and Thomas Ristenpart, editors, *CRYPTO 2020, Part III*, volume 12172 of *LNCS*, pages 127–155. Springer, Heidelberg, August 2020. 2, 3, 8

[BFV19] Gianluca Brian, Antonio Faonio, and Daniele Venturi. Continuously non-malleable secret sharing for general access structures. In Dennis Hofheinz and Alon Rosen, editors, *TCC 2019, Part II*, volume 11892 of *LNCS*, pages 211–232. Springer, Heidelberg, December 2019. 1, 2, 3, 4, 5, 8, 12, 26, 27, 28, 29, 41

[BGW19] Marshall Ball, Siyao Guo, and Daniel Wichs. Non-malleable codes for decision trees. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part I*, volume 11692 of *LNCS*, pages 413–434. Springer, Heidelberg, August 2019. 7, 26, 28

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. *Proceedings of AFIPS 1979 National Computer Conference*, 48:313–317, 1979. 1

[BM15] Rishiraj Bhattacharyya and Pratyay Mukherjee. Non-adaptive programmability of random oracle. *Theor. Comput. Sci.*, 592:97–114, 2015. 40

[BS19] Saikrishna Badrinarayanan and Akshayaram Srinivasan. Revisiting non-malleable secret sharing. In Yuval Ishai and Vincent Rijmen, editors, *EUROCRYPT 2019, Part I*, volume 11476 of *LNCS*, pages 593–622. Springer, Heidelberg, May 2019. 8

[CGGL20] Eshan Chattopadhyay, Jesse Goodman, Vipul Goyal, and Xin Li. Leakage-resilient extractors and secret-sharing against bounded collusion protocols. Cryptology ePrint Archive, Report 2020/478, 2020. https://ia.cr/2020/478. 2

[CL18] Eshan Chattopadhyay and Xin Li. Non-malleable codes, extractors and secret sharing for interleaved tampering and composition of tampering. Cryptology ePrint Archive, Report 2018/1069, 2018. https://ia.cr/2018/1069. 8

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *23rd ACM STOC*, pages 542–552. ACM Press, May 1991. 1

[DDN00] Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000. 1

[DORS03] Yevgeniy Dodis, Rafail Ostrovsky, Leonid Reyzin, and Adam Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. Cryptology ePrint Archive, Report 2003/235, 2003. https://eprint.iacr.org/2003/235. 9

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In Andrew Chi-Chih Yao, editor, *ICS 2010*, pages 434–452. Tsinghua University Press, January 2010. 1

[EHK+13] Alex Escala, Gottfried Herold, Eike Kiltz, Carla Ràfols, and Jorge Villar. An algebraic framework for Diffie-Hellman assumptions. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 129–147. Springer, Heidelberg, August 2013. 41

[FKL18] Georg Fuchsbauer, Eike Kiltz, and Julian Loss. The algebraic group model and its applications. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part II*, volume 10992 of *LNCS*, pages 33–62. Springer, Heidelberg, August 2018. 37, 41

[FMNV14] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 465–488. Springer, Heidelberg, February 2014. 1, 3, 13

[FV19] Antonio Faonio and Daniele Venturi. Non-malleable secret sharing in the computational setting: Adaptive tampering, noisy-leakage resilience, and improved rate. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 448–479. Springer, Heidelberg, August 2019. 1, 3, 8, 12

[GK18a] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th ACM STOC*, pages 685–698. ACM Press, June 2018. 1, 2, 8, 12

[GK18b] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing for general access structures. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part I*, volume 10991 of *LNCS*, pages 501–530. Springer, Heidelberg, August 2018. 1, 2, 8

[GMW87] Oded Goldreich, Silvio Micali, and Avi Wigderson. How to play any mental game or A completeness theorem for protocols with honest majority. In Alfred Aho, editor, *19th ACM STOC*, pages 218–229. ACM Press, May 1987. 30

[GSZ20] Vipul Goyal, Akshayaram Srinivasan, and Chenzhi Zhu. Multi-source non-malleable extractors and applications. *IACR Cryptol. ePrint Arch.*, 2020:157, 2020. 2, 4, 5, 29, 34

[KMS19] Ashutosh Kumar, Raghu Meka, and Amit Sahai. Leakage-resilient secret sharing against colluding parties. In David Zuckerman, editor, *60th FOCS*, pages 636–660. IEEE Computer Society Press, November 2019. 2

[KMZ20] Ashutosh Kumar, Raghu Meka, and David Zuckerman. Bounded collusion protocols, cylinder-intersection extractors and leakage-resilient secret sharing. Cryptology ePrint Archive, Report 2020/473, 2020. https://ia.cr/2020/473. 2

[Kra94] Hugo Krawczyk. Secret sharing made short. In Douglas R. Stinson, editor, *CRYPTO'93*, volume 773 of *LNCS*, pages 136–146. Springer, Heidelberg, August 1994. 6, 8, 22

[LCG+19] Fuchun Lin, Mahdi Cheraghchi, Venkatesan Guruswami, Reihaneh Safavi-Naini, and Huaxiong Wang. Leakage-resilient non-malleable secret sharing in non-compartmentalized models. *CoRR*, abs/1902.06195, 2019. 8

[NS20] Jesper Buus Nielsen and Mark Simkin. Lower bounds for leakage-resilient secret sharing. In Anne Canteaut and Yuval Ishai, editors, *EUROCRYPT 2020, Part I*, volume 12105 of *LNCS*, pages 556–577. Springer, Heidelberg, May 2020. 3

[OPVV18] Rafail Ostrovsky, Giuseppe Persiano, Daniele Venturi, and Ivan Visconti. Continuously non-malleable codes in the split-state model from minimal assumptions. In Hovav Shacham and Alexandra Boldyreva, editors, *CRYPTO 2018, Part III*, volume 10993 of *LNCS*, pages 608–639. Springer, Heidelberg, August 2018. 3, 4, 8, 13

[Rab89] Michael O. Rabin. Efficient dispersal of information for security, load balancing, and fault tolerance. *J. ACM*, 36(2):335–348, 1989. 30

[Sha79] Adi Shamir. How to share a secret. *Communications of the Association for Computing Machinery*, 22(11):612–613, November 1979. 1

[SV19] Akshayaram Srinivasan and Prashant Nalini Vasudevan. Leakage resilient secret sharing and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *CRYPTO 2019, Part II*, volume 11693 of *LNCS*, pages 480–509. Springer, Heidelberg, August 2019. 3, 8, 12

A Leakage-Resilient One-Time Non-Malleability

The blueprint of our scheme is the same as in [GSZ20]. However, we additionally consider a noisy-leakage resilience property from one of the building blocks.

Let $\Pi = (\mathsf{Share}, \mathsf{Rec})$ be the t-out-of-n Shamir's secret sharing scheme. For simplicity we assume that Π can support messages of variable length, namely the sharing procedure chooses a field that is large enough to encode the input message μ, we denote such field as $\mathbb{F}(|\mu|)$, or simply $\mathbb{F}$ when the message is clear from the context. A share σ_i of Π is a tuple (i, x) where $i \in [n]$ and $x \in \mathbb{F}$ is a field element; in particular, if p is the polynomial chosen by the Share algorithm, for all $i \in [n]$, $\sigma_i = (i, p(i))$. Let $\mathcal{S}_i(|\mu|) := \{(i, x) : x \in \mathbb{F}(|\mu|)\}$, clearly a secret sharing of μ has support $\mathcal{S}_1(|\mu|) \times \cdots \times \mathcal{S}_n(|\mu|)$. Consider the function idx that, upon input a tuple $\sigma = (i^*, x)$, outputs the first component $\mathsf{idx}(\sigma) = i^*$; in particular, for a share σ_i generated by the sharing function Share, it holds that $\mathsf{idx}(\sigma_i) = i$. Finally, let $\Pi' = (\mathsf{Share}', \mathsf{Rec}')$ be a split-state code with codeword space $\mathcal{S}_\mathsf{L} \times \mathcal{S}_\mathsf{R}$ and Com be a non-interactive commitment scheme. Consider the following derived scheme $\Pi^* = (\mathsf{Share}^*, \mathsf{Rec}^*)$.

- **Algorithm Share^*:** upon input μ, first compute$(\sigma_1, \ldots, \sigma_n) \leftarrow\$\ \mathsf{Share}(\mu)$. Then, for each $i \in [n]$, compute $(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \leftarrow\$\ \mathsf{Share}'(\sigma_i)$ and $(\sigma^{(1)}_{\mathsf{R},i}, \ldots, \sigma^{(n)}_{\mathsf{R},i}) \leftarrow\$\ \mathsf{Share}(\sigma_{\mathsf{R},i})$. Finally, set $\sigma^*_i = (\sigma_{\mathsf{L},i}, (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]})$ for all $i \in [n]$ and output $(\sigma^*_1, \ldots, \sigma^*_n)$.
- **Algorithm Rec^*:** upon input shares $(\sigma^*_i)_{i\in\mathcal{I}}$, parse $\sigma_i = (\sigma_{\mathsf{L},i}, (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]})$ for all $i \in \mathcal{I}$ and compute $\sigma_{\mathsf{R},i} = \mathsf{Rec}((\sigma^{(j)}_{\mathsf{R},i})_{j\in\mathcal{I}})$ and $\sigma_i = \mathsf{Rec}'(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$; check that there exist no distinct $i_1, i_2 \in \mathcal{I}$ such that $\mathsf{idx}(\sigma_{i_1}) = \mathsf{idx}(\sigma_{i_2})$ (and output $\perp$ otherwise). Finally, output $\mu = \mathsf{Rec}((\sigma_i)_{i\in\mathcal{I}})$.

We are now ready to state the following theorem.

Theorem 6. *Let $n, t \in \mathbb{N}$, $\ell_\mathsf{L}, \ell_\mathsf{R} \geq 0$ and $\epsilon' \in [0, 1]$. Assume that Π' is an asymmetric $(\ell_\mathsf{L}, \ell_\mathsf{R}, \epsilon', t)$-NLR-NMC satisfying the following property: There exists $\sigma^*_\mathsf{L} \in \mathcal{S}_\mathsf{L}$ such that, for any μ, there exists $\sigma_\mathsf{R} \in \mathcal{S}_\mathsf{R}$ such that $\mathsf{Rec}'(\sigma^*_\mathsf{L}, \sigma_\mathsf{R}) = \mu$. Then, the above secret sharing scheme Π^* is a t-out-of-n $(t-1, \ell^*, \epsilon)$-NLR-NMSS so long as:*

$$\ell_\mathsf{R} \geq (t-1) \cdot \ell^* + |\mu| \qquad \textit{and} \qquad \ell_\mathsf{L} \geq \ell^* + n \cdot (t-1) \cdot s_\mathsf{R},$$

where $|\mu| \in \mathbb{N}$ is the length of the message, $s_\mathsf{R} = \log |\mathcal{S}_\mathsf{R}|$ is the size of a right share under Π' and $\epsilon = 2n^2 \cdot 2^{-|\mu|} + 2n\epsilon'$.

The privacy property of Π^* follows readily by privacy of Π; in what follows, we focus on the proof of leakage-resilient continuous non-malleability.

For $r \in [n]$, consider the auxiliary hybrid experiments $\mathbf{Hyb}_r(\lambda, b)$ described in Fig. 5 along with the original experiment in order to highlight the main differences. In particular, in $\mathbf{Hyb}_r(\lambda, b)$, we replace the first r shares $(\sigma_1, \ldots, \sigma_r)$ from the first application of Π with random and independent values $(\sigma'_1, \ldots, \sigma'_r)$, letting the remaining shares $\sigma'_{r+1}, \ldots, \sigma'_n$ the same as the original experiment. Note that, when $r = 0$, we do not replace any share, hence, for all $b \in \{0, 1\}$, $\mathbf{Hyb}_0(\lambda, b) \equiv \mathbf{LR\text{-}NMSS}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, b)$.

The lemma below shows that the above hybrid experiments are all statistically close.

Lemma 9. *For all $b \in \{0, 1\}$, and all $r \in [n]$, it holds that*

$$\left\{\mathbf{Hyb}_{r-1}(\lambda, b)\right\}_{\lambda\in\mathbb{N}} \overset{\mathsf{s}}{\approx} \left\{\mathbf{Hyb}_r(\lambda, b)\right\}_{\lambda\in\mathbb{N}}.$$

Proof. The difference between the two hybrids is that in $\mathbf{Hyb}_r(\lambda, b)$ the share σ'_r is uniformly random, whereas in $\mathbf{Hyb}_{r-1}(\lambda, b)$ the share σ'_r is set to be σ_r (as defined in the original experiment). For any $j \in [n]$, let $\xi(j) \in \{1, 2\}$ be the index such that $j \in \mathcal{B}_{\xi(j)}$. The proof proceeds

$\mathbf{LR\text{-}NMSS}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, b)$ $\boxed{\mathbf{Hyb}^{\mu_0,\mu_1}_{r}(\lambda, b)}$:

$(\sigma_1, \ldots, \sigma_n) \leftarrow\!\$\ \mathsf{Share}(\mu_b)$

$\boxed{(\sigma'_1, \ldots, \sigma'_n) \leftarrow\!\$\ \times_{i\in[n]} \mathcal{S}_i(|\mu_b|)}$

$\boxed{\forall i > r, \sigma'_i := \sigma_i}$

$\forall i \in [n]:$

$\quad (\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \leftarrow\!\$\ \mathsf{Share}'(\sigma_i)$

$\quad \boxed{(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i}) \leftarrow\!\$\ \mathsf{Share}'(\sigma'_i)}$

$\quad (\sigma^{(1)}_{\mathsf{R},i}, \ldots, \sigma^{(n)}_{\mathsf{R},i}) \leftarrow\!\$\ \mathsf{Share}(\sigma_{\mathsf{R},i})$

$\quad \sigma^*_i := (\sigma_{\mathsf{L},i}, (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]})$

$\sigma^* := (\sigma^*_1, \ldots, \sigma^*_n)$

$(\mathcal{T}, f, \alpha) \leftarrow\!\$\ \mathsf{A}_1^{\mathcal{O}_{\mathsf{leak}}(\sigma^*,\cdot)}(1^\lambda)$

$\tilde{\mu} \leftarrow\!\$\ \mathcal{O}_{\mathsf{tamp}}(\sigma^*, \mathcal{T}, f)$

Return $\mathsf{A}_2(\alpha 1; \tilde{\mu})$

Algorithm $\mathsf{Split}((\sigma^*_i)_{i\in\mathcal{T}})$:

$\sigma_{\mathsf{R},i} = \mathsf{Rec}((\sigma^{(j)}_{\mathsf{R},i})_{j\in\mathcal{T}})$

$\sigma_i = \mathsf{Rec}'(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$

Output $(\sigma_i)_{i\in\mathcal{T}}$

Oracle $\mathcal{O}_{\mathsf{tamp}}(\sigma^*, \mathcal{T}, (f_1, \ldots, f_m))$:

$\forall i \in [m]: \tilde{\sigma}^*_{\mathcal{B}_i} := f_i(\sigma^*_{\mathcal{B}_i})$

$\tilde{\sigma}^* = (\sigma^*_1, \ldots, \sigma^*_n)$

$\forall i \in \mathcal{T}, \tilde{\sigma}^*_i = (\tilde{\sigma}_{\mathsf{L},i}, (\tilde{\sigma}^{(i)}_{\mathsf{R},j})_{j\in[n]})$

$(\tilde{\sigma}_i)_{i\in\mathcal{T}} = \mathsf{Split}((\tilde{\sigma}^*_i)_{i\in\mathcal{T}})$

If $\exists i_1, i_2 \in \mathcal{T} : \mathsf{idx}(\tilde{\sigma}_{i_1}) = \mathsf{idx}(\tilde{\sigma}_{i_2})$

$\quad$ Return $\perp$

$\boxed{\forall i_1, i_2 \in \mathcal{T} : \tilde{\sigma}_{i_1} = \sigma'_{i_2}}$

$\quad \boxed{\text{Let } \tilde{\sigma}_{i_1} := \sigma_{i_2}}$

$\tilde{\mu} = \mathsf{Rec}((\tilde{\sigma}_i)_{i\in\mathcal{T}})$

If $\tilde{\mu} \in \{\mu_0, \mu_1\}$, return $\diamond$

Return $\tilde{\mu}$

Oracle $\mathcal{O}_{\mathsf{leak}}(\sigma^*, (g_1, \ldots, g_m))$:

Return $g_1(\sigma^*_{\mathcal{B}_1}), \ldots, g_m(\sigma^*_{\mathcal{B}_m})$

Figure 5: Experiments in the proof of Theorem 6. The instructions boxed in red are the modifications introduced by the hybrid experiment. For compactness, we denote by Split the algorithm that reconstructs the shares σ_i from the shares $(\sigma_{\mathsf{L},i}, (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]})$.

by reduction to leakage-resilient t-time non-malleability of Π'. In more detail, for a fixed choice of $b \in \{0,1\}$ and $r \in [n]$, let A be an adversary telling apart the two hybrids with probability at least $1/\mathsf{poly}(\lambda)$. Consider the following (possibly inefficient) adversary A' attacking Π'.

1. **Setup.** Set the challenge messages to be σ_r and σ'_r sampled as in $\mathbf{Hyb}_r(\lambda, b)$.
2. **Shared randomness.** For every $i \in [n] \setminus \{r\}$, sample $\sigma_{\mathsf{L},i}, (\sigma^{(j)}_{\mathsf{R},i})_{j\in[n]}$ according to $\mathbf{Hyb}_r(\lambda, b)$. Then, let $i_{\mathsf{L}} = \xi(r)$ and $i_{\mathsf{R}} = 3 - i_{\mathsf{L}} \in \{1, 2\}$. Let $\mathcal{J}$ be any set such that $\mathcal{B}_{i_{\mathsf{L}}} \subseteq \mathcal{J} \subseteq [n]$ and $|\mathcal{J}| = t - 1$. For all $j \in \mathcal{J}$, sample the shares $\sigma^{(j)}_{\mathsf{R},r}$ uniformly at random. Finally, sample the left share σ^*_{L} given by property (i) of Π'.
   After this step, the reduction A' knows σ^*_{L} and the following values:

$$\forall i \in [n] \setminus \mathcal{J}: \qquad \sigma_{\mathsf{L},i}, \qquad (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]\setminus\{r\}} \tag{24}$$

$$\forall i \in \mathcal{J} \setminus \{r\}: \qquad \sigma_{\mathsf{L},i}, \qquad (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]} \tag{25}$$

$$\text{For } i = r: \qquad (\sigma^{(i)}_{\mathsf{R},j})_{j\in[n]} \tag{26}$$

3. **Leakage queries.** Upon receiving a leakage query $g = (g_1, g_2)$ from A, construct the following leakage functions.
   (a) Let g_{L} be the leakage function which takes as input the value $\sigma_{\mathsf{L},r}$, plugs it in Eq. (6) and appends the values of Eq. (5) to obtain $(\sigma^*_i)_{i\in\mathcal{B}_{i_{\mathsf{L}}}}$ (recall that $\mathcal{B}_{i_{\mathsf{L}}} \subseteq \mathcal{J}$), and finally outputs $\Lambda_{i_{\mathsf{L}}} = g_{i_{\mathsf{L}}}((\sigma^*_i)_{i\in\mathcal{B}_{i_{\mathsf{L}}}})$.
   (b) Let g_{R} be the leakage function which takes as input the value $\sigma_{\mathsf{R},r}$, computes the values $(\sigma^{(i)}_{\mathsf{R},r})_{i\in[n]\setminus\mathcal{J}}$ using $\sigma_{\mathsf{R},r}$ and the values $(\sigma^{(i)}_{\mathsf{R},r})_{i\in\mathcal{J}}$ and plugs them in Eq. (4) in

order to obtain $(\sigma_i^*)_{i\in[n]\setminus\mathcal{J}}$; then, appends the values of Eq. (5) to obtain $(\sigma_i^*)_{i\in\mathcal{B}_{i_\mathsf{R}}}$, and finally outputs $\Lambda_{i_\mathsf{R}} = g_{i_\mathsf{R}}((\sigma_i^*)_{i\in\mathcal{B}_{i_\mathsf{R}}})$.

Send $(g_\mathsf{L}, g_\mathsf{R})$ to the leakage oracle and forward the answer $\Lambda_1||\Lambda_2$ to A.

4. **Tampering query.** Upon receiving the tampering query $(\mathcal{T}, f = (f_1, f_2))$ from A, construct the following leakage and tampering functions.
   (a) Let $\hat{g}_\mathsf{L}$ be the leakage function which takes as input the value $\sigma_{\mathsf{L},r}$, plugs it in Eq. (6) and appends the values of Eq. (5) to obtain $(\sigma_i^*)_{i\in\mathcal{B}_{i_\mathsf{L}}}$, computes the tampered shares $(\tilde{\sigma}_i^*)_{i\in\mathcal{B}_{i_\mathsf{L}}} = f_{i_\mathsf{L}}((\sigma_i^*)_{i\in\mathcal{B}_{i_\mathsf{L}}})$, and finally outputs the values $(\tilde{\sigma}_{\mathsf{R},j}^{(i)})_{i\in\mathcal{B}_{i_\mathsf{L}},j\in\mathcal{T}}$.
   (b) For all $i\in\mathcal{T}$, let $f_{\mathsf{L},i}$ be the function which takes as input the value $\sigma_{\mathsf{L},r}$, obtains $(\sigma_j^*)_{j\in\mathcal{B}_{i_\mathsf{L}}}$ by appending the values of Eq. (5) and plugging $\sigma_{\mathsf{L},r}$ into Eq. (6), and then computes the tampered shares $(\tilde{\sigma}_j^*)_{j\in\mathcal{B}_{i_\mathsf{L}}} = f_{i_\mathsf{L}}((\sigma_j^*)_{j\in\mathcal{B}_{i_\mathsf{L}}})$ and outputs $\tilde{\sigma}_{\mathsf{L},i}$ if $i\in\mathcal{B}_{i_\mathsf{L}}$ and the special share σ_L^* otherwise.
   (c) For all $i\in\mathcal{T}$, let $f_{\mathsf{R},i}$ be the function which takes as input the value $\sigma_{\mathsf{R},r}$, computes the values $(\sigma_{\mathsf{R},r}^{(i)})_{i\in[n]\setminus\mathcal{J}}$ using $\sigma_{\mathsf{R},r}$ and the values $(\sigma_{\mathsf{R},r}^{(i)})_{i\in\mathcal{J}}$ and plugs them in Eq. (4) in order to obtain $(\sigma_i^*)_{i\in[n]\setminus\mathcal{J}}$; then, appends the values of Eq. (5) to obtain $(\sigma_i^*)_{i\in\mathcal{B}_{i_\mathsf{R}}}$, applies f_{i_R} to $(\sigma_i^*)_{i\in\mathcal{B}_{i_\mathsf{R}}}$, thus obtaining $(\tilde{\sigma}_i^*)_{i\in\mathcal{B}_{i_\mathsf{R}}}$, uses these values along with the values $(\tilde{\sigma}_{\mathsf{R},j}^{(i)})_{i\in\mathcal{B}_{i_\mathsf{L}},j\in\mathcal{T}}$ obtained by $\hat{g}_\mathsf{L}$ in order to reconstruct $\tilde{\sigma}_{\mathsf{R},i}$ for all $i\in\mathcal{T}$, and finally outputs $\tilde{\sigma}_{\mathsf{R},i}$ if $i\in\mathcal{B}_{i_\mathsf{L}}$ and a share $\sigma_{\mathsf{R},i}^*$ such that $\mathsf{Rec}'(\sigma_\mathsf{L}^*, \sigma_{\mathsf{R},i}^*) = \mathsf{Rec}'(\tilde{\sigma}_{\mathsf{L},i}, \tilde{\sigma}_{\mathsf{R},i})$ otherwise.

   Send the leakage query $(\hat{g}_\mathsf{L}, \varepsilon)$, thus obtaining $(\tilde{\sigma}_{\mathsf{R},j}^{(i)})_{i\in\mathcal{B}_{i^*},j\in[n]}$. Next, for all $i\in\mathcal{T}$, send the tampering query $(f_{\mathsf{L},i}, f_{\mathsf{R},i})$, thus obtaining the tampered share $\tilde{\sigma}_i$ (or $\perp$, in which case return $\perp$ to A), and replace $\tilde{\sigma}_i$ with σ_r if $\tilde{\sigma}_i = \diamond$ or replace $\tilde{\sigma}_i$ with σ_j if there exists $j\in[n]$ such that $\tilde{\sigma}_i = \sigma'_j$. Finally, check that there exist no distinct $i_1, i_2\in\mathcal{T}$ s.t. $\mathsf{idx}(\sigma_{i_1}) = \mathsf{idx}(\sigma_{i_2})$ (and output $\perp$ otherwise), reconstruct $\tilde{\mu} = \mathsf{Rec}((\tilde{\sigma}_i)_{i\in\mathcal{T}})$, replace $\tilde{\mu}$ with $\diamond$ if $\tilde{\mu}\in\{\mu_0,\mu_1\}$ and return $\tilde{\mu}$ to A.
5. **Guess.** Return the same distinguishing bit as that of A.

For the analysis, call $\mathbf{Bad}_i$ the event that one tampering query modifies the shares so that the tampered value $(\tilde{\sigma}_{\mathsf{L},i}, \tilde{\sigma}_{\mathsf{R},i})$ is a valid encoding of σ'_r (*i.e.*, the adversary purposely replaces $(\sigma_{\mathsf{L},i}, \sigma_{\mathsf{R},i})$ with a valid encoding of σ'_r). Clearly, the probability of the event $\mathbf{Bad}_i$ in the hybrid experiment $\mathbf{Hyb}_{r-1}$ is $O(2^{-\lambda})$ as provoking the event corresponds to guessing the value σ'_r which is uniformly random over $\mathcal{S}_r(|\mu_b|)$. Furthermore, the reduction perfectly simulates $\mathbf{Hyb}_{r-1}(\lambda, b)$ if the target codeword encodes σ_r and conditioning on $\mathbf{Bad} = \bigcup_i \mathbf{Bad}_i$ not happening. On the other hand, if the target codeword encodes σ'_r, the reduction perfectly simulates $\mathbf{Hyb}_r(\lambda, b)$. In particular, the latter holds because: (i) By perfect privacy of Shamir's secret sharing the distribution of the shares $(\sigma_i^*)_{i\in\mathcal{B}_{i_\mathsf{L}}}$ and $(\sigma_i^*)_{i\in\mathcal{B}_{i_\mathsf{R}}}$ computed inside the leakage and tampering oracles is identical to that of the target secret sharing of either $\mathbf{Hyb}_{r-1}(\lambda, b)$ or $\mathbf{Hyb}_r(\lambda, b)$; (ii) The auxiliary information leaked by the function $\hat{g}_\mathsf{L}$, along with the answer to the tampering queries $(f_{\mathsf{L},i}, f_{\mathsf{R},i})_{i\in[t]}$, yield a perfect simulation of A's tampering query. Hence, to conclude the proof, it only remains to show that the constraints on the leakage hold. The amount of leakage performed by the reduction is exactly the one performed by A, plus the leakage used to obtain the tampered shares $(\tilde{\sigma}_{\mathsf{R},j}^{(i)})_{i\in\mathcal{B}_{i^*},j\in[n]}$, therefore we need:

$$\ell_\mathsf{L} \geq \ell^* + t\cdot(t-1)\cdot s_\mathsf{R} \qquad \text{and} \qquad \ell_\mathsf{R} \geq \ell^*.$$

The lemma follows. □

Hence, by repeatedly applying the triangular inequality along with Lemma 9, we get that, for all $b\in\{0,1\}$,

$$\{\mathbf{Hyb}_0(\lambda, b)\}_{\lambda\in\mathbb{N}} \overset{\mathsf{s}}{\approx} \{\mathbf{Hyb}_n(\lambda, b)\}_{\lambda\in\mathbb{N}}.$$

The following lemma concludes the proof.

Lemma 10. $\{\mathbf{Hyb}_n(\lambda, 0)\}_{\lambda\in\mathbb{N}} \equiv \{\mathbf{Hyb}_n(\lambda, 1)\}_{\lambda\in\mathbb{N}}$.

Proof. First, note that all the leakage functions are computed using the fake shares $\sigma'_1, \dots, \sigma'_n$ and, therefore, the view of the adversary before the tampering query is independent of the message being μ_0 or μ_1. When the tampering query $(\mathcal{T}, f)$ occurs, let $\{\tilde{\sigma}_i\}_{i\in\mathcal{T}}$ be the tampered shares.

- If $\{\tilde{\sigma}_i\}_{i\in\mathcal{T}} \subseteq \{\sigma'_i\}_{i\in[n]}$, then the adversary receives $\diamond$ independently of the message being μ_0 or μ_1.
- If $\{\tilde{\sigma}_i\}_{i\in\mathcal{T}} \cap \{\sigma'_i\}_{i\in[n]} = \emptyset$, then no share $\tilde{\sigma}_i$ is replaced with the real share σ_i inside the tampering oracle, therefore the view of the adversary remains independent of the message being μ_0 or μ_1.
- If none of the above holds, the adversary gets to see at most $t-1$ shares, thus we can reduce to the perfect privacy of Shamir Secret Sharing.

□

B Overcoming the Capacity Upper Bound

In this section, we show that the negative result on the best achievable rate for continuously non-malleable secret sharing against joint tampering attacks can be overcome in idealized models. We do so by describing a general template for a rate compiler based on a cryptographic hash function $\mathsf{H} : \{0,1\}^* \to \{0,1\}^\lambda$.

Let $(\mathsf{IDisp}, \mathsf{IRec})$ be an information dispersal, $\Pi = (\mathsf{Share}, \mathsf{Rec})$ be a secret sharing scheme, and $\Sigma' = (\mathsf{Enc}, \mathsf{Dec})$ be a secret-key encryption scheme. Consider the following derived secret sharing scheme $\Pi^* = (\mathsf{Share}^*, \mathsf{Rec}^*)$.

- **Algorithm Share^*:** upon input a message μ, sample a random key $\kappa \leftarrow\$ \mathcal{K}$, compute $\gamma \leftarrow\$ \mathsf{Enc}(\kappa, \mu)$, $(\gamma_1, \dots, \gamma_n) \leftarrow \mathsf{IDisp}(\gamma)$ and $(\kappa_1, \dots, \kappa_n) \leftarrow\$ \mathsf{Share}(\kappa)$; finally, for all $i \in [n]$, let $\sigma_i = (h, \kappa_i, \gamma_i)$ such that $h = \mathsf{H}(\gamma)$, and output $(\sigma_1, \dots, \sigma_n)$.
- **Algorithm Rec^*:** upon input a set $(\sigma_i)_{i\in\mathcal{I}}$ of at least t shares parse $\sigma_i = (h_i, \kappa_i, \gamma_i)$ for all $i \in \mathcal{I}$, check that all the values h_i agree on a single value h (and return $\perp$ if not), reconstruct $\kappa = \mathsf{Rec}((\kappa_i)_{i\in\mathcal{I}})$ and $\gamma = \mathsf{IRec}((\gamma_i)_{i\in\mathcal{I}})$, check that $h = \mathsf{H}(\gamma)$ (and return $\perp$ if not), and finally output $\mu = \mathsf{Dec}(\kappa, \gamma)$.

We start by showing that the asymptotic rate of the above secret sharing scheme is equal to the reconstruction threshold t, and thus it matches the best achievable rate for (malleable) secret sharing scheme in the computational setting. Furthermore, in the following subsections we will also show that the scheme is continuously non-malleable against joint tampering attacks assuming that either (i) H is modeled as a non-programmable random oracle, or (ii) H is instantiated using Pedersen's hash function in the algebraic group model [FKL18].

Theorem 7. *The asymptotic rate of Π^* is $\varrho = t$.*

Proof. The length of the key κ for the SKE scheme Σ, and thus the size of the shares of the secret sharing scheme Π, only depend on the security parameter λ, the number of parties n and the tolerated leakage ℓ (but not on the length $|\mu|$ of the message); call $s_0(\lambda, n, \ell)$ the length of this portion of the final shares (namely, κ_i). On the other side, it is possible to achieve length of the ciphertext $|\gamma| = |\mu| + O(\lambda)$, hence the length of each share γ_i of the information dispersal amounts to $|\gamma_i| = \frac{|\gamma|}{t} = \frac{|\mu|+O(\lambda)}{t}$. Putting it together, we have obtained:

$$s(\lambda, n, \ell, |\mu|) = s_0(\lambda, n, \ell) + \frac{|\mu| + O(\lambda)}{t},$$

$\mathbf{LR\text{-}CNMSS}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, b)$ $\boxed{\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, b)}$:

$\kappa \leftarrow\!\!\$\ \mathcal{K}$
$\boxed{\hat{\kappa} \leftarrow\!\!\$\ \mathcal{K}}$
$(\kappa_1, \ldots, \kappa_n) \leftarrow\!\!\$\ \mathsf{Share}(\kappa)$
$\boxed{(\kappa_1, \ldots, \kappa_n) \leftarrow\!\!\$\ \mathsf{Share}(\hat{\kappa})}$
$\gamma \leftarrow\!\!\$\ \mathsf{Enc}(\kappa, \mu_b)$
$(\gamma_1, \ldots, \gamma_n) = \mathsf{IDisp}(\gamma)$
$h = \mathsf{H}(\gamma)$
$\forall i \in [n]$:
  $\sigma_i := (h, \kappa_i, \gamma_i)$
$\sigma := (\sigma_1, \ldots, \sigma_n)$
$\texttt{stop} \leftarrow \texttt{false}$
Return $\mathsf{A}^{\mathcal{O}_{\mathsf{tamp}}(\sigma,\cdot),\mathcal{O}_{\mathsf{leak}}(\sigma,\cdot),\mathsf{H}(\cdot)}(1^\lambda)$

Oracle $\mathcal{O}_{\mathsf{leak}}(\sigma, (g_1^{\mathsf{H}}, \ldots, g_m^{\mathsf{H}}))$:

If $\texttt{stop} = \texttt{true}$, return $\perp$
Else, return $g_1^{\mathsf{H}}(\sigma_{\mathcal{B}_1}), \ldots, g_m^{\mathsf{H}}(\sigma_{\mathcal{B}_m})$

Oracle $\mathcal{O}_{\mathsf{tamp}}(\sigma, \mathcal{T}, (f_1^{\mathsf{H}}, \ldots, f_m^{\mathsf{H}}))$:

If $\texttt{stop} = \texttt{true}$, return $\perp$
$\forall i \in [m] : \tilde{\sigma}_{\mathcal{B}_i} := f_i^{\mathsf{H}}(\sigma_{\mathcal{B}_i})$
$\tilde{\sigma} = (\sigma_1, \ldots, \sigma_n)$
$\forall i \in \mathcal{T}, \tilde{\sigma}_i = (\tilde{h}_i, \tilde{\kappa}_i, \tilde{\gamma}_i)$
If $\exists i_1, i_2 \in \mathcal{T} : \tilde{h}_{i_1} \neq \tilde{h}_{i_2}$
  $\texttt{stop} \leftarrow \texttt{true}$ and return $\perp$
Else, let $\tilde{h} := \tilde{h}_i$
$\tilde{\gamma} = \mathsf{IRec}((\tilde{\gamma}_i)_{i\in\mathcal{T}})$
If $\mathsf{H}(\tilde{\gamma}) \neq \tilde{h}$
  $\texttt{stop} \leftarrow \texttt{true}$ and return $\perp$
If $\mathsf{IDisp}(\tilde{\gamma})_{\mathcal{T}} \neq (\tilde{\gamma}_i)_{i\in\mathcal{T}}$
  $\texttt{stop} \leftarrow \texttt{true}$ and return $\perp$
$\tilde{\kappa} = \mathsf{Rec}((\tilde{\kappa}_i)_{i\in\mathcal{T}})$
If $\tilde{\kappa} = \perp$, $\texttt{stop} \leftarrow \texttt{true}$ and return $\perp$
$\boxed{\text{If } \tilde{\kappa} = \hat{\kappa},\ \tilde{\kappa} \leftarrow \kappa}$
$\tilde{\mu} = \mathsf{Dec}(\tilde{\kappa}, \tilde{\gamma})$
If $\tilde{\mu} = \perp$, $\texttt{stop} \leftarrow \texttt{true}$ and return $\perp$
If $\tilde{\mu} \in \{\mu_0, \mu_1\}$, return $\diamond$
Else return $\tilde{\mu}$

Figure 6: Experiments in the proof of Theorem 8. The instructions boxed in red are the modifications introduced by the hybrid experiment.

that translates into

$$\begin{aligned}
\varrho &= \inf_{\lambda\in\mathbb{N}} \lim_{|\mu|\to\infty} \frac{|\mu|}{s(\lambda, n, \ell, |\mu|)} \\
&= \inf_{\lambda\in\mathbb{N}} \lim_{|\mu|\to\infty} \frac{t \cdot |\mu|}{|\mu| + t \cdot \mathsf{poly}(\lambda, n, \ell)} \\
&= t.
\end{aligned}$$

□

B.1 Analysis in the Random Oracle Model

In the random oracle model (ROM), a secret sharing scheme $\Pi^{\mathsf{H}} = (\mathsf{Share}^{\mathsf{H}}, \mathsf{Rec}^{\mathsf{H}})$ may rely on an ideal hash function H sampled uniformly among all possible functions mapping $\{0,1\}^*$ into $\{0,1\}^\lambda$. In this setting, the adversary $\mathsf{A} = \mathsf{A}^{\mathsf{H}}$ is allowed to make random-oracle queries and to specify tampering (resp. leakage) functions f^{H} (resp. g^{H}) that can additionally query the random oracle.

Theorem 8. *Let $n, t, \ell \in \mathbb{N}$ be parameters. Assume that:*

- $(\mathsf{IDisp}, \mathsf{IRec})$ *is a t-out-of-n information dispersal scheme;*
- $(\mathsf{Share}, \mathsf{Rec})$ *is a t-out-of-n $(\ell, t-1)$-LR-CNMSS;*
- $(\mathsf{Enc}, \mathsf{Dec})$ *is an IND-CCA secure secret-key encryption scheme.*

Then, Π^* is a t-out-of-n $(\ell, t-1)$-LR-CNMSS in the non-programmable random oracle model.

Similarly to the proof of Theorem 3, we consider a hybrid experiment in which we replace the secret sharing of the key κ with a secret sharing of an unrelated random key $\hat{\kappa}$ (see Figure 6). The lemma below states that this experiment is computationally indistinguishable from the original security game.

Lemma 11. *For all $\mu_0, \mu_1 \in \mathcal{M}$, all $(t-1)$-sized partitions $\mathcal{B}$ of $[n]$, and all $b \in \{0,1\}$, it holds that:*

$$\{\mathbf{LR\text{-}CNMSS}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, b)\}_{\lambda\in\mathbb{N}} \overset{\mathrm{c}}{\approx} \{\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, b)\}_{\lambda\in\mathbb{N}}.$$

Proof. By reduction to leakage-resilient continuous non-malleability of the underlying secret sharing scheme. Fix any $b \in \{0,1\}$, and suppose that there exist two messages $\mu_0, \mu_1 \in \mathcal{M}$, a $(t-1)$-sized partition $\mathcal{B} = (\mathcal{B}_1, \ldots, \mathcal{B}_m)$ of $[n]$, and a PPT adversary A able to distinguish between the two experiments with non-negligible probability. Consider the following reduction $\hat{\mathsf{A}}$ against leakage-resilient continuous non-malleability of Π.

1. **Setup.** Sample a random key $\kappa_{\mathsf{prf}} \in \{0,1\}^\lambda$ for a pseudorandom function $\mathsf{F}(\kappa_{\mathsf{prf}}, \cdot)$ mapping ciphertexts into λ-bit strings, and let $\mathcal{Q}_{\mathsf{RO}}$ be an initially empty set. Set the challenge messages to be κ and $\hat{\kappa}$ sampled as in the hybrid experiment, then compute $\gamma \leftarrow\!\!\$\ \mathsf{Enc}(\kappa, \mu)$, $(\gamma_1, \ldots, \gamma_n) \leftarrow \mathsf{IDisp}(\gamma)$ and $h = \mathsf{F}(\kappa_{\mathsf{prf}}, \gamma)$. In what follows, all tampering and leakage functions constructed by the reduction implicitly hard-wire the value h, the shares $(\gamma_j)_{j\in\mathcal{B}_i}$, the list $\mathcal{Q}_{\mathsf{RO}}$ and (a description of) the query they refer to.
2. **Random oracle queries.** Upon input any value x, run $h = \mathsf{F}(\kappa_{\mathsf{prf}}, x)$, add (x, h) to $\mathcal{Q}_{\mathsf{RO}}$ and return h.
3. **Leakage queries.** Upon input a leakage query $(g_1, \ldots, g_m)$, for all $i \in [m]$, construct the leakage function $\hat{g}_i$ which takes as input the shares $(\kappa_j)_{j\in\mathcal{B}_i}$, runs $\Lambda_i = g_i((\sigma_j)_{j\in\mathcal{B}_i})$ for $\sigma_j := (h, \kappa_j, \gamma_j)$, and outputs Λ_i; then, send the leakage query $(\hat{g}_1, \ldots, \hat{g}_m)$ to the target leakage oracle and forward the answer to A.
4. **Tampering queries.** Upon input a tampering query $(\mathcal{T}, (f_1, \ldots, f_m))$, execute the following steps.
   (a) *Obtain the tampered hash.* For all $i \in [m]$, construct the function $\hat{h}_i$ which takes as input the shares $(\kappa_j)_{j\in\mathcal{B}_i}$, computes the tampering $(\tilde{h}_j, \tilde{\kappa}_j, \tilde{\gamma}_j)_{j\in\mathcal{B}_i} = f_i((h, \kappa_j, \gamma_j)_{j\in\mathcal{B}_i})$ and outputs $\tilde{h}_j$ for some $j \in \mathcal{B}_i \cap \mathcal{T}$ if $\tilde{h}_{j_1} = \tilde{h}_{j_2}$ for all $j_1, j_2 \in \mathcal{B}_i \cap \mathcal{T}$ (and $\perp$ otherwise). Run procedure $\mathsf{LeakTamper}_{\mathcal{T}}(\hat{h}_1, \ldots, \hat{h}_m)$ to obtain either the tampered hash $\tilde{h}$ or $\perp$.
   (b) *Obtain the tampered ciphertext.* For all $i \in [m]$, construct the function $\hat{h}'_i$ which also hard-wires the value $\tilde{h}$ and all the previous leakage and tampering queries, takes as input the shares $(\kappa_j)_{j\in\mathcal{B}_i}$, runs all the hard-wired leakage and tampering functions, creates a list $\mathcal{Q}^{(i)}_{\mathsf{RO}}$ of all the random-oracle queries performed by such functions and finally searches for an item $(\tilde{\gamma}, \tilde{h})$ in $\mathcal{Q}_{\mathsf{RO}} \cup \mathcal{Q}^{(i)}_{\mathsf{RO}}$ and outputs $\tilde{\gamma}$ if such item is found (and otherwise outputs $\perp$). Run procedure $\mathsf{LeakTamper}_{\mathcal{T}}(\hat{h}'_1, \ldots, \hat{h}'_m)$ to obtain either the tampered ciphertext or $\perp$.
   (c) *Check that everything is correct.* For all $i \in [m]$, construct the function $\hat{h}''_i$ which also hard-wires the values $\tilde{h}$ and $\tilde{\gamma}$, takes as input the shares $(\kappa_j)_{j\in\mathcal{B}_i}$, computes $(\hat{\gamma}_1, \ldots \hat{\gamma}_n) = \mathsf{IDisp}(\tilde{\gamma})$ and $(\tilde{h}_j, \tilde{\kappa}_j, \tilde{\gamma}_j)_{j\in\mathcal{B}_i} = f_i((h, \kappa_j, \gamma_j)_{j\in\mathcal{B}_i})$ and, for all $j \in \mathcal{B}_i \cap \mathcal{T}$, checks that $\tilde{h}_j = \tilde{h}$ and $\tilde{\gamma}_j = \hat{\gamma}_j$, replacing the shares $(\kappa_j)_{j\in\mathcal{B}_i}$ with $\perp$ if not. Run procedure $\mathsf{CheckEq}_{\mathcal{T}}(\hat{h}''_1, \ldots, \hat{h}''_m)$ to obtain either $\diamond$, if the check confirms $\tilde{h}$ and $\tilde{\gamma}$, or $\perp$.
   (d) *Obtain the tampered key.* For all $i \in [m]$, construct the tampering function $\hat{f}_i$ which takes as input all the shares $(\kappa_j)_{j\in\mathcal{B}_i}$, computes the tampering $(\tilde{h}_j, \tilde{\kappa}_j, \tilde{\gamma}_j)_{j\in\mathcal{B}_i} =$

$f_i((h, \kappa_j, \gamma_j)_{j \in \mathcal{B}_i})$ and outputs $(\tilde{\kappa}_j)_{j \in \mathcal{B}_i}$. Send $(\mathcal{T}, (\hat{f}_1, \ldots, \hat{f}_m))$ to the tampering oracle, obtaining a tampered key $\tilde{\kappa} \in \mathcal{K} \cup \{\diamond, \perp\}$.

(e) *Obtain the tampered message.* If any of the previous steps resulted in a $\perp$, output $\perp$ and self-destruct; otherwise, compute $\tilde{\mu} = \mathsf{Dec}(\tilde{\kappa}, \tilde{\gamma})$, replacing $\tilde{\kappa}$ with κ if $\tilde{\kappa} = \diamond$, and return $\tilde{\mu}$ to the adversary (and self-destruct if $\tilde{\mu} = \perp$).

5. **Guess.** Output the same distinguishing bit as A does.

Note that the reduction $\hat{\mathsf{A}}$ answers random oracle queries consistently by replacing H with a PRF $\mathsf{F}(\kappa_{\mathsf{prf}}, \cdot)$.[13] The latter only requires the random oracle to be non-adaptively programmable, which is known to be equivalent to a non-programmable random oracle [BM15]. By security of the PRF, this change is unnoticed by the attacker. (The reduction is straightforward, and therefore omitted.)

Consider the following event **Bad**, defined over the probability space of the original experiment: The event becomes true if the attacker ever submits a query to the tampering oracle that triggers the condition $\tilde{\kappa} = \hat{\kappa}$, where the key $\hat{\kappa}$ is sampled at the beginning of the experiment. It is easy to see that, conditioning on $\overline{\mathbf{Bad}}$, the above reduction perfectly emulates the view of the adversary in experiment $\mathbf{LR\text{-}CNMSS}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A}}(\lambda, b)$ when $(\kappa_1, \ldots, \kappa_n)$ is a secret sharing of κ, and perfectly emulates the view of the adversary in $\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A}}(\lambda, b)$ when $(\kappa_1, \ldots, \kappa_n)$ is a secret sharing of $\hat{\kappa}$. Since, using a union bound, $\Pr[\mathbf{Bad}] \leq \mathsf{poly}(\lambda) \cdot 2^{-\lambda} = \mathsf{negl}(\lambda)$, by a standard argument the two experiments are computationally indistinguishable. The lemma follows. □

The lemma below concludes the proof of Theorem 8.

Lemma 12. *For all $\mu_0, \mu_1 \in \mathcal{M}$, and all $(t-1)$-sized partitions $\mathcal{B}$ of $[n]$, it holds that:*

$$\{\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A}}(\lambda, 0)\}_{\lambda \in \mathbb{N}} \overset{\mathsf{c}}{\approx} \{\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A}}(\lambda, 1)\}_{\lambda \in \mathbb{N}}.$$

Proof. By reduction to IND-CCA security of the symmetric encryption scheme. Suppose that there exist two messages μ_0, μ_1, a $(t-1)$-sized partition $\mathcal{B}$ of $[n]$, and a PPT adversary A that is able to distinguish between $\mathbf{Hyb}(\lambda, 0)$ and $\mathbf{Hyb}(\lambda, 1)$ with non-negligible probability. Consider the following reduction A' attacking IND-CCA security of Σ.

1. **Setup.** Sample a random key $\kappa_{\mathsf{prf}} \in \{0,1\}^\lambda$ for a pseudorandom function $\mathsf{F}(\kappa_{\mathsf{prf}}, \cdot)$ mapping ciphertexts into λ-bit strings, and let $\mathcal{Q}_{\mathsf{RO}}$ be an initially empty set. Set the challenge messages to be μ_0 and μ_1, obtain the challenge ciphertext $\hat{\gamma}$, sample a key $\hat{\kappa} \leftarrow\$ \mathcal{K}$ and compute $(\gamma_1, \ldots, \gamma_n) = \mathsf{IDisp}(\hat{\gamma})$, $(\kappa_1, \ldots, \kappa_n) \leftarrow\$ \mathsf{Share}(\hat{\kappa})$ and $h = \mathsf{F}(\kappa_{\mathsf{prf}}, \hat{\gamma})$. Finally, for all $i \in [n]$, construct the share $\sigma_i^* := (h, \kappa_i, \gamma_i)$.
2. **Random oracle queries.** Upon input any value x, run $h = \mathsf{F}(\kappa_{\mathsf{prf}}, x)$, add (x, h) to $\mathcal{Q}_{\mathsf{RO}}$ and return h.
3. **Leakage queries.** Answer leakage queries as in the original experiment.
4. **Tampering queries.** Upon input a tampering query $(\mathcal{T}, (f_1, \ldots, f_m))$, for all $i \in [m]$, compute $(\tilde{\sigma}_j)_{j \in \mathcal{B}_i} = f_i((\sigma_j)_{j \in \mathcal{B}_i})$, perform the consistency checks on the tampered hash $\tilde{h}$ and the tampered ciphertext $\tilde{\gamma}$ (and output $\perp$ if any test fails) and then reconstruct the tampered key $\tilde{\kappa} \in \mathcal{K}$. If $\tilde{\kappa} = \hat{\kappa}$, obtain the tampered message $\tilde{\mu} \in \mathcal{M} \cup \{\perp\}$ by sending $\tilde{\gamma}$ to the decryption oracle; otherwise, compute $\tilde{\mu} = \mathsf{Dec}(\tilde{\kappa}, \tilde{\gamma})$. Finally, return $\tilde{\mu}$ to A (and self-destruct if $\tilde{\mu} = \perp$).
5. **Guess.** Output the same distinguishing bit as A.

For the analysis, note that the reduction is perfect and, in particular, for $b \in \{0,1\}$, it perfectly simulates $\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, b)$ whenever the challenge ciphertext γ is an encryption of μ_b. The only difference is that random-oracle queries are answered using a PRF, but as before this change

[13] Such a PRF exists assuming one-way functions, which are anyway necessary for continuous non-malleability.

cannot affect the advantage of A by more than a negligible amount (by security of the PRF). This concludes the proof of the lemma. □

The proof of Theorem 8 now follows by combining the above lemmas.

B.2 Analysis in the Algebraic Group Model

Let GroupGen a polynomial-time algorithm that upon input the security parameter 1^λ outputs the description $(\mathbb{G}, g)$ of a cyclic group of prime order $q \geq 2^\lambda$ where the discrete logarithm (DLOG) assumption holds. We use the implicit notation of Escala *et al.* [EHK+13] for groups. In particular, we use the additive notation for $\mathbb{G}$ and let $[x] := x \cdot g$ where g is a generator of $\mathbb{G}$ and $x \in \mathbb{Z}_q$. Let $\mathsf{H} = (\mathsf{H.KGen}, \mathsf{H.Eval})$ be a *keyed* hash function, where algorithm KGen takes as input a description of the group $\mathbb{G}$ and outputs a public key $\mathsf{crs} \in \{0,1\}^*$, whereas algorithm Eval takes as input crs and a string x and outputs a value h.
Consider the following keyed hash function with domain $\mathbb{Z}_q^k$, for some $k \in \mathbb{N}$:

- $\mathsf{H.KGen}(\mathbb{G})$ samples $[\mathbf{g}] \leftarrow\!\!\$\ \mathbb{G}^k$ uniformly at random and output $\mathsf{crs} = (\mathbb{G}, [\mathbf{g}])$
- $\mathsf{H.Eval}(\mathsf{crs}, \mathbf{x})$ outputs $[\mathbf{g}]^\mathsf{T} \cdot \mathbf{x}$.

It is easy to see that the above hash function is collision resistant assuming hardness of DLOG. We say that a tampering function is *algebraic* [FKL18], if given as input the description of a group $\mathbb{G}$ and group elements $[\mathbf{x}] \in \mathbb{G}^n$ for $n \in \mathbb{N}$ and a string $x' \in \{0,1\}^*$ it outputs: (i) $[\mathbf{y}] \in \mathbb{G}^m$ for $m \in \mathbb{N}$, (ii) a string $y' \in \{0,1\}^*$, and (iii) an auxiliary matrix $\mathbf{M}$ such that $\mathbf{y}^\mathsf{T} = \mathbf{x}^\mathsf{T} \cdot \mathbf{M}$.

The theorem below states that the secret sharing Π^* instantiated with the above keyed hash function is leakage-resilient continuously non-malleable under the DLOG assumption, so long as the tampering queries are restricted to be algebraic. We refer to the latter restriction as "security in the algebraic group model". Wlog., we will assume that the symmetric encryption scheme has message and ciphertext space equal to $\mathbb{Z}_q^k$. Note that since we now rely on a *keyed* hash function, Π^* is a secret sharing scheme in the CRS model (as defined in [BFV19]). For simplicity, in the proof we will stick to the case where the partition $\mathcal{B}$ is chosen independently of the CRS. However, it is not hard to see that security still holds even if we let the adversary choose $\mathcal{B}$ as function of the CRS.

Theorem 9. *Let $n, t, \ell \in \mathbb{N}$ be parameters. Assume that:*

- *(IDisp, IRec) is a t-out-of-n information dispersal;*
- *(Share, Rec) is a t-out-of-n $(\ell, t-1)$-LR-CNMSS;*
- *(Enc, Dec) is an IND-CCA secure secret-key encryption scheme.*

Then, Π^* instantiated with the above keyed hash function is a t-out-of-n $(\ell, t-1)$-LR-CNMSS in the algebraic group model under the DLOG assumption.

Proof. Let $(\mathcal{T}, f_1, \ldots, f_m)$ be a generic tampering query issued by the adversary A playing the security experiment defining leakage-resilient continuous non-malleability. Being the functions $f_1, \ldots, f_m$ algebraic, we assume that each function f_i for $i \in [m]$ additionally outputs a matrix $\mathbf{M}^{(i)} \in \mathbb{Z}_q^{(k+1)\times|\mathcal{B}_i|}$ (ignored by the tampering oracle) which is such that $(\tilde{h}_j)_{j\in\mathcal{B}_i} = ([\mathbf{g}]^\mathsf{T}, h) \cdot \mathbf{M}^{(i)}$. For any $j \in [m]$, let $\mathbf{M}^{(i)} = (\mathbf{m}_j)_{j\in\mathcal{B}_i}$ and consider the matrix $\mathbf{M} = (\mathbf{m}_i)_{i\in[n]}$; then, $(\tilde{h}_j)_{j\in[n]} = ([\mathbf{g}]^\mathsf{T}, h) \cdot \mathbf{M}$.

Similarly to the proof of Theorem 3, we consider a hybrid experiment in which we replace the secret sharing of the key κ with a secret sharing of an unrelated random key $\hat{\kappa}$ (see Figure 7). The lemma below states that this experiment is computationally indistinguishable from the original security game.

$\mathbf{LR\text{-}CNMSS}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda,b)$ $\boxed{\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda,b)}$:

$\kappa \leftarrow\!\!\$\ \mathcal{K}$
$\boxed{\hat{\kappa} \leftarrow\!\!\$\ \mathcal{K}}$
$(\kappa_1,\ldots,\kappa_n) \leftarrow\!\!\$\ \mathsf{Share}(\kappa)$
$\boxed{(\kappa_1,\ldots,\kappa_n) \leftarrow\!\!\$\ \mathsf{Share}(\hat{\kappa})}$
$\boldsymbol{\gamma} \leftarrow\!\!\$\ \mathsf{Enc}(\kappa,\mu_b)$
$(\gamma_1,\ldots,\gamma_n) = \mathsf{IDisp}(\boldsymbol{\gamma})$
$(\mathbb{G},q) \leftarrow\!\!\$\ \mathsf{GroupGen}(1^\lambda)$
$\mathsf{crs} = (\mathbb{G},\mathbf{g}) \leftarrow\!\!\$\ \mathsf{H.KGen}(\mathbb{G})$
$h = \mathsf{H.Eval}(\mathsf{crs},\boldsymbol{\gamma})$
$\forall i \in [n]$:
  $\sigma_i := (h,\kappa_i,\gamma_i)$
$\sigma := (\sigma_1,\ldots,\sigma_n)$
$\mathtt{stop} \leftarrow \mathtt{false}$
Return $\mathsf{A}^{\mathcal{O}_{\mathsf{tamp}}(\sigma,\cdot),\mathcal{O}_{\mathsf{leak}}(\sigma,\cdot)}(1^\lambda,\mathsf{crs})$

Oracle $\mathcal{O}_{\mathsf{leak}}(\sigma,(g_1,\ldots,g_m))$:
If $\mathtt{stop} = \mathtt{true}$, return $\perp$
Else, return $g_1(\sigma_{\mathcal{B}_1}),\ldots,g_m(\sigma_{\mathcal{B}_m})$

Oracle $\mathcal{O}_{\mathsf{tamp}}(\sigma,\mathcal{T},(f_1,\ldots,f_m))$:
If $\mathtt{stop} = \mathtt{true}$, return $\perp$
$\forall i \in [m] : (\tilde{\sigma}_{\mathcal{B}_i},\mathbf{M}^{(i)}) := f_i(\sigma_{\mathcal{B}_i})$
For $i \in [m]$, parse $\mathbf{M}^{(i)} := (\mathbf{m}_j)_{j\in\mathcal{B}_i}$
$\mathbf{M} = (\mathbf{m}_j)_{j\in[n]}$
$\tilde{\sigma} = (\sigma_1,\ldots,\sigma_n)$
$(\boldsymbol{\delta}_1,\ldots,\boldsymbol{\delta}_n) = (\mathbf{I}\|\boldsymbol{\gamma})\cdot\mathbf{M}$
$\forall i \in \mathcal{T}, \tilde{\sigma}_i = (\tilde{h}_i,\tilde{\kappa}_i,\tilde{\gamma}_i)$
If $\exists i_1,i_2 \in \mathcal{T} : \tilde{h}_{i_1} \neq \tilde{h}_{i_2}$
  $\mathtt{stop} \leftarrow \mathtt{true}$ and return $\perp$
$\boxed{\text{If } \exists i_1,i_2 \in \mathcal{T} : \boldsymbol{\delta}_{i_1} \neq \boldsymbol{\delta}_{i_2}}$
  $\boxed{\mathtt{stop} \leftarrow \mathtt{true} \text{ and return } \perp}$
Else, let $\tilde{h} := \tilde{h}_i$
$\tilde{\boldsymbol{\gamma}} = \mathsf{IRec}((\tilde{\gamma}_i)_{i\in\mathcal{T}})$
$\boxed{\tilde{\boldsymbol{\gamma}} := \boldsymbol{\delta}_i}$
If $\mathsf{IDisp}(\tilde{\boldsymbol{\gamma}})_{\mathcal{T}} \neq (\tilde{\gamma}_i)_{i\in\mathcal{T}}$
  $\mathtt{stop} \leftarrow \mathtt{true}$ and return $\perp$
$\tilde{\kappa} = \mathsf{Rec}((\tilde{\kappa}_i)_{i\in\mathcal{T}})$
If $\tilde{\kappa} = \perp$, $\mathtt{stop} \leftarrow \mathtt{true}$ and return $\perp$
If $\mathsf{H.Eval}(\mathsf{crs},\tilde{\boldsymbol{\gamma}}) \neq \tilde{h}$
  $\mathtt{stop} \leftarrow \mathtt{true}$ and return $\perp$
$\boxed{\text{If } \tilde{\kappa} = \hat{\kappa},\ \tilde{\kappa} \leftarrow \kappa}$
$\tilde{\mu} = \mathsf{Dec}(\tilde{\kappa},\tilde{\boldsymbol{\gamma}})$
If $\tilde{\mu} = \perp$, $\mathtt{stop} \leftarrow \mathtt{true}$ and return $\perp$
If $\tilde{\mu} \in \{\mu_0,\mu_1\}$, return $\diamond$
Else return $\tilde{\mu}$

Figure 7: Experiments in the proof of Theorem 9. The instructions boxed in red are the modifications introduced by the hybrid experiment.

Lemma 13. *For all $\mu_0,\mu_1 \in \mathcal{M}$, all $(t-1)$-sized partitions of $[n]$, and all $b \in \{0,1\}$ it holds that:*

$$\{\mathbf{LR\text{-}CNMSS}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda,b)\}_{\lambda\in\mathbb{N}} \overset{c}{\approx} \{\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda,b)\}_{\lambda\in\mathbb{N}}.$$

Proof. We claim that the two experiments are indistinguishable unless the adversary can find a collision in the hash function or can break leakage-resilient continuous non-malleability of the underlying secret sharing scheme.

More precisely, consider the event that the adversary submits a tampering query such that $\exists i_1,i_2 \in \mathcal{T} :\ \tilde{h}_{i_1} = \tilde{h}_{i_2} \wedge \boldsymbol{\delta}_{i_1} \neq \boldsymbol{\delta}_{i_2}$. Note that this bad event is indeed a distinguishing event between the two experiments. In fact, in the hybrid experiment the event triggers a self-destruct while the latter does not happen in the original experiment. However, it is immediate to see that if such an event happens then we have an algebraic adversary that breaks the collision resistance of H (thus contradicting DLOG). In particular, the latter follows by the homomorphic property of the hash function: Recall that $\mathbf{M} = (\mathbf{m}_j)_{j\in[n]}$, then for any $i \in [m]$ we have:

$$\mathsf{H.Eval}(\mathsf{crs},\boldsymbol{\delta}_i) = [\mathbf{g}^\mathsf{T}]\cdot((\mathbf{I}\|\boldsymbol{\gamma})\cdot\mathbf{m}_i) = ([\mathbf{g}^\mathsf{T}],h)\cdot\mathbf{m}_i = \tilde{h}_i.$$

Hence, in what follows, we can assume that whenever a tampering query is such that the hash values $(\tilde{h}_i)_{i\in\mathcal{T}}$ are all the same, then also the pre-images $(\boldsymbol{\delta}_i)_{i\in\mathcal{T}}$ are the same. We now present an adversary $\hat{\mathsf{A}}$ attacking leakage-resilient continuous non-malleability of the underlying secret sharing scheme. The adversary is almost identical to the corresponding one in the proof of Lemma 11, the only difference being in how it handles the tampering queries from the adversary A:

- **Tampering queries.** Upon input a tampering query $(\mathcal{T}, (f_1, \ldots, f_m))$, execute the following steps.
  - *Obtain the tampered hash.* Identical to the corresponding step in the proof of Lemma 11.
  - *Obtain the tampered ciphertext.* For all $i \in [m]$, construct the function $\hat{h}'_i$ which also hard-wires the value $\tilde{h}$ and all the previous leakage and tampering queries, takes as input the shares $(\kappa_j)_{j\in\mathcal{B}_i}$, runs all the hard-wired leakage and tampering functions, and it computes $(\boldsymbol{\delta}_j)_{j\in\mathcal{B}_i} = (\mathbf{I}\|\boldsymbol{\gamma}) \cdot \mathbf{M}^{(i)}$. If $\exists i_1, i_2 \in \mathcal{B}_i \cap \mathcal{T} : \boldsymbol{\delta}_{i_1} \neq \boldsymbol{\delta}_{i_2}$ the function outputs $\perp$, else let $\tilde{\boldsymbol{\gamma}}$ be this unique value and output $\tilde{\boldsymbol{\gamma}}$. Run procedure $\mathsf{LeakTamper}_{\mathcal{T}}(\hat{h}'_1, \ldots, \hat{h}'_m)$ to obtain either the tampered ciphertext $\tilde{\boldsymbol{\gamma}}$ or $\perp$.
  - *Check that everything is correct.* Identical to the corresponding step in the proof of Lemma 11.
  - *Obtain the tampered key.* Identical to the corresponding step in the proof of Lemma 11.
  - *Obtain the tampered message.* Identical to the corresponding step in the proof of Lemma 11.

□

The next lemma concludes the proof of Theorem 9. We omit the proof, as it is almost identical to the proof of Lemma 12.

Lemma 14. *For all $\mu_0, \mu_1 \in \mathcal{M}$, and all $(t-1)$-sized partitions of $[n]$, it holds that:*

$$\{\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, 0)\}_{\lambda\in\mathbb{N}} \overset{\mathsf{c}}{\approx} \{\mathbf{Hyb}^{\mu_0,\mu_1}_{\Pi^*,\mathsf{A},\mathcal{B}}(\lambda, 1)\}_{\lambda\in\mathbb{N}}.$$

□