Stronger Leakage-Resilient and Non-Malleable Secret-Sharing Schemes for General Access Structures

Divesh Aggarwal[3], Ivan Damgård[1], Jesper Buus Nielsen[1], Maciej Obremski[3], Erick Purwanto[3], João Ribeiro[2], and Mark Simkin[1]

[1]Aarhus University,
{ivan, jbn, simkin}@cs.au.dk
[2]Imperial College London,
j.lourenco-ribeiro17@imperial.ac.uk
[3]National University of Singapore,
{divesh, erickp}@comp.nus.edu.sg, obremski.math@gmail.com

Abstract

In this work we present a collection of compilers that take secret sharing schemes for an arbitrary access structures as input and produce either leakage-resilient or non-malleable secret sharing schemes for the same access structure. A leakage-resilient secret sharing scheme hides the secret from an adversary, who has access to an unqualified set of shares, even if the adversary additionally obtains some size-bounded leakage from *all* other secret shares. A non-malleable secret sharing scheme guarantees that a secret that is reconstructed from a set of tampered shares is either equal to the original secret or completely unrelated. To the best of our knowledge we present the first generic compiler for leakage-resilient secret sharing for general access structures. In the case of non-malleable secret sharing, we strengthen previous definitions, provide separations between them, and construct a non-malleable secret sharing scheme for general access structures that fulfills the strongest definition with respect to independent share tampering functions. More precisely, our scheme is secure against *concurrent tampering*: The adversary is allowed to (non-adaptively) tamper the shares multiple times, and in each tampering attempt can freely choose the qualified set of shares to be used by the reconstruction algorithm to reconstruct the tampered secret. This is a strong analogue of the multiple-tampering setting for split-state non-malleable codes and extractors.

We show how to use leakage-resilient and non-malleable secret sharing schemes to construct leakage-resilient and non-malleable threshold signatures. Classical threshold signatures allow to distribute the secret key of a signature scheme among a set of parties, such that certain qualified subsets can sign messages. We construct threshold signature schemes that remain secure even if an adversary leaks from or tampers with all secret shares.

1 Introduction

In a *secret sharing scheme*, a dealer who holds a secret s chosen from a domain $\mathcal{M}$ can compute a set of *shares* by evaluating a randomized function on s which we write as $\mathbf{Share}(s) = (s_1, \ldots, s_n)$.

A secret sharing comes with an *access structure* $\mathcal{A}$, which is a family of subsets of the indices $1, \ldots, , n$, such that if one is given a subset of the shares of s corresponding to a set $A \in \mathcal{A}$ (a *qualified* set), then one can compute s efficiently, whereas any subset of shares corresponding to a set not in $\mathcal{A}$ (an *unqualified* set) contains no, or almost no information about the secret. An important special case is *threshold* secret sharing, where the access structure contains all set of size at least some threshold value.

Secret-sharing is one of the most basic and oldest primitives in cryptography, introduced by Shamir and Blakely in the 70-ties. It allows to strike a meaningful balance between availability and confidentiality of secret information. Namely, we can store the n shares in n different servers and as long as a qualified set of servers is alive, the secret is available, but even if an unqualified set of shares is stolen, the secret remains confidential.

After its introduction, several variants of secret sharing have been suggested that address the problem of authenticity of the secret: we want to guarantee that we reconstruct the original value, even if not all players are honest. One such variant is *robust* secret-sharing where the dealer is honest, but some unqualified set of share holders are malicious and may return incorrect shares. In *verifiable* secret-sharing the dealer may be dishonest as well, but via interaction in the sharing phase we can enforce that a unique secret is still determined and that this is the value that will be reconstructed later.

In all these older schemes, the adversary is of the classic type that completely corrupts a certain subset of the players in the protocol, either to steal information or to corrupt data. Whereas the players who are not corrupt are "completely honest". In many scenarios, however, this may not be the most realistic model of attacks. Instead, it may make more sense to assume that the adversary will try to attack all share holders, and will have some partial success in all or most of the cases.

For the case of attacks against confidentiality, we can model this as *leakage resilient* secret-sharing, where the adversary is allowed to specify a leakage function Leak and will be told the value $\mathsf{Leak}(s_1, ..., s_n)$. Then, under certain restrictions on Leak, we want that the adversary learns essentially nothing about s. Typically, so called *local leakage* is considered, where $\mathsf{Leak}(s_1, ..., s_n) = (\mathsf{Leak}_1(s_1), ..., \mathsf{Leak}_n(s_n))$ for local leakage functions Leak_i with bounded output size. This makes sense in a scenario where shares are stored in physically separated locations. It is known that some secret-sharing schemes are naturally leakage-resilient against local leakage whereas others are not [BDIR18]. Boyle et al. [BGK14] showed how to construct (locally) leakage-resilient verifiable secret sharing for threshold access structures. Goyal and Kumar [GK18a] construct a specific type of leakage-resilient 2-out-of-n secret sharing as part of non-malleable secret sharing construction. To the best of our knowledge, it is not known how to construct leakage-resilient schemes from regular secret sharing schemes in general.

The case of attacks that try to corrupt the secret has been considered only recently, and for this purpose the notion of *non-malleable secret-sharing* was introduced by Goyal and Kumar [GK18a]. In this model, the adversary specifies a tampering function f which acts on the shares, and then the reconstruction algorithm is applied to a qualified subset of $f(s_1, ..., s_n)$. The demand, simplistically speaking, is that either the original secret is reconstructed or it is destroyed, i.e., the reconstruction result is unrelated to the original secret. Note that since f is allowed to touch all shares, we cannot avoid the case where an unrelated secret is reconstructed, as f could always replace all shares by shares of a different secret. In line with all previous works, we consider local tampering functions, that individually tamper with each share. This is a sensible assumption if, for example, each share is stored in a different server. Of course, such a tampering is closely related to the earlier notion of non-malleable codes against split-state tampering [DPW10]. The main difference between non-malleable codes and secret-sharing schemes is that, in addition to non-malleability, we also insist that the correctness and privacy properties of the secret-sharing scheme are satisfied. Interestingly, some non-malleable codes can also be seen as primitive versions of general non-malleable secret-sharing schemes. In fact, non-malleable codes in the 2-split-state model (where each codeword is split into two halfs which are tampered independently) are 2-out-of-2 non-malleable secret-sharing schemes [ADKO15].

The first non-malleable secret sharing schemes were constructed in [GK18a] for threshold access structures, and, in a follow-up work [GK18b], for general access structures, where an adversary is allowed to independently tamper with each share in a minimal reconstruction set. In the latter work, a general compiler was given that builds a non-malleable secret sharing scheme from a regular secret-sharing scheme.

An application of non-malleable secret-sharing to secure message transmission was given in [GK18a], but another very natural application, which does not seem to have been considered before, is to threshold cryptography. Let us consider, for instance, a threshold signature scheme. In such an application, the secret key is secret-shared among n servers, who then collaborate to generate a signature such that the signature itself is the only new information released.

Now, some threshold signature schemes have some "bult-in" protection against tampering. Namely, they establish a public commitment to each share of the secret key, and when a server contributes to a new signature, it must prove in zero-knowledge that it is behaving consistently with the commitment. If the commitment cannot be tampered, this will imply that tampered shares cannot contribute to a signature. However, in many protocols for signature generation, one can avoid zero-knowledge proofs by optimistically generating a signature assuming that all players behave correctly. The observation is that one can always verify the signature in the end and take some alternative action if it fails. This will be very efficient if players behave honestly almost always. Such a protocol is not secure if executed on tampered shares, and adding zero-knowledge proofs does not make sense in this case.

It therefore seems natural to try to use a non-malleable secret-sharing scheme instead. This of course raises the question of how we can generate signatures efficiently and securely – existing threshold signatures assume regular secret sharing, and it is not clear how we can use existing non-malleable schemes without resorting to generic multiparty computation.

However, suppose for a moment that we could solve this issue. Now, if the shares have in fact been tampered with, this tampering will become clear once it is found out that the signature does not verify, and one can then take action (e.g., stop the system and restore the secret key from a back-up). The intuition now is that we have managed to make the tampering harmless, because non-malleability implies that the faulty signature is generated from an unrelated secret.

Unfortunately, however, the original definition is unlikely to be sufficient to prove this intuition for a realistic system. The problem is that a real-life system will most likely have to serve many different signature requests that arrive in an uncoordinated fashion over an asynchronous network like the Internet. Therefore, once the first faulty signature has been detected and action has been taken, we should assume that in the mean time several other signature requests have already been served, possibly by different qualified sets of servers.

The standard definition of non-malleable secret-sharing [GK18a, GK18b] is not sufficient to prove security in this case because it only talks about one invocation of the reconstruction algorithm. What we need is a stronger definition, namely *non-malleability with concurrent reconstruction*. In this model, we consider an experiment where, after the tamperings have been done, the reconstruction algorithm is run (in parallel) on *several qualified subsets*. We require that all the instances of the reconstruction return either the original secret or something unrelated. It is not known how to construct secret-sharing schemes with this stronger property.

Independent Work In the late stages of this work, it came to our knowledge that other independent, concurrent works obtained results similar to ours:

- Srinivasan and Vasudevan [SV18] give a compiler that transforms a secret sharing scheme for any access structure into a leakage-resilient secret-sharing schemes for the same access structure. Their compiler is rate-preserving and has leakage rate approaching 1. In comparison, if the underlying secret sharing scheme is constant rate, our leakage-resilient secret sharing compiler achieves rate $\Omega(1/n)$ and leakage rate $1-c$ for an arbitrarily small constant $c > 0$, and must be rate-0 if we require leakage rate $1 - o(1)$. Srinivasan and Vasudevan also construct leakage resilient schemes in a stronger leakage model, where leakage functions may be chosen adaptively.

  Srinivasan and Vasudevan use the results obtained to construct positive rate non-malleable threshold secret sharing schemes against a single tampering that modifies each share independently for 4-monotone access structures[1]. In comparison, the non-malleable secret sharing compiler that we obtain for a single tampering works for all 3-monotone access

[1]An access structure $\mathcal{A}$ is said to be *k-monotone* if $|T| \geq k$ for all $T \in \mathcal{A}$.

structures but has rate $\Theta(\frac{1}{n \log m})$ in the same setting, where m denotes the length of the secret and n denotes the number of parties, and so converges to 0. Finally, they consider applications to leakage-resilient secure multiparty computation.

- Badrinarayanan and Srinivasan [BS18] construct non-malleable secret sharing schemes with respect to independent share tampering, both against a single tampering and against multiple tamperings. They are able to realize all 4-monotone access structures. Moreover, they optimize the rates of their constructions to obtain schemes with positive rate and a concretely efficient scheme. However, their tampering model is weaker than ours: While in our model, named *concurrent reconstruction*, the adversary is allowed to (non-adaptively) tamper the shares multiple times and in each tampering can choose a potentially different reconstruction set for the tampering experiment, the model studied in [BS18] forces the adversary to always choose the same reconstruction set for all tamperings. Their schemes are not secure in the stronger concurrent reconstruction model, and the authors explicitly mention the concurrent reconstruction model as a natural strengthening of their tampering model. In contrast, our compiler transforms any secret sharing scheme realizing a 3-monotone access structure into a (rate-0) non-malleable secret sharing scheme secure against multiple tamperings in the concurrent reconstruction model.
- Kumar, Meka, and Sahai [KMS18] also study leakage-resilient and non-malleable secret sharing. They consider a stronger leakage model than ours, where each leaked bit may depend on up to p shares which can be chosen adaptively by the adversary. They give a compiler that transforms a standard secret sharing scheme into a leakage-resilient one in the model just described, for p logarithmic in the number of parties. It is also shown that noticeably improving the dependence of the share length on p obtained there would lead to non-trivial progress on important open questions related to communication complexity. Finally, they consider the notion of *leakage-resilient non-malleable* secret sharing with respect to independent share tampering. Here, the adversary has access to leakage from the shares, which he can then make use of to choose tampering functions. They construct schemes in this model for the case of a single tampering. For comparison, our non-malleable secret sharing schemes cannot withstand leakage, but, as already mentioned in the previous paragraph, allow the adversary to tamper the shares multiple times, each time with a potentially different reconstruction set in the associated tampering experiment.

1.1 Our Contributions

In this paper, we resolve all of the above open questions:

- We present a general compiler that transforms any secret sharing scheme into a *leakage-resilient* one for the same access structure and preserves the efficiency of the original scheme. The compiled scheme withstands bounded size local leakage from all shares. The result extends to attacks that are strictly stronger than previously considered: the adversary can be told complete information on an unqualified set of shares and can in addition be given local leakage from all the other shares, and still will not learn the secret. To the best of our knowledge, this is the first result of its kind.

  If the share length of the underlying secret-sharing scheme is ℓ, then the compiler can yield a leakage-resilient scheme with shares of length $O(\ell)$ and leakage rate $1 - c$ for an arbitrarily small constant $c > 0$. Moreover, if we allow a blow-up of the share length in the compiled scheme from ℓ to $\omega(\ell)$, then we can achieve a leakage rate of $1 - o(1)$.
- We present another compiler that transforms any secret sharing scheme realizing an access structure $\mathcal{A}$ where every qualified set T has size at least 3 into a scheme for the same access structure that is *non-malleable with concurrent reconstruction* with respect to individual share tampering. More precisely, the adversary chooses a polynomial (in the number of parties) number of qualified sets $T_1, T_2, \dots$, where it may be the case that $T_i = T_j$ for some i and j, along with associated tampering functions $f^{(1)}, f^{(2)}, \dots$, where $f^{(i)}$ tampers each

share independently. We may think of this setting as a strong analogue of the multiple-tampering paradigm for non-malleable codes and extractors: The adversary is allowed to (non-adaptively) tamper the shares multiple times, and in each tampering attempt is further allowed to freely choose the qualified set to be used by the reconstruction algorithm in the tampering experiment.

- We present a compiler that turns any threshold signature scheme into one that is secure against tampering, assuming the original scheme is secure in the standard sense. In particular, the compiled scheme is secure even if faulty signatures are constructed from several qualified sets after tampering. We allow the adversary to either tamper with all shares of the secret key, or to maliciously corrupt an unqualified subset of the signature servers. The compiler adds two rounds to the signing protocol of the original scheme. The computational complexity is essentially that of the original signature protocol plus that of the reconstruction in a non-malleable secret-sharing scheme. The overhead is actually only necessary each time the system is initialized from storage that may have been tampered, and therefore its cost amortizes over all signatures generated while the system is on-line.
- We present a compiler that turns any threshold signature scheme into one that is secure in the standard sense even if the adversary, additionally, obtains size-bounded leakage from *all* secret key shares. The compiler follows the same blueprint and is as efficient as our compiler for non-malleable threshold signatures.

1.2 Technical Overview

In this section, we give a high-level overview of the proof ideas and techniques used to construct each one of our compilers.

All of our secret-sharing scheme compilers are based on the same key idea: Let $s_1, \ldots, s_n$ denote the shares obtained via the underlying secret-sharing scheme. We encode each share s_i using some (randomized) coding scheme $(\mathbf{Enc}, \mathbf{Dec})$ to obtain two values L_i and R_i. Then, the new compiled shares are obtained by, for each $i = 1, \ldots, n$, giving L_i to the i-th party, and R_i to every other party. At the end of this procedure, the i-th party has a compiled share, denoted S_i, of the form

$$S_i = (R_1, \ldots, R_{i-1}, L_i, R_{i+1}, \ldots, R_n)\,.$$

Reconstruction of the underlying secret is possible from any qualified set of parties, as they will learn the corresponding pairs (L_i, R_i), and hence the underlying share s_i. The different compilers arise by instantiating the idea above with coding schemes satisfying different properties. One basic property that is required from all coding schemes is that one half of the codeword (L_i, R_i) reveals almost nothing about s_i.

1.2.1 Leakage-Resilient Secret-Sharing Scheme

In order to obtain a leakage-resilient secret-sharing scheme via the idea above, we instantiate the coding scheme $(\mathbf{Enc}, \mathbf{Dec})$ as follows: Let Ext be a strong seeded extractor. Roughly speaking, a strong seeded extractor is a deterministic function that produces a close-to-uniform output when given a sample from a source with high min-entropy along with a short, independent, and uniform seed, *even when the seed is known to the distinguisher.* Then, $\mathbf{Enc}(m)$ samples (L, R) from the preimage $\mathsf{Ext}^{-1}(m)$ uniformly at random. Here, L corresponds to the weak source, while R corresponds to the uniform, independent seed. To recover m from a codeword c, we simply set $\mathbf{Dec}(L, R) := \mathsf{Ext}(L, R)$. This coding scheme is efficient if Ext is itself efficient, and furthermore Ext supports *efficient preimage sampling.* More precisely, this means that, given m, there exists an efficient algorithm that samples an element of $\mathsf{Ext}^{-1}(m)$ uniformly at random. The idea behind this coding scheme is the same as the one used by Cheraghchi and Guruswami [CG14] in order to obtain split-state non-malleable codes from non-malleable extractors (variations of these objects are defined in Section 2, but are not important for this discussion).

We note that a construction of a strong seeded extractor with near-optimal parameters which additionally supports efficient preimage sampling was given in [GUV09]. We instantiate our compiler with this extractor coupled with a careful choice of parameters in order to obtain a leakage-resilient scheme with good leakage rate.

We now discuss why this construction is leakage-resilient. For simplicity, assume that L_i and R_i are independent and uniform for $i = 1, \ldots, n$. This is not true in practice, and a little more care is needed to show that leakage-resilience holds in Section 4. However, it lets us present the main idea behind the proof in a clearer way.

Suppose the adversary holds shares from a set of unqualified parties T. Without loss of generality, let $T = \{1, \ldots, t\}$. Furthermore, we also assume the adversary learns some limited information about all shares, i.e., he learns $\mathsf{Leak}_i(S_i)$ for some function Leak_i and all $i = 1, \ldots, n$. Note that the adversary knows the pairs (L_i, R_i) for $i = 1, \ldots, t$, and hence the shares $s_1, \ldots, s_t$ obtained via the underlying secret-sharing scheme. Furthermore, he knows R_i (the seeds of the extractor) for $i = t+1, \ldots, n$. The goal of the adversary is now to obtain extra knowledge about $L_{t+1,}, \ldots, L_n$ from the leaked information. Since, by hypothesis, the leaked information about L_i is only a small linear fraction of its length, and is independent of R_i, we can condition L_i on the output of $\mathsf{Leak}_i(S_i)$. As a result, L_i conditioned on $\mathsf{Leak}_i(S_i)$ is still independent of R_i, and still has high min-entropy. This means that the output of $\mathsf{Ext}(L_i, R_i)$ still looks close-to-uniform to the adversary, even when R_i is given (recall that we use a strong extractor). It follows that the leaked information gives almost no information about the shares outside T, and hence we can use the statistical privacy of the underlying secret-sharing scheme to conclude the proof.

1.2.2 Non-Malleable Secret-Sharing Scheme with Concurrent Reconstruction

In order to obtain a non-malleable scheme, we use the same basic idea as before, but with a few modifications. To begin, we require the following primitives:

- A secret-sharing scheme $(\mathbf{Share}, \mathbf{Rec})$ for an access structure in which every qualified set has size at least 3;
- A strong two-source non-malleable extractor $\mathbf{nmExt}$ secure against multiple tamperings which supports *efficient preimage sampling*, in the sense that we can sample uniformly from its preimages $\mathbf{nmExt}^{-1}(z)$.

A non-malleable extractor is a stronger notion of an extractor introduced in [CG14]. More precisely, its output must still be close to uniform even conditioned on the output of the extractor on a tampered version of the original input. Similarly as before, such an extractor is said to be *strong* if the property above still holds when the distinguisher is also given the value of one of the input sources. Since their introduction, non-malleable extractors have received a lot of attention due to their connection to split-state non-malleable codes [CG14, CZ14, CGL16, Li17]. We note that constructions of such strong non-malleable extractors handling a sublinear (in the input length) number of tamperings and supporting efficient preimage sampling are known [CGL16, GKP+18].

The coding scheme $(\mathbf{Enc}, \mathbf{Dec})$ is obtained from $\mathbf{nmExt}$ analogously to the leakage-resilient scheme. Namely, $\mathbf{Enc}(m)$ samples (L, R) uniformly at random from $\mathbf{nmExt}^{-1}(m)$, and we set $\mathbf{Dec}(L', R') := \mathbf{nmExt}(L', R')$.

To encode the shares $(s_1, \ldots, s_n)$ into $(S_1, \ldots, S_n)$, we proceed as follows:

1. Sample $P \leftarrow \{0,1\}^p$;
2. Set $(L_i, R_i) \leftarrow \mathbf{Enc}(P||s_i)$ for $i = 1, \ldots, n$, where $||$ denotes string concatenation;
3. Set $S_i = (R_1, \ldots, R_{i-1}, L_i, R_{i+1}, \ldots, R_n)$ for $i = 1, \ldots, n$.

We will now briefly walk through the proof of statistical privacy and non-malleability for a single reconstruction set. Statistical privacy follows from the statistical privacy properties of the underlying secret-sharing scheme and the fact that $(\mathbf{Enc}, \mathbf{Dec})$ as defined above can be seen as a 2-out-of-2 secret-sharing scheme.

In order to show statistical privacy, fix an unqualified set of parties T, which we may assume is $T = \{1, \dots, t\}$. First, the fact that a split-state non-malleable code is also a 2-out-of-2 secret-sharing scheme implies that we can replace the values $R_{t+1}, \dots, R_n$ in all shares by independent and uniformly random values. Second, the pairs $(L_1, R_1), \dots, (L_t, R_t)$ encode shares $s_1, \dots, s_t$, respectively, belonging to an unqualified set of the underlying secret-sharing scheme. As a result, the statistical privacy of that scheme implies we can replace these encodings by those induced by a different secret.

In order to show non-malleability, fix a qualified set of parties T, with $t = |T| \geq 3$. For simplicity, assume again $T = \{1, \dots, t\}$. An adversary that wishes to tamper the shares in T chooses tampering functions $f_1, \dots, f_t$, one per share. Write a tampered share $S'_i = f_i(S_i)$ as

$$S'_i = (R'^{(i)}_1, \dots, R'^{(i)}_{i-1}, L'_i, R'^{(i)}_{i+1}, \dots, R'^{(i)}_n)$$

for $i = 1, \dots, t$. We now have the following reconstruction procedure, which may output a special symbol $\perp$ if it detects tampering:

1. For each $i = 1, \dots, n$, check that $R'^{(j_1)}_i = R'^{(j_2)}_i$ for all $j_1, j_2 \neq i$. If this is not the case, then output $\perp$;
2. If the check holds, set $R'_1 = R'^{(2)}_1$ and $R'_i = R'^{(1)}_i$ for $i = 2, \dots, t$. Then, decode and parse $P'_i || s'_i \leftarrow \mathbf{Dec}(L'_i, R'_i)$ for $i = 1, \dots, t$;
3. If $P'_i \neq P'_j$ for some $i, j \leq t$, output $\perp$. Else, output $\mathbf{Rec}^T(s'_1, \dots, s'_t)$.

Note that the consistency checks in Steps 1 and 3 correspond to properties that must be satisfied if $(S'_1, \dots, S'_t)$ is a valid set of shares. Roughly speaking, in order to show non-malleability we must be able to simulate the reconstruction of tampered shares without knowledge of the encoded secret m (except if the adversary does not modify any share, in which case we may output m).

We prove non-malleability in two steps. First, we consider the following *intermediate* tampering experiment on $(S_1, \dots, S_t)$:

- For each $i = 1, \dots, n$, check that $R'^{(j_1)}_i = R'^{(j_2)}_i$ for all $j_1, j_2 \neq i$. If this is not the case, then output $\perp$;
- If the check holds, set $R'_1 = R'^{(2)}_1$ and $R'_i = R'^{(1)}_i$ for $i = 2, \dots, t$. For each $i = 1, \dots, t$, set $\mathsf{output}_i = \mathsf{same}^*$ if $L'_i = L_i$ and $R'_i = R_i$. Otherwise, set $\mathsf{output}_i \leftarrow \mathbf{Dec}(L'_i, R'_i)$;
- If $\mathsf{output}_i = \mathsf{same}^*$ for all $i = 1, \dots, t$, output same^*. Else, output $(\mathsf{output}_1, \dots, \mathsf{output}_t)$.

This is an intermediate tampering experiment in the sense that it corresponds to a stage of the reconstruction procedure on the tampered shares where the values of the shares that remain the same have not yet been revealed. A key result we show is that the output of the intermediate tampering experiment described above has almost no correlation with the initial values $P || s_i$ for $i = 1, \dots, n$. In particular, we can replace each such value by an independent and uniformly random one, and hence by a set of uniform values independent of the secret m encoded by the shares $s_1, \dots, s_n$. We leverage a novel property of strong non-malleable extractors (Lemma 29) to prove this result, which may be of independent interest.

By the result just described, we now know how to simulate the intermediate tampering experiment for any secret m without any knowledge of m itself. However, to be able to simulate the behavior of the real reconstruction procedure on the tampered shares, we must know what the simulator must output when $\mathsf{output}_i = \mathsf{same}^*$ and $\mathsf{output}_j \neq \mathsf{same}^*$ for some $i, j \leq t$. In the second step, we show that the reconstruction procedure will output $\perp$ (i.e., tampering is detected, and hence the procedure is aborted) with high probability in this situation. This is because, with high probability, the decoded prefixes will not match among all parties in this case. As a result, we can simply have our simulator output $\perp$ in such a case, and it will coincide with the output of the real reconstruction procedure with high probability.

The argument above implies that our secret-sharing scheme is non-malleable against a single tampering of a reconstruction set. This result extends to the concurrent reconstruction setting, where the adversary is allowed to tamper the shares multiple times with different tampering functions and qualified sets. We refer to the later sections for details on the proof for the general case.

1.2.3 Threshold Signature Scheme Secure Against Tampering

Finally, our threshold signature compiler starts from the assumption that the secret key is to be secret-shared among a set of servers. We assume that we have protocols for generating n signature shares as well as a protocol for computing the final signature from these shares. Further, we assume that these protocols are secure even if an adversary maliciously corrupts an unqualified subset of size t of the $n \geq 2t+1$ servers.

To construct the compiled protocol, we first apply our second compiler from above, such that we now share the secret key using non-malleable secret sharing. Recall that this scheme involves encoding the original share s_i to get a pair (L_i, R_i) where the i-th server holds L_i and all other servers hold R_i. If now the i-th server wants to generate a signature share, it requests R_i from all other servers and waits until it gets back $n-t$ responses. If all received R_i are the same, it accepts the value and decodes (L_i, R_i) to obtain key share s_i. Note that since $n \geq 2t+1$ and the the server gets $n-t$ responses, we ensure that it gets back at least one honest response. At this point the server generates a signature share as it would do in the original protocol.

A rough intuition on why this is secure follows: Recall that our model says that the adversary can either tamper with the shares, or corrupt t of servers. If he tampers, he is not allowed to corrupt anyone, and this means that the servers are executing the non-malleable reconstruction protocol securely, and will either get the correct original shares (and thus create correct signatures) or will get something unrelated, in which case the output cannot compromise any secret key share. In the other case, the adversary has chosen to corrupt a set of servers. However, then we know that the shares we start from are correct. This means that sending the required R_i's in the clear to i-th server does not leak any extra information than it should. In fact, it merely enables the server to get his original share. The checks we enforce ensure that an honest player get its correct original share, and hence security follows from the threshold signature scheme we started with.

1.3 Open Questions

Several exciting questions remain open. The first natural direction is to improve the rates of our constructions. This can be achieved indirectly by coming up with better explicit constructions of strong seeded extractors and strong seedless non-malleable extractors. Another possibility is to improve the relationship between the share length of the compiled scheme and the number of parties. All of our constructions, as well as the constructions of Goyal and Kumar [GK18a, GK18b], have share sizes which are at least linear in the number of parties, and it would be interesting to see whether one can obtain a weaker dependence.

Our work introduces stronger definitions for non-malleable secret sharing schemes. However, our new notions, as well as the previous ones, are fundamentally non-adaptive in the sense that the tampering functions and reconstruction sets have to be chosen without seeing any of the shares a priori. We believe it would be more in the spirit of secret sharing if the tampering functions and reconstruction sets could be chosen *after* seeing some unqualified set of shares. On a similar note, a logical next step would be to define and attempt to construct continuous non-malleable secret-sharing schemes (in the spirit of [FMNV14]), where the adversary is allowed to choose the tampering function and qualified set to be reconstructed adaptively.

Our definition of leakage-resilient secret sharing schemes is also non-adaptive. It would be interesting to construct schemes which remain leakage resilient even if the adversary has access to an unqualified set of shares prior to choosing the leakage functions. Moreover, we obtain leakage rate $1-c$ for an arbitrarily small constant $c > 0$ while preserving the share length (up to a multiplicative constant). However, our share length suffers a polynomial blow-up if we want to achieve leakage rate $1-o(1)$. It would be interesting to give constructions of leakage-resilient schemes (even in the non-adaptive setting) with an improved tradeoff between leakage rate and share length.

1.4 Organization

The rest of the paper is organized as follows: We present notation, relevant definitions, and known lemmas that will prove useful in Section 2. We present and study our compiler for non-malleable secret-sharing in Section 3, along with separations between different definitions of non-malleable secret-sharing introduced in Section 2. In Section 4, we present our compiler for leakage-resilient secret-sharing. Finally, in Section 5, we discuss our compiler for non-malleable and leakage-resilient threshold signatures.

2 Preliminaries and Definitions

We denote the set $\{1, \ldots, n\}$ by $[n]$. We denote sets by calligraphic letters such as $\mathcal{A}$ and $\mathcal{M}$. We use the notation $z \leftarrow Z$ to denote that z is sampled according to distribution Z. If instead we write, say, $s \leftarrow \mathcal{S}$, this means that s is sampled uniformly at random from the set $\mathcal{S}$. Given an n-tuple x and a set $\mathcal{S} \subseteq [n]$ with $\mathcal{S} = \{i_1, \ldots, i_s\}$ and $i_j < i_{j+1}$ for $j = 1, \ldots, s-1$, we define $x_{\mathcal{S}} = (x_{i_1}, \ldots, x_{i_s})$. By an efficient algorithm, we mean an algorithm that runs in time polynomial in the length of the input.

2.1 Statistical Distance and Entropy

In this section, we define the statistical distance between two distributions and the min-entropy of a distribution, and we state some useful properties of these two quantities.

Definition 1 (Statistical Distance)**.** *Let X and Y be two distributions over a set S. The* statistical distance between X and Y*, denoted by $\Delta(X;Y)$, is given by*

$$\Delta(X;Y) := \max_{T \subseteq S}(|X(T) - Y(T)|) = \frac{1}{2}\sum_{s \in S} |X(s) - Y(s)| .$$

We say X is ε-close to Y, denoted $X \approx_\varepsilon Y$, if $\Delta(X;Y) \leq \varepsilon$, and we write $\Delta(X;Y|Z)$ as shorthand for $\Delta((X,Z);(Y,Z))$.

The following known properties of the statistical distance will be useful throughout the paper.

Lemma 2. *For any two random variables X and Y, and any randomized function f, we have that*

$$\Delta(f(X);f(Y)) \leq \Delta(X;Y) .$$

Lemma 3 ([CG14])**.** *Fix random variables X and Y such that*

$$X \approx_\varepsilon Y .$$

Let X' and Y' denote X and Y conditioned on an event E, respectively. If $X(E) = p$, then

$$X' \approx_{\varepsilon/p} Y'.$$

Definition 4 (Min-Entropy and Conditional Min-Entropy)**.** *Fix a distribution X over $\mathcal{X}$. The* min-entropy of X*, denoted by $\mathbf{H}_\infty(X)$, is given by*

$$\mathbf{H}_\infty(X) := -\log\left(\max_{x \in \mathcal{X}} X(x)\right) .$$

Moreover, the conditional min-entropy of X given Z*, denoted by $\mathbf{H}_\infty(X|Z)$, is given by*

$$\mathbf{H}_\infty(X|Z) := -\log\left(\mathbb{E}_{z \leftarrow Z}\left[2^{-\mathbf{H}_\infty(X|Z=z)}\right]\right) ,$$

where $\mathbb{E}_{z \leftarrow Z}$ denotes the expected value over Z.

The following property of the conditional min-entropy will be fundamental in later sections.

Lemma 5 ([DORS08]). *Let (X, Z) be some joint probability distribution. Then, if Z is supported on at most 2^ℓ values, we have*

$$\mathbf{H}_\infty(X|Z) \geq \mathbf{H}_\infty(X) - \ell\,.$$

2.2 Extractors and Non-Malleable Codes

In order to enforce our compilers have the desired properties, we will need to use some variants of extractors and non-malleable codes. We present the relevant definitions and results in this section.

Definition 6 (Coding Scheme). *A tuple of functions* $(\mathbf{Enc}, \mathbf{Dec})$*, where* $\mathbf{Enc} : \mathcal{M} \to \mathcal{C}$ *may be randomized but* $\mathbf{Dec} : \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ *is deterministic, is said to be a* coding scheme *if the correctness property*

$$\Pr(\mathbf{Dec}(\mathbf{Enc}(m)) = m) = 1$$

holds for every $m \in \mathcal{M}$, where the probability is taken over the randomness of the encoder $\mathbf{Enc}$*.*

Definition 7 (Non-Malleable Code [DPW10]). *We say that a coding scheme* $(\mathbf{Enc} : \mathcal{M} \to \mathcal{X} \times \mathcal{X},\ \mathbf{Dec} : \mathcal{X} \times \mathcal{X} \to \mathcal{M} \cup \{\perp\})$ *is* ε-non-malleable in the split-state model *if for all functions* $F, G : \mathcal{X} \to \mathcal{X}$ *there exists a distribution* $SD^{F,G}$ *over* $\mathcal{M} \cup \{\mathsf{same}^*, \perp\}$ *such that*

$$\mathsf{Tamper}_{\boldsymbol{m}}^{\boldsymbol{F,G}} \approx_\varepsilon \mathsf{Sim}_{\boldsymbol{m}}^{\boldsymbol{F,G}}$$

for all $m \in \mathcal{M}$, where

$$\mathsf{Tamper}_{\boldsymbol{m}}^{\boldsymbol{F,G}} = \begin{Bmatrix} (L, R) \leftarrow \mathbf{Enc}(m) \\ \textit{Output } \mathbf{Dec}(F(L), G(R)) \end{Bmatrix},$$

and

$$\mathsf{Sim}_{\boldsymbol{m}}^{\boldsymbol{F,G}} = \begin{Bmatrix} d \leftarrow SD^{F,G} \\ \textit{If } d = \mathsf{same}^*, \textit{ output } m \\ \textit{Else, output } d \end{Bmatrix}.$$

Additionally, $SD^{F,G}$ should be efficiently samplable given oracle access to $F(\cdot)$ and $G(\cdot)$.

We will also require a few variants of randomness extractors. We begin with the basic definition.

Definition 8 (Extractor). *An efficient function* $\mathsf{Ext} : \mathcal{X} \times \{0,1\}^d \to \mathcal{Z}$ *is an (average-case, strong)* (k, ε)-extractor*, if for all X, Z such that X is distributed over $\mathcal{X}$ and* $\mathbf{H}_\infty(X|Z) \geq k$*, we have*

$$\mathsf{Ext}(X; U_d), Z, U_d \approx_\varepsilon U_{\mathcal{Z}}, Z, U_d\,.$$

Moreover, we say Ext *supports* efficient preimage sampling *if, given $z \in \mathcal{Z}$, there exists an efficient algorithm that samples an element of* $\mathsf{Ext}^{-1}(z)$ *uniformly at random.*

In [GUV09], extractors with almost optimal parameters were constructed.

Lemma 9 ([GUV09]). *For any $\delta > 0$, there exists an efficient (k, ε)-extractor* $\mathsf{Ext} : \{0,1\}^N \times \{0,1\}^d \to \{0,1\}^\ell$ *for $\ell = k(1-\delta)$, $k \geq C(\log N + \log(1/\varepsilon))$, and $d = C(\log N + \log(1/\varepsilon))$, for some universal constant C. Moreover, this extractor supports efficient preimage sampling.*

We will also need a stronger notion of an (independent-source) extractor, for which the output still looks uniform even conditioned on the output of the extractor on a tampered version of the original input.

Definition 10 (Strong Two-Source Non-Malleable Extractor). *A function* **nmExt** : $\mathcal{X}^2 \to \mathcal{Z}$ *is said to be a* (k, ε, τ) strong two-source non-malleable extractor *if the following property holds: For independent distributions* X, Y *over* $\mathcal{X}$ *such that* $\mathbf{H}_\infty(X), \mathbf{H}_\infty(Y) \geq k$*, and for all tampering functions* $(f_1, g_1), \ldots, (f_\tau, g_\tau)$ *it holds that*

$$\mathbf{nmExt}(X, Y), Y, \{\mathcal{D}_{f_i,g_i}(X, Y)\}_{i\in[\tau]} \approx_\varepsilon U_\mathcal{Z}, Y, \{\mathcal{D}_{f_i,g_i}(X, Y)\}_{i\in[\tau]},$$

where $\mathcal{D}_{f,g}(X, Y)$ *is defined as*

$$\mathcal{D}_{f,g}(X, Y) := \begin{cases} \mathsf{same}^*, & \text{if } f(X) = X \text{ and } g(Y) = Y, \\ \mathbf{nmExt}(f(X), g(Y)), & \text{otherwise.} \end{cases}$$

The function **nmExt** *is said to support* efficient preimage sampling *if, given* $z \in \mathcal{Z}$*, there is an efficient algorithm that samples an element of the preimage* $\mathbf{nmExt}^{-1}(z)$ *uniformly at random.*

There exist explicit constructions of strong two-source non-malleable extractors with good parameters, supporting efficient preimage sampling, both against single and multiple tamperings [CGL16, Li17]. Although it is not stated in [CGL16] that the extractor found there is strong, it is known that this property does hold [Kum18]. A statement and proof of this result appears in [GKP$^+$18]. We will use the following two explicit non-malleable extractors.

Lemma 11 ([Li17]). *For any field* $\mathbb{F}$ *of cardinality* 2^N*, there exists a constant* $\delta \in (0, 1)$ *and a function* $\mathbf{nmExt} : \mathbb{F}^2 \to \{0, 1\}^\ell$ *such that* **nmExt** *is an efficient* $((1-\delta)N, \varepsilon, 1)$ *strong two-source non-malleable extractor with* $\ell = \Omega(N)$ *and* $\varepsilon = 2^{-\Omega(N/\log N)}$.

Lemma 12 ([CGL16, GKP$^+$18]). *For any field* $\mathbb{F}$ *of cardinality* 2^N*, there exists a constant* $\delta \in (0, 1)$ *and a function* $\mathbf{nmExt} : \mathbb{F}^2 \to \{0, 1\}^\ell$ *such that* **nmExt** *is an efficient* $(N - N^\delta, \varepsilon, \tau)$ *strong two-source non-malleable extractor with* $\ell = N^{\Omega(1)}$, $\tau = N^{\Omega(1)}$*, and* $\varepsilon = 2^{-N^{\Omega(1)}}$.

The connection between non-malleable extractors with efficient preimage sampling and split-state non-malleable codes is made clear by the following result.

Lemma 13 ([CG14]). *Fix an explicit two-source* $(n, \varepsilon, 1)$*-non-malleable extractor* $\mathbf{nmExt} : \mathbb{F}^2 \to \{0, 1\}^\ell$ *that supports efficient preimage sampling. The coding scheme* (**NMEnc**, **NMDec**) *is defined as follows:*

- $\mathbf{NMEnc}(m)$*: Sample* $(L, R) \leftarrow \mathbf{nmExt}^{-1}(m)$*, and output* (L, R)*;*
- $\mathbf{NMDec}(L', R')$*: Output* $\mathbf{nmExt}(L', R')$.

Then, (**NMEnc**, **NMDec**) *is an efficient split-state* ε'*-non-malleable code for* $\varepsilon' = \varepsilon(2^\ell + 1)$.

Combining Li's non-malleable extractor [Li17] and Lemma 13 immediately leads to the following result, also found in [Li17].

Corollary 14 ([Li17]). *For any field* $\mathbb{F}$ *of cardinality* 2^N*, there exists an efficient split-state* ε*-non-malleable code* (**NMEnc**, **NMDec**) *with* $\mathbf{NMEnc} : \{0, 1\}^\ell \to \mathbb{F}^2$, $\mathbf{NMDec} : \mathbb{F}^2 \to \{0, 1\}^\ell \cup \{\perp\}$, $\ell = \Theta(N/\log N)$*, and* $\varepsilon = 2^{-\Omega(N/\log N)}$.

2.3 Secret-Sharing Schemes

In this section, we introduce our definitions of leakage-resilient and non-malleable secret-sharing schemes. We begin with basic secret sharing concepts.

Definition 15 (Access Structure). *We say* $\mathcal{A}$ *is an* access structure for n parties *if* $\mathcal{A}$ *is a monotone class of subsets of* $[n]$*, i.e., if* $A \in \mathcal{A}$ *and* $A \subseteq B$*, then* $B \in \mathcal{A}$*. We call sets* $T \in \mathcal{A}$ authorized *or* qualified*, and* unauthorized *or* unqualified *otherwise.*

Definition 16 (Secret Sharing Scheme [Bei11]). *Let* $\mathcal{M}$ *be a finite set of secrets, where* $|\mathcal{M}| \geq 2$*. A (randomized) sharing function* $\mathbf{Share} : \mathcal{M} \to \mathcal{S}_1 \times \cdots \times \mathcal{S}_n$ *is an* (n, ε)-Secret Sharing Scheme for secret space $\mathcal{M}$ realizing access structure $\mathcal{A}$ *if the following two properties hold :*

1. ***Correctness.*** *The secret can be reconstructed by any authorized set of parties. That is, for any set $T \in \mathcal{A}$, where $T = \{i_1, \ldots, i_t\}$, there exists a deterministic reconstruction function $\mathbf{Rec}^T : \otimes_{i \in T} \mathcal{S}_i \to \mathcal{M}$ such that for every $m \in \mathcal{M}$,*

$$\Pr[\mathbf{Rec}^T(\mathbf{Share}(m)_T) = m] = 1\,,$$

   where the probability is taken over the randomness of **Share**.

2. ***Statistical Privacy.*** *Any collusion of unauthorized parties should have "almost" no information about the underlying secret. More formally, for all unauthorized sets $T \notin \mathcal{A}$ and for every pair of secrets $a, b \in \mathcal{M}$, we have*

$$\mathbf{Share}(a)_T \approx_\varepsilon \mathbf{Share}(b)_T.$$

We can additionally require that the unauthorized parties do not learn anything about the underlying secret, even if given some leakage from all the shares. This leads to the notion of *leakage-resilient* secret-sharing.

Definition 17 (Leakage-Resilient Secret-Sharing Scheme)**.** *A secret-sharing scheme* (**Share**, **Rec**) *realizing access structure $\mathcal{A}$ is said to be an* (n, ε, ρ)-leakage-resilient secret-sharing scheme *if the following property additionally holds:*

- ***Leakage-Resilient Statistical Privacy.*** *For all unauthorized sets $T \notin \mathcal{A}$, functions $\mathsf{Leak}_i : \mathcal{S}_i \to \{0,1\}^{\lfloor \rho \log |\mathcal{S}_i| \rfloor}$ for $i = 1, \ldots, n$, and for every pair of secrets $a, b \in \mathcal{M}$, we have*

$$\mathbf{Share}(a)_T, \{\mathsf{Leak}_i(\mathbf{Share}(a)_i)\}_{i \in [n]} \approx_\varepsilon \mathbf{Share}(b)_T, \{\mathsf{Leak}_i(\mathbf{Share}(b)_i)\}_{i \in [n]}\ .$$

Alternatively, we can require some security against tampering attacks on the shares produced by the secret-sharing scheme: Either the secret reconstructed from the tampered shares is the same as the original secret, or it is almost independent of it. The notion of *non-malleable* secret-sharing was first considered in [GK18a, GK18b], but only with respect to tampering attacks on qualified sets belonging to the minimal access structure.

Definition 18 (Non-Malleable Secret Sharing Scheme)**.** *Let* (**Share**, **Rec**) *be an (n, ε)-secret sharing scheme for secret space $\mathcal{M}$ realizing access structure $\mathcal{A}$. Let $\mathcal{F}$ be some family of tampering functions. For each $f \in \mathcal{F}$, $m \in \mathcal{M}$ and authorized set $T \in \mathcal{A}$, define the tampering experiment*

$$\mathsf{STamper}_{\boldsymbol{m}}^{\boldsymbol{f,T}} = \begin{Bmatrix} \mathbf{s} \leftarrow \mathbf{Share}(m) \\ \widetilde{\mathbf{s}} \leftarrow f(\mathbf{s}) \\ \widetilde{m} \leftarrow \mathbf{Rec}(\widetilde{\mathbf{s}}_T) \\ \textit{Output } \widetilde{m} \end{Bmatrix},$$

which is a random variable over the randomness of the sharing function **Share**. *We say that* (**Share**, **Rec**) *is* ε'-non-malleable with respect to $\mathcal{F}$ *if for each $f \in \mathcal{F}$ and authorized set $T \in \mathcal{A}$, there exists a distribution $SD^{f,T}$ (corresponding to the simulator) over $\mathcal{M} \cup \{\mathsf{same}^*, \perp\}$ such that we have*

$$\mathsf{STamper}_{\boldsymbol{m}}^{\boldsymbol{f,T}} \approx_{\varepsilon'} \mathsf{SSim}_{\boldsymbol{m}}^{\boldsymbol{f,T}}\,,$$

for all $m \in \mathcal{M}$ and authorized sets $T \in \mathcal{A}$, where

$$\mathsf{SSim}_{\boldsymbol{m}}^{\boldsymbol{f,T}} = \begin{Bmatrix} \widetilde{m} \leftarrow SD^{f,T} \\ \textit{If } \widetilde{m} = \mathsf{same}^*, \textit{ output } m \\ \textit{Else, output } \widetilde{m} \end{Bmatrix}.$$

Additionally, $SD^{f,T}$ should be efficiently samplable given oracle access to $f(\cdot)$.

We also consider a stronger notion of non-malleable secret-sharing, where the adversary is allowed to tamper the shares multiple times, and in each tampering attempt is free to choose the qualified set to be used by the reconstruction algorithm in the tampering experiment.

Definition 19 (Non-Malleable Secret Sharing Scheme with Concurrent Reconstruction)**.** *Let* $(\mathbf{Share}, \mathbf{Rec})$ *be an* (n, ε)*-secret sharing scheme for secret space* $\mathcal{M}$ *realizing access structure* $\mathcal{A}$*. Let* τ *be a fixed constant. Let* $\mathcal{F}$ *be some family of tampering functions. For* $m \in \mathcal{M}$*,* $\mathbf{f} = (f^{(1)}, \ldots, f^{(\tau)}) \in \mathcal{F}^\tau$*, and* $\mathbf{T} = (T_1, \ldots, T_\tau) \in \mathcal{A}^\tau$*, define the tampering experiment*

$$\mathsf{SCRTamper}_{\boldsymbol{m}}^{\mathbf{f},\boldsymbol{T}} = \left(\mathsf{STamper}_{\boldsymbol{m}}^{\boldsymbol{f}^{(1)},\boldsymbol{T}_1}, \mathsf{STamper}_{\boldsymbol{m}}^{\boldsymbol{f}^{(2)},\boldsymbol{T}_2}, \ldots, \mathsf{STamper}_{\boldsymbol{m}}^{\boldsymbol{f}^{(\tau)},\boldsymbol{T}_\tau}\right),$$

where each $\mathsf{STamper}_{\boldsymbol{m}}^{\boldsymbol{f}^{(i)},\boldsymbol{T}_i}$ *is defined as in Definition 18. We say that* $(\mathbf{Share}, \mathbf{Rec})$ *is* (ε', τ)- concurrent-reconstruction-non-malleable with respect to $\mathcal{F}$ *if for each tuple* $\mathbf{f} \in \mathcal{F}^\tau$ *and tuple of authorized sets* $\mathbf{T} \in \mathcal{A}^\tau$*, there exists a distribution* $SD^{\mathbf{f},\mathbf{T}}$ *over* $(\mathcal{M} \cup \{\perp, \mathsf{same}^*\})^\tau$ *such that*

$$\mathsf{SCRTamper}_{\boldsymbol{m}}^{\mathbf{f},\mathbf{T}} \approx_{\varepsilon'} \mathsf{SCRSim}_{\boldsymbol{m}}^{\mathbf{f},\mathbf{T}}$$

for all $m \in \mathcal{M}$*, where*

$$\mathsf{SCRSim}_{\boldsymbol{m}}^{\mathbf{f},\mathbf{T}} = \left\{ \begin{array}{l} (\widetilde{m}_1, \ldots, \widetilde{m}_\tau) \leftarrow SD^{\mathbf{f},\mathbf{T}} \\ \textit{Output } (\widetilde{m}'_1, \ldots, \widetilde{m}'_\tau), \textit{ where } \widetilde{m}'_i = m \textit{ if } \widetilde{m}_i = \mathsf{same}^*, \textit{ and } \widetilde{m}'_i = \widetilde{m}_i \textit{ otherwise} \end{array} \right\}.$$

Additionally, $SD^{\mathbf{f},\mathbf{T}}$ *should be efficiently samplable given oracle access to* $f^{(1)}(\cdot), \ldots, f^{(\tau)}(\cdot)$*.*

In this work, we will focus on the case where each share is tampered independently. With this in mind, we define the family of so-called t-split-state tampering functions, which we denote by $\mathcal{F}_t^{\mathrm{split}}$.

Definition 20 (t-Split-State Tampering Functions)**.** *The family of* t-split-state tampering functions over a domain $\mathcal{X}$*, denoted by* $\mathcal{F}_t^{split}$ *(the domain is ommitted for brevity), consists of all functions* $f : \mathcal{X}^t \to \mathcal{X}^t$ *for which there exist functions* $f_i : \mathcal{X} \to \mathcal{X}$ *with* $i \in [t]$ *such that*

$$f(x) = (f_1(x_1), \ldots, f_t(x_t)),$$

where $x = (x_1, \ldots, x_t)$ *and* $x_i \in \mathcal{X}$ *for* $i \in [t]$*.*

In particular, split-state tampering of non-malleable codes and extractors as in Definitions 7 and 10 corresponds to considering the family of tampering functions $\mathcal{F}_2^{\mathrm{split}}$.

The following result states that split-state non-malleable codes are 2-out-of-2 non-malleable secret-sharing schemes.

Lemma 21 ([ADKO15])**.** *Suppose* $(\mathbf{NMEnc}, \mathbf{NMDec})$ *is an* ε*-non-malleable code in the split-state model. Fix messages* m *and* m'*, and let* $(L, R) \leftarrow \mathbf{NMEnc}(m)$ *and* $(L', R') \leftarrow \mathbf{NMEnc}(m')$*. Then, we have*

$$L \approx_{2\varepsilon} L',$$

and

$$R \approx_{2\varepsilon} R'.$$

3 Non-Malleable Secret-Sharing

3.1 Separations between Notions of Non-Malleable Secret-Sharing

In this section, we show separations between the different notions of non-malleable secret sharing introduced in Section 2 and in [GK18b]. We recall the definition of non-malleable secret sharing for general access structures given in [GK18b].

Definition 22 (Minimal Access Structure). *Given an access structure $\mathcal{A}$, its* minimal access structure*, denoted by $\mathcal{A}^{\min}$, consists of all $T \in \mathcal{A}$ such that if $W \subsetneq T$, then $W \notin \mathcal{A}$.*

Definition 23 (Non-Malleable Secret-Sharing as in [GK18b]). *Let $(\mathbf{Share}, \mathbf{Rec})$ be an (n, ε)-secret sharing scheme for secret space $\mathcal{M}$ realizing access structure $\mathcal{A}$ with minimal access structure $\mathcal{A}^{\min}$. Let $\mathcal{F}$ be some family of tampering functions. For each $f \in \mathcal{F}$, $m \in \mathcal{M}$ and authorized set $T \in \mathcal{A}^{\min}$, define the tampering experiment*

$$\mathsf{STamper}_{\boldsymbol{m}}^{\boldsymbol{f},\boldsymbol{T}} = \begin{Bmatrix} \mathbf{s} \leftarrow \mathbf{Share}(m) \\ \widetilde{\mathbf{s}} \leftarrow f(\mathbf{s}) \\ \widetilde{m} \leftarrow \mathbf{Rec}(\widetilde{\mathbf{s}}_T) \\ \textit{Output } \widetilde{m} \end{Bmatrix},$$

which is a random variable over the randomness of the sharing function **Share**. *We say that* $(\mathbf{Share}, \mathbf{Rec})$ *is* ε-non-malleable with respect to $\mathcal{F}$ *if for each $f \in \mathcal{F}$ and authorized set $T \in \mathcal{A}^{\min}$, there exists a distribution $SD^{f,T}$ (corresponding to the simulator) over $\mathcal{M} \cup \{\mathsf{same}^*, \perp\}$ such that we have*

$$\mathsf{STamper}_{\boldsymbol{m}}^{\boldsymbol{f},\boldsymbol{T}} \approx_\varepsilon \mathsf{SSim}_{\boldsymbol{m}}^{\boldsymbol{f},\boldsymbol{T}},$$

for all $m \in \mathcal{M}$ and authorized sets $T \in \mathcal{A}$, where

$$\mathsf{SSim}_{\boldsymbol{m}}^{\boldsymbol{f},\boldsymbol{T}} = \begin{Bmatrix} \widetilde{m} \leftarrow SD^{f,T} \\ \textit{If } \widetilde{m} = \mathsf{same}^*, \textit{ output } m \\ \textit{Else, output } \widetilde{m} \end{Bmatrix}.$$

Additionally, $SD^{f,T}$ should be efficiently samplable given oracle access to $f(\cdot)$.

The difference between Definitions 18 and 23 is that in Definition 23 one only has to deal with reconstruction from minimal qualified sets $T \in \mathcal{A}^{\min}$. Our first result in this section states that there exist secret-sharing schemes for n parties which satisfy Definition 23 for $\mathcal{F} = \mathcal{F}_n^{\text{split}}$, but do not satisfy the stronger Definition 18.

Lemma 24. *There exists a secret-sharing scheme* $(\mathbf{AShare}, \mathbf{ARec})$ *for n parties satisfying Definition 23 with respect to $\mathcal{F} = \mathcal{F}_n^{split}$, but which does not satisfy Definition 18.*

Proof. Fix some secret-sharing scheme $(\mathbf{AShare}, \mathbf{ARec})$ for n parties satisfying Definition 23 with respect to $\mathcal{F} = \mathcal{F}_n^{\text{split}}$. Suppose that the corresponding secrets lie in some field $\mathbb{F}_0$, while shares lie in $\mathbb{F}_1$. Consider now the secret-sharing scheme $(\mathbf{SHARE}, \mathbf{REC})$, with secrets over $\mathbb{F}_0$ and shares over $\mathbb{F}_1 \cup \{\square\}$ for some special symbol $\square$, defined as follows:

- For a secret m, set $\mathbf{SHARE}(m) = \mathbf{AShare}(m)$.
- Given a possibly tampered set of shares $\widetilde{\mathbf{s}} = \widetilde{s}_1, \dots, \widetilde{s}_t$ corresponding to a qualified set of parties $T \in \mathcal{A}$, $\mathbf{REC}(\widetilde{\mathbf{s}})$ proceeds as follows:
  1. If $\widetilde{s}_i \neq \square$ for all i, output $\mathbf{ARec}(\widetilde{\mathbf{s}})$;
  2. Else, if there exist i such that $\widetilde{s}_i = \square$ and $T' \subseteq T$ satisfying $T' \in \mathcal{A}$ and $\widetilde{s}_j \neq \square$ for all $j \in T'$, then output
  $$\mathbf{ARec}(\widetilde{\mathbf{s}}_{T'}) + |\{i \in T : \widetilde{s}_i = \square\}|;$$
  3. Else, for each $i \in [t]$, if $\widetilde{s}_i = \square$, overwrite $\widetilde{s}_i \leftarrow 0 \in \mathbb{F}$. Reconstruct as $\mathbf{ARec}(\widetilde{\mathbf{s}})$.

It is clear that if $(\mathbf{AShare}, \mathbf{ARec})$ fullfills Definition 23, then so does $(\mathbf{SHARE}, \mathbf{REC})$. The reason is that, if $T \in \mathcal{A}^{\min}$, then we do not land on the second case of the reconstruction procedure $\mathbf{REC}(\widetilde{\mathbf{s}})$ above.

However, $(\mathbf{SHARE}, \mathbf{REC})$ does not fullfill Definition 18. In fact, fix a qualified set $T \in \mathcal{A}$ (again, for simplicity assume $T = \{1, \dots, t\}$) such that $T \neq [n]$. Let $T' = T \cup \{t+1\} \in \mathcal{A}$.

Consider the tampering functions $f_1, \ldots, f_t, f_{t+1}$ such that $f_1, \ldots, f_t$ are identity, and $f_{t+1}(x) = \Box$ for all $x \in \mathbb{F}_1$. Then,

$$\begin{aligned}\mathbf{REC}(\widetilde{s}_1, \ldots, \widetilde{s}_t, \widetilde{s}_{t+1}) &= \mathbf{REC}(s_1, \ldots, s_t, \Box) \\ &= \mathbf{ARec}(s_1, \ldots, s_t) + 1 \\ &= m + 1\,,\end{aligned}$$

which is clearly correlated with m. $\Box$

We now show that there exists a secret-sharing scheme satisfying Definition 18 which does not satisfy Definition 19, provided the number of parties is large enough. In words, such a scheme is non-malleable with respect to the reconstruction of a single arbitrary qualified set $T \in \mathcal{A}$, but is not non-malleable if one allows concurrent reconstruction of several qualified sets.

Lemma 25. *There exists a secret-sharing scheme* $(\mathbf{AShare}, \mathbf{ARec})$ *for n parties (with n large enough) satisfying Definition 18 with respect to $\mathcal{F} = \mathcal{F}_n^{split}$, but which does not satisfy Definition 19.*

Proof. Consider an access structure $\mathcal{A}$ over $2n$ parties such that $T \in \mathcal{A}$ if and only if $i, j \in T$ for some $i \leq n$ and $j > n$. In words, the set of parties is split into two halfs $\{1, \ldots, n\}$ and $\{n+1, \ldots, 2n\}$, and the qualified sets in $\mathcal{A}$ are exactly those that contain at least a party from each half of the parties. The secret-sharing scheme $(\mathbf{AShare}, \mathbf{ARec})$ requires a split-state non-malleable code $(\mathbf{NMEnc}, \mathbf{NMDec})$, and proceeds as follows:

- $\mathbf{AShare}(m)$:
  1. Set $(L, R) \leftarrow \mathbf{NMEnc}(m)$;
  2. Set $s_i = L$ for all $i \leq n$, and $s_i = R$ for all $i > n$.
- $\mathbf{ARec}(\widetilde{\mathbf{s}}_T)$ for a qualified set $T \in \mathcal{A}$:
  1. Find $i, j \in T$ such that $i \leq n$ and $j > n$. This is possible because $T \in \mathcal{A}$;
  2. Set $\widetilde{L} \leftarrow \widetilde{s}_i$ and $\widetilde{R} \leftarrow \widetilde{s}_j$;
  3. Set $\widetilde{m} \leftarrow \mathbf{NMDec}(\widetilde{L}, \widetilde{R})$.

The fact that $(\mathbf{AShare}, \mathbf{ARec})$ satisfies Definition 18 follows directly from the non-malleability of the underlying split-state code $(\mathbf{NMEnc}, \mathbf{NMDec})$.

To see that Definition 19 is not satisfied, consider an adversary that concurrently tampers all sets of the form $\{i, n+i\}$ for $i = 1, \ldots, n$. Equivalently, the adversary can tamper $\mathbf{NMEnc}(m)$ a total of n times in parallel. If $n \geq |L| + |R|$, then we can perform the attack described in [FMNV14, Section 3.1] in order to recover m, and thus break non-malleability. For completeness, we describe the attack here: First, we note that for any split-state non-malleable code, there exist $L^\star$ and $R_1^\star \neq R_2^\star$ such that $\mathbf{Dec}(L^\star, R_1^\star), \mathbf{Dec}(L^\star, R_2^\star) \neq \perp$, and $\mathbf{Dec}(L^\star, R_1^\star) \neq \mathbf{Dec}(L^\star, R_2^\star)$. An analogous property holds with the left encoding in place of the right encoding and vice-versa. We now show how to fully recover R with $|R|$ (non-adaptive) tamperings. For $i = 1, \ldots, |R|$, define the left tampering function F_i as $F_i(L) = L^\star$ for all L. Also, define the right tampering function G_i as

$$G_i(R) = \begin{cases} R_1^\star, & \text{if } R_i = 0, \\ R_2^\star, & \text{if } R_i = 1. \end{cases}$$

Then, we can recover R_i from the output of $\mathbf{Dec}(F_i(L), G_i(R))$, and hence we recover R completely. A similar procedure can be undertaken to recover L with an additional $|L|$ tamperings. $\Box$

3.2 Non-Malleable Secret-Sharing Scheme against Individual Tamperings

Before proceeding to the more general case of non-malleability with concurrent reconstruction, we describe our candidate secret-sharing scheme and prove it is non-malleable against a single tampering with respect to functions which tamper the shares independently.

Theorem 26. *Fix a number of parties n and an integer p. Furthermore, assume we have access to the following primitives:*

1. *For $\varepsilon_1 \geq 0$, let $(\mathbf{AShare}, \mathbf{ARec})$ be an (n, ε_1)-secret sharing scheme realizing an access structure $\mathcal{A}$ such that $|T| \geq 3$ holds whenever $T \in \mathcal{A}$. Suppose the corresponding shares lie in $\{0,1\}^r$ and the secrets in some set $\mathcal{M}$;*
2. *Let $\mathbf{nmExt} : \{0,1\}^N \times \{0,1\}^N \to \{0,1\}^\ell$ be the $((1-\delta)N, \varepsilon_2, 1)$ strong two-source non-malleable extractor from Lemma 11, where $\ell = r + p$. Hence, $\ell \leq \Omega(N)$ and $\varepsilon_2 = 2^{-\Omega(N/\log N)}$.*

Then, there exists an $(n, \varepsilon_1 + 4n\varepsilon_2(2^\ell + 1))$-secret sharing scheme realizing access structure $\mathcal{A}$ that is $n(2^{\ell+1}(\varepsilon_2 + 2^{-\delta N/2+1}) + 2^{-p})$-non-malleable w.r.t. $\mathcal{F}_n^{split}$. The resulting scheme $(\mathbf{NMShare}, \mathbf{NMRec})$ shares an element of $\mathcal{M}$ into n shares, where each share contains n elements of $\{0,1\}^N$. Finally, if the two primitives are efficient and the access structure $\mathcal{A}$ supports efficient membership queries, then the constructed scheme $(\mathbf{NMShare}, \mathbf{NMRec})$ is also efficient.

We describe our construction of the non-malleable secret sharing scheme $(\mathbf{NMShare}, \mathbf{NMRec})$.

NMShare: Our sharing function takes as input a secret $m \in \mathcal{M}$ and proceeds as follows:

1. Share m using **AShare** to obtain $s_1, \ldots, s_n \leftarrow \mathbf{AShare}(m)$;
2. Pick $P \leftarrow \{0,1\}^p$;
3. For each $i \in [n]$, encode the share s_i to obtain $(L_i, R_i) \leftarrow \mathbf{nmExt}^{-1}(P||s_i)$;
4. For each $i \in [n]$, construct $share_i = (R_1, \ldots, R_{i-1}, L_i, R_{i+1}, \ldots, R_n)$;
5. Output $(share_1, \ldots, share_n)$.

NMRec: Our reconstruction function takes as input shares $\{share_i : i \in T\}$ corresponding to an authorized set $T \in \mathcal{A}$ and proceeds as follows:

1. Sort T so that $T = \{i_1, \ldots, i_t\}$, where $t = |T|$, and $i_j < i_{j+1}$;
2. For each $j \in [t]$, parse the shares in T to obtain $(R_1^{(i_j)}, \ldots, R_{i_j-1}^{(i_j)}, L_{i_j}, R_{i_j+1}^{(i_j)}, \ldots, R_n^{(i_j)}) \leftarrow share_{i_j}$;
3. For every $\ell \in [n]$, check that the $R_\ell^{(i_j)}$ have the same value for all j such that $i_j \neq \ell$. If this is not the case, output $\perp$;
4. For every $j \in [t]$, decode and parse $P_{i_j}||s_{i_j} \leftarrow \mathbf{nmExt}(L_{i_j}, R_{i_j}^{(i_k)})$, where i_k is the smallest element of $T - \{i_j\}$;
5. If there exist $j, j' \in [t]$ such that $P_{i_j} \neq P_{i_{j'}}$, output $\perp$;
6. Else, reconstruct $m \leftarrow \mathbf{ARec}(s_{i_1}, \ldots, s_{i_t})$, and output m.

Correctness and Efficiency: Follows in a straightforward manner from the construction.

Statistical Privacy: Fix two secrets a and b, and let T be an unauthorized set of size t. Without loss of generality, we may assume that $T = \{1, 2, \ldots, t\}$. Set

$$aS_T \leftarrow \mathbf{NMShare}(a)_T\,,$$
$$bS_T \leftarrow \mathbf{NMShare}(b)_T\,.$$

Furthermore, let $as_1, \ldots, as_n$ and $bs_1, \ldots, bs_n$ be the shares obtained from $\mathbf{AShare}(a)$ and $\mathbf{AShare}(b)$, respectively, in Step 1 of the **NMShare** procedure.

Our goal is to show that the distributions of these two sets of shares, aS_T and bS_T, are close in statistical distance. More precisely, we will show that

$$aS_T \approx_{\varepsilon_1+4n\varepsilon_2(2^\ell+1)} bS_T$$

for all unauthorized sets T and secrets a, b.

We have $aS_T = (as_1, \dots, as_t)$ and $bS_T = (bs_1, \dots, bs_t)$, with

$$as_i = (aR_1, \dots, aR_{i-1}, aL_i, aR_{i+1}, \dots, aR_n)\,,$$
$$bs_i = (bR_1, \dots, bR_{i-1}, bL_i, bR_{i+1}, \dots, bR_n)\,.$$

As a result, we can write

$$aS_T = [(aL_i, aR_i)_{i\le t}, aR_{t+1}, \dots, aR_n]\,,$$
$$bS_T = [(bL_i, bR_i)_{i\le t}, bR_{t+1}, \dots, bR_n]\,.$$

Our first claim is that we can replace $aR_{t+1}, \dots, aR_n$ by encodings of independent, uniformly random messages with small penalty in statistical distance by invoking Lemma 21.

Lemma 27. *Let $R^*_{t+1}, \dots, R^*_n \in \mathbb{F}$ be sampled as follows: For each $j = t+1, \dots, n$, independently sample a uniformly random message m^*, encode and parse $(L^*, R^*) \leftarrow \mathbf{nmExt}^{-1}(m^*)$, and set $R^*_j = R^*$. Then,*

$$(aL_i, aR_i)_{i\le t}, aR_{t+1}, \dots, aR_n \approx_{2n\varepsilon_2(2^\ell+1)} (aL_i, aR_i)_{i\le t}, R^*_{t+1}, \dots, R^*_n\,.$$

Proof. We prove the lemma via a hybrid argument. Consider the following hybrids:

$\mathbf{Hybrid_0}$ Sample $aS_T \leftarrow \mathbf{NMShare}(a)$. Recall we may write $aS_T = [(aL_i, aR_i)_{i\le t}, aR_{t+1}, \dots, aR_n]$.

$\mathbf{Hybrid_1}$ Sample aS_T as in the previous hybrid $\mathbf{Hybrid_0}$. Replace aR_{t+1} by R^*_{t+1} sampled as in the lemma statement.

⋮

$\mathbf{Hybrid_{n-t}}$ Sample aS_T as in the previous hybrid $\mathbf{Hybrid_{n-t-1}}$. Replace aR_n by R^*_n sampled as in the lemma statement. Observe that the output of this hybrid is distributed exactly as $[(aL_i, aR_i)_{i\le t}, R^*_{t+1}, \dots, R^*_n]$.

It suffices now to see that

$$\mathbf{Hybrid_{j-1}} \approx_{2\varepsilon_2(2^\ell+1)} \mathbf{Hybrid_j}$$

for $j = 1, \dots, n-t$. Observe that aR_j is conditionally independent of $(aL_i, aR_i)_{i\ne j}$ given the prefix P and the share as_j. Therefore, we have

$$\begin{aligned}
&\Delta(\mathbf{Hybrid_{j-1}}; \mathbf{Hybrid_j})\\
&= \Delta([(aL_i, aR_i)_{i\le t}, R^*_{t+1}, \dots, R^*_{t+j-1}, aR_{t+j}, \dots, aR_n]; [(aL_i, aR_i)_{i\le t}, R^*_{t+1}, \dots, R^*_{t+j}, aR_{t+j+1}, \dots, aR_n])\\
&\le \Delta([(aL_i, aR_i)_{i\le t}, R^*_{t+1}, \dots, R^*_{t+j-1}, aR_{t+j}, \dots, aR_n]; [(aL_i, aR_i)_{i\le t}, R^*_{t+1}, \dots, R^*_{t+j}, aR_{t+j+1}, \dots, aR_n] | P, as_{t+j})\\
&= \Delta(aR_{t+j}; R^*_{t+j} | P, as_{t+j})\,, \qquad (1)
\end{aligned}$$

where the first inequality follows from the triangle inequality, and the second equality follows by conditional independence as previously stated. Now, note that aR_{t+j} is the right part of $\mathbf{nmExt}^{-1}(P||s_{t+j})$. Thus, since the coding scheme $(\mathbf{nmExt}^{-1}, \mathbf{nmExt})$ is an $(\varepsilon_2(2^\ell+1))$-non-malleable code by Lemma 13, it follows that Lemma 21 yields

$$aR_{t+j}, P, as_{t+j} \approx_{2\varepsilon_2(2^\ell+1)} R^*_{t+j}, P, as_{t+j}\,. \qquad (2)$$

Combining (1) and (2) leads to

$$\mathbf{Hybrid_{j-1}} \approx_{2\varepsilon_2(2^\ell+1)} \mathbf{Hybrid_j}\,,$$

as desired. □

Observe that, by the statistical privacy of the underlying secret sharing scheme, we have

$$\begin{aligned}&\Delta((aL_i, aR_i)_{i\leq t}; (bL_i, bR_i)_{i\leq t})\\ &\leq \Delta((aL_i, aR_i)_{i\leq t}; (bL_i, bR_i)_{i\leq t}|P)\\ &\leq \varepsilon_1\,, \end{aligned} \tag{3}$$

where P is the prefix used when encoding the shares with $\mathbf{nmExt}^{-1}$. This is because T is an unauthorized set, and each (aL_i, aR_i) (resp. (bL_i, bR_i)) depends on (aL_j, aR_j) (resp. (bL_j, bR_j)) for $j \neq i$ only through the share as_i or bs_i it encodes, when the prefix P is fixed. Combining Lemma 27 with (3) and a repeated application of the triangle inequality yields

$$\begin{aligned}\Delta(aS_T; bS_T) &= \Delta([(aL_i, aR_i)_{i\leq t}, aR_{t+1}, \ldots, aR_n]; [(bL_i, bR_i)_{i\leq t}, bR_{t+1}, \ldots, bR_n])\\ &\leq \Delta([(aL_i, aR_i)_{i\leq t}, aR_{t+1}, \ldots, aR_n]; [(aL_i, aR_i)_{i\leq t}, R^*_{t+1}, \ldots, R^*_n])\\ &\quad + \Delta([(aL_i, aR_i)_{i\leq t}, R^*_{t+1}, \ldots, R^*_n]; [(bL_i, bR_i)_{i\leq t}, R^*_{t+1}, \ldots, R^*_n])\\ &\quad + \Delta([(bL_i, bR_i)_{i\leq t}, R^*_{t+1}, \ldots, R^*_n]; [(bL_i, bR_i)_{i\leq t}, bR_{t+1}, \ldots, bR_n])\\ &\leq 2n\varepsilon_2(2^\ell + 1) + \varepsilon_1 + 2n\varepsilon_2(2^\ell + 1)\\ &= \varepsilon_1 + 4n\varepsilon_2(2^\ell + 1)\,, \end{aligned}$$

which concludes the proof of statistical privacy.

Statistical Non-Malleability: Let T be an authorized set of size $t \geq 3$. Without loss of generality, we may assume that $T = \{1, 2, \ldots, t\}$. Let $f_1, \ldots, f_t$ be the corresponding tampering functions. Let $s_1, \ldots, s_n \in \{0,1\}^{k+p}$ be arbitrary strings, and let $\mathbf{s} = (s_1, \ldots, s_n)$.

Definition 28. *We define the following partial tampering experiment* $\mathsf{IntTamp}^{T,f}_{\mathbf{s}}$.

1. *For each* $i \in [n]$, $(L_i, R_i) \leftarrow \mathbf{nmExt}^{-1}(s_i)$.
2. *For each* $i \in [n]$, *let* $S_i = (R_1, \ldots, R_{i-1}, L_i, R_{i+1}, \ldots, R_n)$.
3. *For each* $j \in [t]$, *let* f_j *be a function that maps* S_j *to* $\widetilde{R}^{(j)}_1, \ldots, \widetilde{R}^{(j)}_{j-1}, \widetilde{L}_j, \widetilde{R}^{(j)}_{j+1}, \ldots, \widetilde{R}^{(j)}_n$.
4. *Check whether* $\widetilde{R}^{(j_1)}_i = \widetilde{R}^{(j_2)}_i$ *for all distinct* i, j_1, j_2 *where* $i \in [n]$, *and* $j_1, j_2 \in T$. *If any of them is not true, then* $\mathsf{IntTamp}^{T,f}_{\mathbf{s}} = \perp$.
5. *For each* $i \geq 2$, *let* $\widetilde{R}_i = \widetilde{R}^{(1)}_i$, *and let* $\widetilde{R}_1 = \widetilde{R}^{(2)}_1$.
6. *For each* $i \in [t]$, *if* $L_i = \widetilde{L}_i$ *and* $R_i = \widetilde{R}_i$, *then* $\mathsf{output}_i = \mathsf{same}^*$, *else* $\mathsf{output}_i = \mathbf{nmExt}(\widetilde{L}_i, \widetilde{R}_i)$.
7. $\mathsf{IntTamp}^{T,f}_{\mathbf{s}} = (\mathsf{output}_1, \mathsf{output}_2, \ldots, \mathsf{output}_t)$.

We now show the following auxiliary lemma.

Lemma 29. *Let* $\mathbf{nmExt} : \{0,1\}^N \times \{0,1\}^N \to \{0,1\}^\ell$ *be a* (k, ε, τ) *strong non-malleable two-source extractor. Also, let* $h_1 : \{0,1\}^N \to \mathcal{Z}$, $h_2 : \{0,1\}^N \to \mathcal{Z}$, *and* $h_3 : \{0,1\}^N \to \{0,1\}$ *be functions for some set* $\mathcal{Z}$. *For functions* $F, G : \{0,1\}^N \to \{0,1\}^N$, *let* $\mathcal{A}_{F,G}$ *be an algorithm that takes as input* $x, y \in \{0,1\}^N$, *and does the following: If* $h_1(x) \neq h_2(y)$, *or if* $h_3(y) = 1$, *then output* $\perp$, *else if* $F(x) = x$, *and* $G_j(y) = y$, *output* same^*, *else output* $\mathbf{nmExt}(F(x), G(y))$. *For* X, Y *uniform and independent in* $\{0,1\}^N$, *we have that*

$$\Delta := \Delta(\mathbf{nmExt}(X, Y)\,;\, U_\ell \mid Y,\, \mathcal{A}_{F,G}(X, Y)) \leq \varepsilon + 2^{-\frac{N-k}{2}+1}\,.$$

Proof. Let $\mathcal{L}$ be defined as follows:

$$\mathcal{L} := \{x \in \{0,1\}^N \,:\, |h_1^{-1}(h_1(x))| \geq 2^k\}\,.$$

Let $\overline{\mathcal{L}} = \{0,1\}^N \setminus \mathcal{L}$. Let $X^\star$ be uniform in $\mathcal{L}$, and let $\overline{X}^\star$ be uniform in $\overline{\mathcal{L}}$. First we bound the required statistical distance assuming X is restricted to being uniform in $\mathcal{L}$. Notice that $\mathbf{H}_\infty(X^\star | h_1(X^\star)) \geq k$. Thus, by Lemma 12 we have that

$$\Delta(\mathbf{nmExt}(X^\star, Y) \; ; \; U_\ell \mid h_1(X^\star), Y, \; \mathcal{D}_{F,G}(X^\star, Y)) \leq \varepsilon \; .$$

Notice that $\mathcal{A}_{F,G}(X^\star, Y)$ is a deterministic function of $\mathcal{D}_{F,G}(X^\star, Y), h_1(X^\star), Y$. Thus, we have that

$$\Delta_1 := \Delta(\mathbf{nmExt}(X^\star, Y) \; ; \; U_\ell \mid Y, \; \mathcal{A}_{F,G}(X^\star, Y)) \leq \varepsilon \; .$$

We now proceed by cases.

CASE 1: $|\overline{\mathcal{L}}| \geq 2^{\frac{N+k}{2}}$. In this case, $\mathbf{H}_\infty(h_1(\overline{X}^\star)) = \frac{N-k}{2}$, which implies that

$$\Pr[\mathcal{A}_{F,G}(\overline{X}^\star, Y) \neq \perp] \leq \Pr[h_1(\overline{X}^\star) = h_2(Y)] \leq 2^{-\frac{N-k}{2}} \; .$$

Thus,

$$\Delta\Big(\mathcal{A}_{F,G}(\overline{X}^\star, Y) \; ; \; \perp \mid \mathbf{nmExt}(\overline{X}^\star, Y), \; Y\Big) \leq 2^{-\frac{N-k}{2}} \; ,$$

and

$$\Delta\Big(\mathcal{A}_{F,G}(\overline{X}^\star, Y) \; ; \; \perp \mid U_\ell, \; Y\Big) \leq 2^{-\frac{N-k}{2}} \; ,$$

Since $\mathbf{H}_\infty(\overline{X}^\star) \geq \frac{N+k}{2} \geq k$, by Lemma 12 we have that

$$\Delta\Big(\mathbf{nmExt}(\overline{X}^\star, Y) \; ; \; U_\ell \mid Y, \; \perp\Big) \leq \varepsilon \; .$$

Thus, by the triangle inequality, it follows that

$$\Delta_2 := \Delta\Big(\mathbf{nmExt}(\overline{X}^\star, Y) \; ; \; U_\ell \mid Y, \; \mathcal{A}_{F,G}(\overline{X}^\star, Y)\Big) \leq \varepsilon + 2^{-\frac{N-k}{2}+1} \; .$$

Combining, we conclude that

$$\Delta \leq \Delta_1 \cdot \Pr[X \in \mathcal{L}] + \Delta_2 \cdot \Pr[X \in \overline{\mathcal{L}}] \leq \varepsilon + 2^{-\frac{N-k}{2}+1} \; .$$

CASE 2: $|\overline{\mathcal{L}}| < 2^{\frac{N+k}{2}}$. In this case, we have that

$$\Delta \leq \Delta_1 \cdot \Pr[X \in \mathcal{L}] + \Delta_2 \cdot \Pr[X \in \overline{\mathcal{L}}] \leq \varepsilon + \Pr[X \in \overline{\mathcal{L}}] \leq \varepsilon + 2^{-\frac{N-k}{2}} \leq \varepsilon + 2^{-\frac{N-k}{2}+1} \; .$$

□

We now show the key component of our non-malleability proof.

Lemma 30. *For any* $\mathbf{s}, \mathbf{s}' \in \{0,1\}^{n\ell}$ *we have that*

$$\mathsf{IntTamp}_{\mathbf{s}}^{T,f} \; \approx_{n2^{\ell+1}\gamma} \; \mathsf{IntTamp}_{\mathbf{s}'}^{T,f} \; ,$$

where $\gamma = \varepsilon + 2^{-\delta N/2+1}$.

Proof. We show that, for $\mathbf{s} = (s_1, s_2 \ldots, s_n)$, and $\mathbf{s}' = (s_1', s_2, \ldots, s_n)$, we have

$$\mathsf{IntTamp}_{\mathbf{s}}^{T,f} \; \approx_{2^{\ell+1}\gamma} \; \mathsf{IntTamp}_{\mathbf{s}'}^{T,f}.$$

The general result then follows by a hybrid argument using an analogous reasoning.

For $i = 2, \ldots, n$, let $(L_i, R_i) \leftarrow \mathbf{nmExt}^{-1}(s_i)$, and let L_1^*, R_1^* be chosen independently and uniformly at random from $\{0,1\}^N$. Fix $L_2, \ldots, L_n, R_2, \ldots, R_n$. Assume that we run Steps 3 to 7 of the $\mathsf{IntTamp}_{\mathbf{s}}^{T,f}$ experiment described above, with L_1, R_1 replaced by L_1^*, R_1^*. We replace Step 5 by the following:

- For each $i \neq 2$, let $\widetilde{R}_i = \widetilde{R}_i^{(2)}$, and let $\widetilde{R}_2 = \widetilde{R}_2^{(3)}$,

i.e., we ensure that $\widetilde{R}_2, \ldots, \widetilde{R}_n$ are not a function of L_1^*. Notice that due to the consistency check in Step 4, the output of the tampering experiment remains the same. Then, recalling the variables we have fixed, it follows that L_1' is a deterministic functions of L_1^*, and $\widetilde{R}_1, \ldots, \widetilde{R}_n, \widetilde{L}_2, \ldots, \widetilde{L}_n$ are deterministic functions of R_1^*. Define

$$\begin{aligned} h_1(L_1^*) &:= (\widetilde{R}_2^{(1)}, \ldots, \widetilde{R}_n^{(1)}), \\ h_2(R_1^*) &:= (\widetilde{R}_2^{(3)}, \widetilde{R}_3^{(2)}, \ldots, \widetilde{R}_n^{(2)}), \\ F(L_1^*) &:= \widetilde{L}_1, \\ G(R_1^*) &:= \widetilde{R}_1^{(2)}. \end{aligned}$$

Also, let $h_3(R_1^*) = 1$ if and only if any of the checks in Step 4 with $j_1, j_2 \neq 1$ (i.e., the checks that are not dependent on L_1^*) fail. We can now instantiate Lemma 29 with h_1, h_2, h_3, F, G and the strong two-source non-malleable extractor from Lemma 11 to obtain

$$\Delta(\mathbf{nmExt}(L_1^*, R_1^*); U_\ell \mid \mathcal{A}_{F,G}(L_1^*, R_1^*), L_2, \ldots, L_n, R_2, \ldots, R_n, R_1^*) \leq \gamma \; . \tag{4}$$

Let $(L_1', R_1') \leftarrow \mathbf{nmExt}^{-1}(s_1')$, and observe that $\Pr[U_\ell = s] = 2^{-\ell}$ for all s. We can condition (4) on $U_\ell = s_1$ (resp. $U_\ell = s_1'$) and invoke Lemma 3 to obtain

$$\mathcal{A}_{F,G}(L_1, R_1), L_2, \ldots, L_n, R_2, \ldots, R_n, R_1 \approx_{2^\ell \gamma} \mathcal{A}_{F,G}(L_1^*, R_1^*), L_2, \ldots, L_n, R_2, \ldots, R_n, R_1^*,$$

and

$$\mathcal{A}_{F,G}(L_1', R_1'), L_2, \ldots, L_n, R_2, \ldots, R_n, R_1' \approx_{2^\ell \gamma} \mathcal{A}_{F,G}(L_1^*, R_1^*), L_2, \ldots, L_n, R_2, \ldots, R_n, R_1^*,$$

respectively. Applying the triangle inequality yields

$$\mathcal{A}_{F,G}(L_1, R_1), L_2, \ldots, L_n, R_2, \ldots, R_n, R_1 \approx_{2^{\ell+1}\gamma} \mathcal{A}_{F,G}(L_1', R_1'), L_2, \ldots, L_t, R_2, \ldots, R_t, R_1' \; . \tag{5}$$

Observe that $\mathsf{IntTamp}_{\mathbf{s}}^{T,f}$ and $\mathsf{IntTamp}_{\mathbf{s}'}^{T,f}$ are deterministic functions of the left hand side and right hand side of (3.3), respectively. As a result, we conclude that

$$\mathsf{IntTamp}_{\mathbf{s}}^{T,f} \;\approx_{2^{\ell+1}\gamma}\; \mathsf{IntTamp}_{\mathbf{s}'}^{T,f},$$

as desired. □

We are now ready to prove statistical non-malleability of our proposed construction.

Theorem 31. *The secret sharing scheme* $(\mathbf{NMShare}, \mathbf{NMRec})$ *is* ε*-non-malleable with respect to* $\mathcal{F}_n^{split}$ *for* $\varepsilon = n(2^{\ell+1}\gamma + 2^{-p})$*, where* $\gamma = \varepsilon_2 + 2^{-\delta N/2+1}$*.*

Proof. Fix a tampering function f and an authorized set $T \in \mathcal{A}$ of size $t \geq 3$. Without loss of generality, suppose that $T = \{1, \ldots, t\}$. Recall that our goal is to design a distribution $SD^{f,T}$ over $\mathcal{M} \cup \{\mathsf{same}^*, \perp\}$ such that

$$\mathbf{STamper}_{\boldsymbol{m}}^{\boldsymbol{f,T}} \approx_\varepsilon \mathbf{SSim}_{\boldsymbol{m}}^{\boldsymbol{f,T}} \tag{6}$$

for every secret m, where $\mathbf{STamper}_{\boldsymbol{m}}^{\boldsymbol{f,T}}$ and $\mathbf{SSim}_{\boldsymbol{m}}^{\boldsymbol{f,T}}$ are as in Definition 18.

We define $SD^{f,T}$ as

$$SD^{f,T} = \left\{ \begin{array}{l} \mathbf{s}' = (s_1', \ldots, s_n') \leftarrow \{0,1\}^{n\ell} \\ \widetilde{\mathbf{s}}' = (\widetilde{s}_1', \ldots, \widetilde{s}_t') \leftarrow \mathsf{IntTamp}_{\mathbf{s}'}^{T,f} \\ \text{If } \widetilde{s}_i' = \mathsf{same}^* \text{ for all } i, \text{ output } \mathsf{same}^* \\ \text{Else, if } \widetilde{s}_i' \neq \mathsf{same}^* \text{ for all } i, \text{ check if first } p \text{ bits of } \widetilde{\mathbf{s}}_1', \ldots, \widetilde{\mathbf{s}}_t' \text{ match:} \\ \text{If not, output } \perp. \text{ Otherwise, output } A\mathbf{Rec}^T(\widetilde{\mathbf{s}}_T''), \text{ where } \widetilde{\mathbf{s}}_i'' \text{ denotes the last } k \text{ bits of } \widetilde{\mathbf{s}}_i'. \\ \text{Else, output } \perp \end{array} \right\}.$$

We now prove (6) via a hybrid argument. Consider the following hybrids:

Hybrid$_0$ We proceed as follows:

1. $\mathbf{s} = (s_1, \ldots, s_n) \leftarrow \mathbf{AShare}(m)$;
2. Sample $P \leftarrow \{0,1\}^p$, and set $s_i \leftarrow P||s_i$;
3. $\widetilde{\mathbf{s}} = (\widetilde{s}_1, \ldots, \widetilde{s}_t) \leftarrow \mathsf{IntTamp}_{\mathbf{s}}^{T,f}$;
4. If $\widetilde{s}_i = \mathsf{same}^*$, set $\widetilde{s}_i \leftarrow s_i$;
5. Let $\widetilde{P}_i$ denote the first p bits of $\widetilde{s}_i$. If $\widetilde{P}_i \neq \widetilde{P}_j$ for some $i, j \leq t$, output $\perp$. Else, let $\widehat{s}_i$ denote the last k bits of $\widetilde{s}_i$, and output $\mathbf{ARec}^T(\widehat{s}_1, \ldots, \widehat{s}_t)$.

Observe that the output of **Hybrid$_0$** is distributed exactly like $\mathsf{STamper}_{\boldsymbol{m}}^{\boldsymbol{f},\boldsymbol{T}}$.

Hybrid$_1$ We proceed similarly to **Hybrid$_0$**, but replace $\mathbf{s}$ by a random vector of shares $\widetilde{\mathbf{s}}$:

1. $\mathbf{s} = (s_1, \ldots, s_n) \leftarrow \mathbf{AShare}(m)$;
2. Sample $P \leftarrow \{0,1\}^p$, and set $s_i \leftarrow P||s_i$;
3. $\mathbf{s}' = (s'_1, \ldots, s'_n) \leftarrow \{0,1\}^{n\ell}$;
4. $\widetilde{\mathbf{s}}' = (\widetilde{s}'_1, \ldots, \widetilde{s}'_t) \leftarrow \mathsf{IntTamp}_{\mathbf{s}'}^{T,f}$;
5. If $\widetilde{s}'_i = \mathsf{same}^*$, set $\widetilde{s}'_i \leftarrow s_i$;
6. Let $\widetilde{P}'_i$ denote the first p bits of $\widetilde{s}'_i$. If $\widetilde{P}'_i \neq \widetilde{P}'_j$ for some $i, j \leq t$, output $\perp$. Else, let $\widehat{s}'_i$ denote the last k bits of $\widetilde{s}'_i$, and output $\mathbf{ARec}^T(\widehat{s}'_1, \ldots, \widehat{s}'_t)$.

Hybrid$_2$ We proceed similarly to **Hybrid$_1$**, but modify the reconstruction procedure:

1. $\mathbf{s}' = (s'_1, \ldots, s'_n) \leftarrow \{0,1\}^{n\ell}$;
2. $\widetilde{\mathbf{s}}' = (\widetilde{s}'_1, \ldots, \widetilde{s}'_t) \leftarrow \mathsf{IntTamp}_{\widetilde{\mathbf{s}}}^{T,f}$;
3. If $\widetilde{s}'_i = \mathsf{same}^*$ for all $i = 1, \ldots, t$, output m;
4. Else, if $\widetilde{s}'_i \neq \mathsf{same}^*$ for all $i = 1, \ldots, t$, proceed as follows: Let $\widetilde{P}'_i$ denote the first p bits of $\widetilde{s}'_i$. If $\widetilde{P}'_i \neq \widetilde{P}'_j$ for some $i, j \leq t$, output $\perp$. Else, let $\widehat{s}'_i$ denote the last k bits of $\widetilde{s}'_i$, and output $\mathbf{ARec}^T(\widehat{s}'_1, \ldots, \widehat{s}'_t)$;
5. Else, output $\perp$.

Observe that the output of **Hybrid$_2$** is distributed exactly like $\mathsf{SSim}_{\boldsymbol{m}}^{\boldsymbol{f},\boldsymbol{T}}$.

Lemma 30 implies that

$$\mathbf{Hybrid_0} \approx_{n2^{\ell+1}\gamma} \mathbf{Hybrid_1}.$$

Therefore, it suffices to compare **Hybrid$_1$** and **Hybrid$_2$**. Observe that **Hybrid$_1$** and **Hybrid$_2$** may only differ if **Hybrid$_2$** reaches Step 5 of the procedure. This happens exactly when there exist $i, j \leq t$ such that $\widetilde{s}'_i = \mathsf{same}^*$ and $\widetilde{s}'_j \neq \mathsf{same}^*$. In this case, **Hybrid$_2$** always output $\perp$. However, **Hybrid$_1$** may not output $\perp$ in such a case if all prefixes $\widetilde{P}'_1, \ldots, \widetilde{P}'_t$ match in Step 6 of its procedure. Say **Hybrid$_1$** is *bad* if this event holds. We have

$$\begin{aligned}
\Pr[\mathbf{Hybrid_1} \text{ is bad}] &\leq \Pr[\exists (i,j) : \widetilde{s}'_i = \mathsf{same}^*, \widetilde{s}'_j \neq \mathsf{same}^*, \widetilde{P}'_j = P] \\
&\leq \Pr[\exists j : \widetilde{P}'_j = P] \\
&\leq \sum_{j=1}^{t} \Pr[\widetilde{P}'_j = P] \\
&\leq n2^{-p}.
\end{aligned}$$

The third inequality follows via a union bound, while the fourth inequality holds because $\widetilde{P}'_j$ and P are independent for all j, and P is uniform over $\{0,1\}^p$. This implies that

$$\mathbf{Hybrid_1} \approx_{n2^{-p}} \mathbf{Hybrid_2},$$

and hence (6) holds, as desired. □

We now instantiate Theorem 31 to obtain a compiler that transforms regular secret-sharing schemes into non-malleable ones with concrete parameters.

Corollary 32. *Let* (**AShare**, **ARec**) *be an efficient* (n, ε)*-secret-sharing scheme realizing access structure* $\mathcal{A}$ *such that* $|T| \geq 3$ *holds for all* $T \in \mathcal{A}$*. Furthermore, suppose* **AShare** *maps* m*-bit secrets to* n *binary shares of length* r*. Then, there exists an efficient* (n, ε_1')*-secret-sharing scheme* (**NMShare**, **NMRec**) *realizing access structure* $\mathcal{A}$ *that is* ε_2'*-non-malleable w.r.t.* $\mathcal{F}_n^{split}$*, with*

$$\varepsilon_1' = \varepsilon + 4n2^{-\Omega(r+p)}$$

and

$$\varepsilon_2' = n(2^{-\Omega(r+p)} + 2^{-p}).$$

Furthermore, it holds that **NMShare** *maps* m*-bit secrets to* n *binary shares of length* $O(n(r+p)\log(r+p))$*. In particular:*

- *If we set* $p = r$*, we obtain*

$$\varepsilon_1' = \varepsilon + 4n2^{-\Omega(r)},$$
$$\varepsilon_2' = n2^{-\Omega(r)},$$

  and shares of length $O(nr\log r)$*;*

- *If we set* $p = r + n$*, we obtain*

$$\varepsilon_1' = \varepsilon + 4n2^{-\Omega(r+n)},$$
$$\varepsilon_2' = n2^{-\Omega(r+n)},$$

  and shares of length $O(n(r+n)\log(r+n))$*;*

Proof. Let $\ell = r + p$, and recall that the $((1-\delta)N, \varepsilon_2)$ strong non-malleable extractor **nmExt** : $\{0,1\}^N \times \{0,1\}^N \to \{0,1\}^\ell$ from Lemma 11 handles $\ell = \Omega(N)$ and $\varepsilon_2 = 2^{-\Omega(N/\log N)}$.

We set $N = C_0 \cdot \ell \log \ell$, for some sufficiently large constant $C_0 > 0$. Then, we have

$$N/\log N \geq \frac{C_0 \cdot \ell}{2},$$

for ℓ large enough. As a result, we have $\varepsilon_2 \leq 2^{-C_1 \ell}$ for some constant C_1. We can choose C_0 large enough so that $C_1 \gg 1$. As a result, we conclude that

$$\varepsilon_1' = \varepsilon + 4n\varepsilon_2(2^\ell + 1) = \varepsilon + 4n2^{-\Omega(\ell)}.$$

Moreover, we also obtain

$$\varepsilon_2' = n(2^{\ell+1}(\varepsilon_2 + 2^{-\delta N/2+1}) + 2^{-p}) = n(2^{-\Omega(\ell)} + 2^{-p}).$$

Recall that **NMShare** shares the secret into n shares of length $n \cdot N = O(n\ell \log \ell)$, as desired. The statements in the lemma now follow by instantiating p. □

3.3 Non-Malleability with Concurrent Reconstruction

In this section, we show that the secret-sharing scheme described in Section 3.2 also satisfies the stronger notion of non-malleability with concurrent reconstruction as in Definition 19. Recall that in the concurrent reconstruction setting, the adversary is allowed to choose qualified sets $T_1, \ldots, T_\tau$ along with associated tampering functions $f^{(1)}, \ldots, f^{(\tau)}$, and can observe the outcomes of the experiments $\textsf{STamper}_m^{f^{(i)}, T_i}$ for $i \in [\tau]$. We have the following result.

Theorem 33. *Fix a number of parties* n *and an integer* p*. Furthermore, assume we have access to the following primitives:*

1. *For* $\varepsilon_1 \geq 0$*, let* (**AShare**, **ARec**) *be an* (n, ε_1)*-secret sharing scheme realizing an access structure* $\mathcal{A}$ *such that* $|T| \geq 3$ *holds whenever* $T \in \mathcal{A}$*. Suppose the corresponding shares lie in* $\{0,1\}^r$ *and the secrets in some set* $\mathcal{M}$*;*

2. *Let* $\mathbf{nmExt} : \{0,1\}^N \times \{0,1\}^N \to \{0,1\}^\ell$ *be the* $(N - N^\delta, \varepsilon_2, \tau)$ *strong two-source non-malleable extractor from Lemma 12, where* $\ell = r + p$*. Hence,* $\tau = N^\delta$*,* $\ell \leq N^{\Omega(1)}$*, and* $\varepsilon_2 = 2^{-N^{\Omega(1)}}$*.*

Then, there exists an $(n, \varepsilon_1 + 4n\varepsilon_2(2^\ell + 1))$*-secret sharing scheme realizing access structure* $\mathcal{A}$ *that is* (ε, τ)*-concurrent-reconstruction-non-malleable w.r.t.* $\mathcal{F}_n^{split}$*, where*

$$\varepsilon = n(2^{\ell+1}(\varepsilon_2 + 4\tau 2^\tau 2^{-N^\delta/4\tau}) + \tau \cdot 2^{-p}).$$

The resulting scheme $(\mathbf{NMShare}, \mathbf{NMRec})$ *shares an element of* $\mathcal{M}$ *into* n *shares, where each share contains* n *elements of* $\{0,1\}^N$*. Finally, if the two primitives are efficient and the access structure* $\mathcal{A}$ *supports efficient membership queries, then the constructed scheme* $(\mathbf{NMShare}, \mathbf{NMRec})$ *is also efficient.*

The candidate scheme for Theorem 33 has been defined in Section 3.2, and statistical privacy is already proved there. We now proceed to state and prove an auxiliary lemma, which generalizes Lemma 29 to the case of multiple tamperings.

Lemma 34. *Let* $\mathbf{nmExt} : \{0,1\}^N \times \{0,1\}^N \to \{0,1\}^\ell$ *be an* $(N - N^\delta, \varepsilon, \tau)$ *strong non-malleable two-source extractor. Also, let* $h_{1j} : \{0,1\}^N \to \mathcal{Z}$*,* $h_{2j} : \{0,1\}^N \to \mathcal{Z}$*, and* $h_{3j} : \{0,1\}^N \to \{0,1\}$ *for* $1 \leq j \leq \tau$ *be functions mapping to some set* $\mathcal{Z}$*. For functions* $F_1, \ldots, F_\tau, G_1, \ldots, G_\tau : \{0,1\}^N \to \{0,1\}^N$*, let* $\mathcal{A}_{F_j,G_j}$ *be an algorithm that takes as input* $x, y \in \{0,1\}^N$ *and does the following: If* $h_{1j}(x) \neq h_{2j}(y)$*, or if* $h_{3j}(y) = 1$*, then output* $\perp$*, else if* $F_j(x) = x$*, and* $G_j(y) = y$*, output* same^**, else output* $\mathbf{nmExt}(F_j(x), G_j(y))$*. For* X, Y *uniform and independent in* $\{0,1\}^N$*, we have that*

$$\Delta := \Delta(\mathbf{nmExt}(X,Y) \; ; \; U_\ell \mid Y, \; \mathcal{A}_{F_1,G_1}(X,Y), \ldots, \mathcal{A}_{F_\tau,G_\tau}(X,Y)) \leq \varepsilon + 4\tau 2^\tau 2^{-N^\delta/4\tau} \; .$$

Proof. We begin by observing that, for any partition $\mathcal{P}$ of $\{0,1\}^N$, we have

$$\Delta \leq \sum_{P \in \mathcal{P}} \Delta|_{X \in P} \cdot \Pr[X \in P],$$

where $\Delta|_{X \in P}$ denotes the statistical distance between the two distributions in the lemma statement conditioned on X being uniform in P.

We will now consider a relevant partition $\mathcal{P}$ of $\{0,1\}^N$, and analyze $\Delta|_{X \in P}$ for each set $P \in \mathcal{P}$ separately. First, we focus on the set

$$P_1 := \{x \in \{0,1\}^N : |h_{11}^{-1}(h_{11}(x)) \cap \cdots \cap h_{1\tau}^{-1}(h_{1\tau}(x))| \geq 2^{N-N^\delta}\}.$$

Let $X^\star$ be uniform in P_1. Then, by the definition of P_1, we have

$$\mathbf{H}_\infty(X^\star | h_{11}(X^\star), \ldots, h_{1\tau}(X^\star)) \geq N - N^\delta.$$

As a result, by Lemma 12 it follows that

$$\Delta(\mathbf{nmExt}(X^\star, Y) \; ; \; U_\ell \mid h_{11}(X^\star), \ldots, h_{1\tau}(X^\star), \; Y, \; \mathcal{D}_{F_1,G_1}(X^\star, Y), \ldots, \mathcal{D}_{F_\tau,G_\tau}(X^\star, Y)) \leq \varepsilon.$$

Since $\mathcal{A}_{F_j,G_j}(X^\star, Y)$ is a deterministic function of $h_{1j}(X^\star)$, Y, and $\mathcal{D}_{F_j,G_j}(X^\star, Y)$, we also have

$$\Delta(\mathbf{nmExt}(X^\star, Y) \; ; \; U_\ell \mid Y, \; \mathcal{A}_{F_1,G_1}(X^\star, Y), \ldots, \mathcal{A}_{F_\tau,G_\tau}(X^\star, Y)) \leq \varepsilon.$$

Hence, it holds that $\Delta|_{X \in P_1} \leq \varepsilon$.

For a set $\mathcal{I} \subseteq [\tau]$, define $P_\mathcal{I}$ as

$$P_\mathcal{I} := \left\{ x \in \{0,1\}^N \;\middle|\; \begin{array}{l} |h_{11}^{-1}(h_{11}(x)) \cap \cdots \cap h_{1\tau}^{-1}(h_{1\tau}(x))| < 2^{N-N^\delta}, \\ |h_{1i}^{-1}(h_{1i}(x))| < 2^{N-N^\delta/2\tau} \text{ for } i \in \mathcal{I}, \\ |h_{1j}^{-1}(h_{1j}(x))| \geq 2^{N-N^\delta/2\tau} \text{ for } j \notin \mathcal{I} \end{array} \right\}.$$

Observe that P_1 and the sets $(P_{\mathcal{I}})_{\mathcal{I}\subseteq[\tau]}$ are all pairwise disjoint and their union is $\{0,1\}^N$. Therefore, these sets form a partition of $\{0,1\}^N$, as desired. We now proceed to bound the terms $\Delta|_{X\in P_{\mathcal{I}}} \cdot \Pr[X \in P_{\mathcal{I}}]$ by partitioning $P_{\mathcal{I}}$ into two disjoint subsets and analyzing each one separately:

1. $P_{\mathcal{I}1} := \{x \in P_{\mathcal{I}} : |\bigcap_{j\notin\mathcal{I}} h_{1j}^{-1}(h_{1j}(x)) \cap P_{\mathcal{I}}| \geq 2^{N-N^\delta}\}$

   We distinguish two cases:

   (a) $|P_{\mathcal{I}1}| \leq 2^{N-N^\delta/4\tau}$:

   Then, we have

   $$\Delta|_{X\in P_{\mathcal{I}1}} \cdot \Pr[X \in P_{\mathcal{I}1}] \leq \Pr[X \in P_{\mathcal{I}1}] \leq 2^{-N^\delta/4\tau}.$$

   (b) $|P_{\mathcal{I}1}| \geq 2^{N-N^\delta/4\tau}$:

   Let $X_1^\star$ be uniform over $P_{\mathcal{I}1}$. Then, we have

   $$\mathbf{H}_\infty(h_{1i}(X_1^\star)) \geq (N - N^\delta/4\tau) - (N - N^\delta/2\tau) = N^\delta/4\tau$$

   for all $i \in \mathcal{I}$. Therefore,

   $$\Pr[h_{1i}(X_1^\star) = h_{2i}(Y)] \leq 2^{-N^\delta/4\tau},$$

   and so

   $$\Pr[\mathcal{A}_{F_i,G_i}(X_1^\star, Y) \neq \perp] \leq 2^{-N^\delta/4\tau}. \tag{7}$$

   Combining (7) with a union bound yields

   $$\Delta((\mathcal{A}_{F_i,G_i}(X_1^\star,Y))_{i\in\mathcal{I}}; \perp^{\mathcal{I}} | \mathbf{nmExt}(X_1^\star,Y), Y, (\mathcal{A}_{F_j,G_j}(X_1^\star,Y))_{j\notin\mathcal{I}}) \leq \tau 2^{-N^\delta/4\tau}, \tag{8}$$

   and

   $$\Delta((\mathcal{A}_{F_i,G_i}(X_1^\star,Y))_{i\in\mathcal{I}}; \perp^{\mathcal{I}} | U_\ell, Y, (\mathcal{A}_{F_j,G_j}(X_1^\star,Y))_{j\notin\mathcal{I}}) \leq \tau 2^{-N^\delta/4\tau}, \tag{9}$$

   where U_ℓ is uniform over $\{0,1\}^\ell$ and independent of the rest. Moreover, by definition of $P_{\mathcal{I}1}$, we have

   $$\mathbf{H}_\infty(X_1^\star | (h_{1j}(X_1^\star))_{j\notin\mathcal{I}}) \geq N - N^\delta,$$

   and so, with an analogous reasoning to that used for P_1, by Lemma 12 it follows that

   $$\Delta\big(\mathbf{nmExt}(X_1^\star, Y) \; ; \; U_\ell \mid Y, \; \perp^{\mathcal{I}}, (\mathcal{A}_{F_j,G_j}(X_1^\star,Y))_{j\notin\mathcal{I}}\big) \leq \varepsilon. \tag{10}$$

   Combining (10) with (8) and (9) via a repeated application of the triangle inequality yields

   $$\Delta(\mathbf{nmExt}(X_1^\star,Y) \; ; \; U_\ell \mid Y, \; \mathcal{A}_{F_1,G_1}(X_1^\star,Y), \ldots, \mathcal{A}_{F_\tau,G_\tau}(X_1^\star,Y)) \leq \varepsilon + 2\tau 2^{-N^\delta/4\tau}.$$

2. $P_{\mathcal{I}2} := \{x \in P_{\mathcal{I}} : |\bigcap_{j\notin\mathcal{I}} h_{1j}^{-1}(h_{1j}(x)) \cap P_{\mathcal{I}}| < 2^{N-N^\delta}\}$

   We claim that $|P_{\mathcal{I}2}| \leq 2^{N-N^\delta/2}$. In fact, note that $h_{1j}(x)$ takes on at most $2^{N^\delta/2\tau}$ distinct values for $j \notin \mathcal{I}$ and $x \in P_{\mathcal{I}}$. As a result, there are at most $(2^{N^\delta/2\tau})^\tau = 2^{N^\delta/2}$ sets of the form

   $$\bigcap_{j\notin\mathcal{I}} h_{1j}^{-1}(h_{1j}(x)) \cap P_{\mathcal{I}}$$

   with $x \in P_{\mathcal{I}}$. By definition of $P_{\mathcal{I}2}$, each such set contributes at most 2^{N-N^δ} elements to $P_{\mathcal{I}2}$. Therefore, we have

   $$|P_{\mathcal{I}2}| \leq 2^{N^\delta/2} \cdot 2^{N-N^\delta} = 2^{N-N^\delta/2},$$

   as desired. We thus conclude that $\Pr[X \in P_{\mathcal{I}2}] \leq 2^{-N^\delta/2}$.

Observing that there are 2^τ choices for $\mathcal{I}$, we can combine the bounds above to conclude that

$$\begin{aligned}\Delta &\le \Delta|_{X\in P_1}\cdot \Pr[X\in P_1] + \sum_{\mathcal{I}}(\Delta|_{P_{\mathcal{I}1}}\cdot \Pr[X\in P_{\mathcal{I}1}] + \Delta|_{P_{\mathcal{I}2}}\cdot \Pr[X\in P_{\mathcal{I}2}])\\ &\le \varepsilon + 4\tau 2^\tau 2^{-N^\delta/4\tau}.\end{aligned}$$

□

Given a tuple of qualified sets $\mathbf{T}=(T_1,\dots,T_\tau)$ and a tuple of associated tampering functions $\mathbf{f}=(f^{(1)},\dots,f^{(\tau)})$, we define the intermediate tampering experiment for $\mathbf{T}$ as follows:

$$\mathsf{IntTamp}_{\mathbf{s}}^{\mathbf{T},\mathbf{f}} := \mathsf{IntTamp}_{\mathbf{s}}^{T_1,f^{(1)}},\dots,\mathsf{IntTamp}_{\mathbf{s}}^{T_\tau,f^{(\tau)}}.$$

We may also denote the tampering function f associated to a reconstruction set $T\in\mathbf{T}$ by $f^{(T)}$. The following lemma is the main component of our proof of non-malleability with concurrent reconstruction.

Lemma 35. *For any* $\mathbf{s},\mathbf{s}'\in\{0,1\}^{n\ell}$ *we have that*

$$\mathit{IntTamp}_{\mathbf{s}}^{\mathbf{T},\mathbf{f}} \approx_{n2^{\ell+1}\gamma} \mathit{IntTamp}_{\mathbf{s}'}^{\mathbf{T},\mathbf{f}},$$

where $\gamma=\varepsilon_2+4\tau 2^\tau 2^{-N^\delta/4\tau}$.

Proof. We show that for $\mathbf{s}=(s_1,s_2\dots,s_n)$, and $\mathbf{s}'=(s_1',s_2,\dots,s_n)$, we have

$$\mathsf{IntTamp}_{\mathbf{s}}^{\mathbf{T},\mathbf{f}} \approx_{2^{\ell+1}\gamma} \mathsf{IntTamp}_{\mathbf{\tilde{s}}}^{\mathbf{T},\mathbf{f}}.$$

The general result then follows by a hybrid argument using an analogous reasoning.

For a given reconstruction set $T\in\mathbf{T}$ and associated tampering function $f^{(T)}$, we denote the tampered version of the j-th share $(R_1,\dots,R_{j-1},L_j,R_{j+1},\dots,R_n)$ under $f^{(T)}$ by

$$(\widetilde{R}^{(j)}_{1,T},\dots,\widetilde{R}^{(j)}_{j-1,T},\widetilde{L}_{j,T},\widetilde{R}^{(j)}_{j+1,T},\dots,\widetilde{R}^{(j)}_{n,T}).$$

For $i=2,\dots,n$, let $(L_i,R_i)\leftarrow \mathbf{nmExt}^{-1}(s_i)$, and let L_1^*,R_1^* be chosen independently and uniformly at random from $\{0,1\}^N$. Fix $L_2,\dots,L_n,R_2,\dots,R_n$. Assume that we run Steps 3 to 7 of the $\mathsf{IntTamp}_{\mathbf{s}}^{T,f}$ experiment in Definition 28 with L_1,R_1 replaced by L_1^*,R_1^* for each set $T\in\mathbf{T}$. Also, we replace Step 5 by the following:

- For each i, let $\widetilde{R}_{i,T}=\widetilde{R}^{(j_i)}_{i,T}$,

for the smallest index $j_i\in T\setminus\{1,i\}$, i.e., we ensure that $\widetilde{R}_{2,T},\dots,\widetilde{R}_{n,T}$ do not depend on L_1^*. There always exists a valid choice of j_i since we assume $|T|\ge 3$ for all $T\in\mathbf{T}$. Notice that due to the consistency check in Step 4, the output of the tampering experiment remains the same. Then, recalling the variables we have fixed, it follows that $L'_{1,T}$ is a deterministic function of L_1^*, and $\widetilde{R}_{1,T},\dots,\widetilde{R}_{n,T},\widetilde{L}_{2,T},\dots,\widetilde{L}_{n,T}$ for every $T\in\mathbf{T}$ are deterministic functions of R_1^*. Without loss of generality, suppose $T_1,\dots,T_m$ are exactly those reconstructions sets in $\mathbf{T}$ that contain 1. Notice that $\mathsf{IntTamp}_{\mathbf{s}}^{\mathbf{T},\mathbf{f}}$ is a deterministic function of

$$R_1^*,\mathsf{IntTamp}_{\mathbf{s}}^{T_1,f^{(1)}},\dots,\mathsf{IntTamp}_{\mathbf{s}}^{T_m,f^{(m)}},$$

since for all other reconstruction sets T_i with $i>m$ we have that $\mathsf{IntTamp}_{\mathbf{s}}^{T_i,f^{(i)}}$ is a deterministic function of R_1^*. For $1\le j\le m$, define

$$\begin{aligned}h_{1j}(L_1^*) &:= (\widetilde{R}^{(1)}_{2,T_j},\dots,\widetilde{R}^{(1)}_{n,T_j}),\\ h_{2j}(R_1^*) &:= (\widetilde{R}_{2,T_j},\widetilde{R}_{3,T_j},\dots,\widetilde{R}_{n,T_j}),\end{aligned}$$

$$F_j(L_1^*) := \widetilde{L}_{1,T_j},$$
$$G_j(R_1^*) := \widetilde{R}_{1,T_j}.$$

Also, let $h_{3j}(R_1^*) = 1$ if and only if any of the checks in Step 4 with $j_1, j_2 \neq 1$ (i.e., the checks that are not dependent on L_1^*) fail for the reconstruction set T_j. We can now instantiate Lemma 34 with these choices to obtain

$$\Delta(\mathbf{nmExt}(L_1^*, R_1^*); U_\ell \mid \mathcal{A}_{F_1,G_1}(L_1^*, R_1^*), \ldots, \mathcal{A}_{F_m,G_m}(L_1^*, R_1^*), L_2, \ldots, L_n, R_2, \ldots, R_n, R_1^*) \leq \gamma\ . \tag{11}$$

Let $(L_1', R_1') \leftarrow \mathbf{nmExt}^{-1}(s_1')$, and observe that $\Pr[U_\ell = s] = 2^{-\ell}$ for all s. We can condition (11) on $U_\ell = s_1$ (resp. $U_\ell = s_1'$) and invoke Lemma 3 to obtain

$$\begin{gathered} R_1^*, \mathcal{A}_{F_1,G_1}(L_1^*, R_1^*), \ldots, \mathcal{A}_{F_m,G_m}(L_1^*, R_1^*), L_2, \ldots, L_n, R_2, \ldots, R_n \\ \approx_{2^\ell \gamma} \\ R_1, \mathcal{A}_{F_1,G_1}(L_1, R_1), \ldots, \mathcal{A}_{F_m,G_m}(L_1, R_1), , L_2, \ldots, L_n, R_2, \ldots, R_n, \end{gathered}$$

and

$$\begin{gathered} R_1^*, \mathcal{A}_{F_1,G_1}(L_1^*, R_1^*), \ldots, \mathcal{A}_{F_m,G_m}(L_1^*, R_1^*), L_2, \ldots, L_n, R_2, \ldots, R_n \\ \approx_{2^\ell \gamma} \\ R_1', \mathcal{A}_{F_1,G_1}(L_1', R_1'), \ldots, \mathcal{A}_{F_m,G_m}(L_1', R_1'), L_2, \ldots, L_n, R_2, \ldots, R_n, \end{gathered}$$

respectively. Applying the triangle inequality yields

$$\begin{gathered} R_1, \mathcal{A}_{F_1,G_1}(L_1, R_1), \ldots, \mathcal{A}_{F_m,G_m}(L_1, R_1), , L_2, \ldots, L_n, R_2, \ldots, R_n \\ \approx_{2^{\ell+1} \gamma} \\ R_1', \mathcal{A}_{F_1,G_1}(L_1', R_1'), \ldots, \mathcal{A}_{F_m,G_m}(L_1', R_1'), L_2, \ldots, L_n, R_2, \ldots, R_n. \end{gathered}$$

Observe that the left hand side completely determines $\mathsf{IntTamp}_{\mathbf{s}}^{\mathbf{T},\mathbf{f}}$, while the right hand side completely determines $\mathsf{IntTamp}_{\mathbf{s}'}^{\mathbf{T},\mathbf{f}}$. As a result, we conclude that

$$\mathsf{IntTamp}_{\mathbf{s}}^{\mathbf{T},\mathbf{f}} \approx_{2^{\ell+1}\gamma} \mathsf{IntTamp}_{\mathbf{s}'}^{\mathbf{T},\mathbf{f}},$$

as desired. □

We are now ready to prove statistical non-malleability of our proposed construction.

Theorem 36. *The secret sharing scheme* $(\mathbf{NMShare}, \mathbf{NMRec})$ *is* (ε, τ)*-concurrent reconstruction non-malleable with respect to* $\mathcal{F}_n^{split}$ *for* $\varepsilon = n(2^{\ell+1}\gamma + \tau 2^{-p})$*, where* $\gamma = \varepsilon_2 + 4\tau 2^\tau 2^{-N^\delta/4\tau}$.

Proof. Fix authorized sets $\mathbf{T} = (T_1, \ldots, T_\tau)$, with $|T_i| = t_i \geq 3$ for all $i \in [\tau]$, and associated tampering functions $\mathbf{f} = (f^{(1)}, \ldots, f^{(\tau)})$. Recall that our goal is to design a distribution $SD^{\mathbf{f},\mathbf{T}}$ over $(\mathcal{M} \cup \{\mathsf{same}^*, \perp\})^\tau$ such that

$$\mathsf{SCRTamper}_m^{\mathbf{f},\mathbf{T}} \approx_\varepsilon \mathsf{SCRSim}_m^{\mathbf{f},\mathbf{T}} \tag{12}$$

for every secret m, where $\mathsf{SCRTamper}_m^{\mathbf{f},\mathbf{T}}$ and $\mathsf{SCRSim}_m^{\mathbf{f},\mathbf{T}}$ are as in Definition 19.

We define $SD^{\mathbf{f},\mathbf{T}}$ as follows:

$$SD^{\mathbf{f},\mathbf{T}} = \left\{ \begin{array}{l} \mathbf{s}' = (s_1', \ldots, s_n') \leftarrow \{0,1\}^{n\ell} \\ \text{For each } i \in [\tau], \text{ set } \widetilde{\mathbf{s}}^{(i)} = (\widetilde{s}_1'^{(i)}, \ldots, \widetilde{s}_{t_i}'^{(i)}) \leftarrow \mathsf{IntTamp}_{\mathbf{s}'}^{T_i, f^{(i)}} \\ \text{For each } i \in [\tau], \text{ do the following:} \\ \text{If } \widetilde{s}_j'^{(i)} = \mathsf{same}^* \text{ for all } j \in [n], \text{ output } \mathsf{same}^* \\ \text{Else, if } \widetilde{s}_j'^{(i)} \neq \mathsf{same}^* \text{ for all } j, \text{ check if first } p \text{ bits of } \widetilde{\mathbf{s}}_1', \ldots, \widetilde{\mathbf{s}}_{t_i}' \text{ match:} \\ \text{If not, output } \perp. \text{ Otherwise, output } \mathbf{ARec}^T(\widehat{\mathbf{s}}_{T_i}'^{(i)}), \text{ where } \widehat{\mathbf{s}}_j'^{(i)} \text{ denotes the last } k \text{ bits of } \widetilde{\mathbf{s}}_j'^{(i)}. \\ \text{Else, output } \perp \end{array} \right\}.$$

We now prove (6) via a hybrid argument. Consider the following hybrids:

$\mathbf{Hybrid_0}$ We proceed as follows:

1. $\mathbf{s} = (s_1, \dots, s_n) \leftarrow \mathbf{AShare}(m)$;
2. Sample $P \leftarrow \{0,1\}^p$, and set $s_i \leftarrow P||s_i$;
3. For each $i \in [\tau]$, do the following:
   (a) $\widetilde{\mathbf{s}}^{(i)} = (\widetilde{s}_1^{(i)}, \dots, \widetilde{s}_{t_i}^{(i)}) \leftarrow \mathsf{IntTamp}_{\mathbf{s}}^{T_i, f^{(i)}}$;
   (b) If $\widetilde{s}_j^{(i)} = \mathsf{same}^*$, set $\widetilde{s}_j^{(i)} \leftarrow s_j$;
   (c) Let $\widetilde{P}_j^{(i)}$ denote the first p bits of $\widetilde{s}_j^{(i)}$. If $\widetilde{P}_{j_1}^{(i)} \neq \widetilde{P}_{j_2}^{(i)}$ for some $j_1, j_2 \leq t_i$, output $\perp$. Else, let $\widehat{s}_j^{(i)}$ denote the last k bits of $\widetilde{s}_j^{(i)}$, and output $\mathbf{ARec}^{T_i}(\widehat{s}_1^{(i)}, \dots, \widehat{s}_{t_i}^{(i)})$.

Observe that the output of $\mathbf{Hybrid_0}$ is distributed exactly like $\mathsf{SCRTamper}_{\boldsymbol{m}}^{\mathbf{f},\mathbf{T}}$.

$\mathbf{Hybrid_1}$ We proceed similarly to $\mathbf{Hybrid_0}$, but replace $\mathbf{s}$ by a random vector of shares $\mathbf{s}'$:

1. $\mathbf{s} = (s_1, \dots, s_n) \leftarrow \mathbf{AShare}(m)$;
2. Sample $P \leftarrow \{0,1\}^p$, and set $s_i \leftarrow P||s_i$;
3. $\mathbf{s}' = (s'_1, \dots, s'_n) \leftarrow \{0,1\}^{n\ell}$;
4. For each $i \in [\tau]$, do the following:
   (a) $\widetilde{\mathbf{s}}'^{(i)} = (\widetilde{s}_1'^{(i)}, \dots, \widetilde{s}_{t_i}'^{(i)}) \leftarrow \mathsf{IntTamp}_{\mathbf{s}'}^{T_i, f^{(i)}}$;
   (b) If $\widetilde{s}_j'^{(i)} = \mathsf{same}^*$, set $\widetilde{s}_j'^{(i)} \leftarrow s_j$;
   (c) Let $\widetilde{P}_j'^{(i)}$ denote the first p bits of $\widetilde{s}_j'^{(i)}$. If $\widetilde{P}_{j_1}'^{(i)} \neq \widetilde{P}_{j_2}'^{(i)}$ for some $j_1, j_2 \leq t_i$, output $\perp$. Else, let $\widehat{s}_j'^{(i)}$ denote the last k bits of $\widetilde{s}_j'^{(i)}$, and output $\mathbf{ARec}^{T_i}(\widehat{s}_1'^{(i)}, \dots, \widehat{s}_{t_i}'^{(i)})$.

$\mathbf{Hybrid_2}$ We proceed similarly to $\mathbf{Hybrid_1}$, but modify the reconstruction procedure:

1. $\mathbf{s}' = (s'_1, \dots, s'_n) \leftarrow \{0,1\}^{n\ell}$;
2. For each $i \in [\tau]$, do the following:
   (a) $\widetilde{\mathbf{s}}'^{(i)} = (\widetilde{s}_1'^{(i)}, \dots, \widetilde{s}_{t_i}'^{(i)}) \leftarrow \mathsf{IntTamp}_{\mathbf{s}'}^{T_i, f^{(i)}}$;
   (b) If $\widetilde{s}_j'^{(i)} = \mathsf{same}^*$ for all $j \in [t_i]$, output m;
   (c) Else, if $\widetilde{s}_j'^{(i)} \neq \mathsf{same}^*$ for all $j \in [t_i]$, proceed as follows: Let $\widetilde{P}_j'^{(i)}$ denote the first p bits of $\widetilde{s}_j'^{(i)}$. If $\widetilde{P}_{j_1}'^{(i)} \neq \widetilde{P}_{j_2}'^{(i)}$ for some $j_1, j_2 \leq t_i$, output $\perp$. Else, let $\widehat{s}_j'^{(i)}$ denote the last k bits of $\widetilde{s}_j'^{(i)}$, and output $\mathbf{ARec}^{T_i}(\widehat{s}_1'^{(i)}, \dots, \widehat{s}_{t_i}'^{(i)})$.
   (d) Else, output $\perp$.

Observe that the output of $\mathbf{Hybrid_2}$ is distributed exactly like $\mathsf{SCRSim}_{\boldsymbol{m}}^{\mathbf{f},\mathbf{T}}$.

Since the only difference between $\mathbf{Hybrid_0}$ and $\mathbf{Hybrid_1}$ is in the $\mathsf{IntTamp}$ experiment to be used for all $i \in [\tau]$ (Steps 3.a and 4.a, respectively), Lemma 35 implies that

$$\mathbf{Hybrid_0} \approx_{n2^{\ell+1}\gamma} \mathbf{Hybrid_1}.$$

Therefore, it suffices to compare $\mathbf{Hybrid_1}$ and $\mathbf{Hybrid_2}$. Observe that $\mathbf{Hybrid_1}$ and $\mathbf{Hybrid_2}$ may only differ if $\mathbf{Hybrid_2}$ reaches Step 2.d of the procedure for some $i \in [\tau]$. This happens exactly when there exist $j_1, j_2 \leq t_i$ such that $\widetilde{s}_{j_1}'^{(i)} = \mathsf{same}^*$ and $\widetilde{s}_{j_2}'^{(i)} \neq \mathsf{same}^*$. In this case, $\mathbf{Hybrid_2}$ always output $\perp$. However, $\mathbf{Hybrid_1}$ may not output $\perp$ in such a case if all prefixes $\widetilde{P}_1'^{(i)}, \dots, \widetilde{P}_{t_i}'^{(i)}$ match in Step 6 of its procedure. The reasoning in the proof of Theorem 31 shows that this happens for a fixed $i \in [\tau]$ with probability at most $n2^{-p}$. By a union bound, it follows that the probability that this happens for some $i \in [\tau]$ is at most $\tau n 2^{-p}$. This implies that

$$\mathbf{Hybrid_1} \approx_{\tau n 2^{-p}} \mathbf{Hybrid_2},$$

and hence (12) holds, as desired. □

We now instantiate our compiler with concrete parameters.

Corollary 37. *Let* (**AShare**, **ARec**) *be an efficient* (n, ε)*-secret-sharing scheme realizing access structure* $\mathcal{A}$ *such that* $|T| \geq 3$ *holds for all* $T \in \mathcal{A}$*. Furthermore, suppose* **AShare** *maps* m*-bit secrets to* n *binary shares of length* r*. Then, there exists an efficient* (n, ε'_1)*-secret-sharing scheme* (**NMShare**, **NMRec**) *realizing access structure* $\mathcal{A}$ *that is* $(\varepsilon'_2, \mathrm{poly}(r+n))$*-concurrent-reconstruction-non-malleable w.r.t.* $\mathcal{F}_n^{split}$*, with*

$$\varepsilon'_1 = \varepsilon + 4n2^{-\mathrm{poly}(r+n)}$$

and

$$\varepsilon'_2 = n2^{-\Omega(r+n)}.$$

Furthermore, it holds that **NMShare** *maps* m*-bit secrets to* n *binary shares of length* $\mathrm{poly}(r+n)$*.*

Proof. Let $\ell = r+p$, and recall that the $(N - N^\delta, \varepsilon_2, \tau)$ strong non-malleable extractor **nmExt** : $\{0,1\}^N \times \{0,1\}^N \to \{0,1\}^\ell$ from Lemma 11 handles $\ell = N^{\Omega(1)}$, $\tau = N^{\Omega(1)}$, and $\varepsilon_2 = 2^{-N^{\Omega(1)}}$.

We set $N = \ell^{C_0}$, for some sufficiently large constant $C_0 > 1$. As a result, we can have $\varepsilon_2 \leq 2^{-\ell^{C_1}}$ and $\tau = \ell^{C_2}$ for some constants C_1, C_2. We can choose C_0 large enough so that $C_1 \gg 1$. As a result, we conclude that

$$\varepsilon'_1 = \varepsilon + 4n\varepsilon_2(2^\ell + 1) = \varepsilon + 4n2^{-\mathrm{poly}(\ell)}.$$

Moreover, by setting $p = (r+n)^2$ and C_0 large enough so that $\delta C_0 \gg C_2$, we also obtain

$$\begin{aligned}\varepsilon'_2 &= n(2^{\ell+1}(\varepsilon_2 + 4\tau 2^\tau 2^{-\ell^{\delta C_0}}) + \tau 2^{-p}) \\ &\leq n(2^{-\mathrm{poly}(\ell)} + \ell^{C_2} 2^{-(r+n)^2}) \\ &\leq n2^{-\Omega(r+n)}.\end{aligned}$$

Recall that **NMShare** shares the secret into n shares of length $n \cdot N = \mathrm{poly}(r+n)$, as desired. □

4 Leakage-Resilient Secret-Sharing Scheme

In this section, we give a construction of a compiler that turns any secret-sharing scheme into a leakage-resilient one. More precisely, we have the following result.

Theorem 38. *Fix a number of parties* n *and* $\rho \in (0,1)$*. Furthermore, suppose we have access to the following primitives:*

1. *For any* $\varepsilon_1 \geq 0$*, let* (**AShare**, **ARec**) *be any* (n, ε_1)*-secret sharing scheme realizing an access structure* $\mathcal{A}$ *which does not contain singletons, which shares an element of the set* $\mathcal{M}$ *into* n *shares of length* ℓ*, and*
2. *Let* $\mathsf{Ext} : \{0,1\}^N \times \{0,1\}^d \to \{0,1\}^\ell$ *be the* (k, ε_2)*-extractor from Lemma 9 with parameters* $k = (1-\rho)N/2$*,* $\ell \leq \rho N/3$*, and* $d = C(\log N + \log(1/\varepsilon_2))$*.*

If $N \geq \frac{2nd}{1-\rho}$*, then there exists an* $(n, \varepsilon_1 + 2\varepsilon_2 \cdot n \cdot 2^{\ell n}, \rho)$*-leakage resilient secret sharing scheme realizing access structure* $\mathcal{A}$*.*

We describe our construction of the non-malleable secret sharing scheme (**LRShare**, **LRRec**).

LRShare: Our sharing function takes as input a secret $m \in \mathcal{M}$ and proceeds as follows:

1. Share m using **AShare** to obtain $s_1, \ldots, s_n \leftarrow$ **AShare**(m);
2. For each $i \in [n]$, sample $(L_i, R_i) \leftarrow \mathsf{Ext}^{-1}(s_i)$;
3. For each $i \in [n]$, construct $share_i = (R_1, \ldots, R_{i-1}, L_i, R_{i+1}, \ldots, R_n)$;
4. Output $(share_1, \ldots, share_n)$.

LRRec: Our reconstruction function takes as input shares $\{share_i : i \in T\}$ corresponding to an authorized set $T \in \mathcal{A}$ and proceeds as follows:

1. Sort T so that $T = \{i_1, \ldots, i_t\}$, where $t = |T|$, and $i_j < i_{j+1}$;
2. For each $j \in [t]$, obtain L_j from $share_j$, and R_j from $share_k$ for some $k \in T \setminus \{j\}$, and compute $s_j = \mathsf{Ext}(L_j, R_j)$.
3. Reconstruct $m \leftarrow \mathbf{ARec}(s_{i_1}, \ldots, s_{i_t})$, and output m.

Correctness and Efficiency: Follows in a straightforward manner from the construction.

Leakage-Resilient Statistical Privacy: Fix two secrets a and b, and let T be an unauthorized set of size t. Without loss of generality, we may assume that $T = \{1, 2, \ldots, t\}$. Set

$$aS_T \leftarrow \mathbf{LRShare}(a)_T, \{\mathsf{Leak}_i(\mathbf{LRShare}(a)_i) \ : \ 1 \le i \le n\},$$
$$bS_T \leftarrow \mathbf{LRShare}(b)_T, \{\mathsf{Leak}_i(\mathbf{LRShare}(b)_i) \ : \ 1 \le i \le n\}.$$

Furthermore, let $as_1, \ldots, as_n$ and $bs_1, \ldots, bs_n$ be the shares obtained from $\mathbf{AShare}(a)$ and $\mathbf{AShare}(b)$, respectively, in Step 1 of the $\mathbf{LRShare}$ procedure.

Our goal is to show that the distributions of these two sets of shares, aS_T and bS_T, are close in statistical distance, even given a ρ fraction leakage from the other shares.

We have $aS_T = (as_1, \ldots, as_t)$ and $bS_T = (bs_1, \ldots, bs_t)$, with

$$as_i = (aR_1, \ldots, aR_{i-1}, aL_i, aR_{i+1}, \ldots, aR_n, \{\mathsf{Leak}_i(\mathbf{LRShare}(a)_i)\}_{i \in [n]}),$$
$$bs_i = (bR_1, \ldots, bR_{i-1}, bL_i, bR_{i+1}, \ldots, bR_n, \{\mathsf{Leak}_i(\mathbf{LRShare}(b)_i)\}_{i \in [n]}).$$

As a result, we can write

$$aS_T = [(aL_i, aR_i)_{i \le t}, aR_{t+1}, \ldots, aR_n, \{\mathsf{Leak}_i(\mathbf{LRShare}(a)_i)\}_{i \in [n]}],$$
$$bS_T = [(bL_i, bR_i)_{i \le t}, bR_{t+1}, \ldots, bR_n, \{\mathsf{Leak}_i(\mathbf{LRShare}(b)_i)\}_{i \in [n]}].$$

Let $L^*_{t+1}, \ldots, L^*_n$ be uniform in $\{0,1\}^N$, and let $R^*_{t+1}, \ldots, R^*_n$ be uniform in $\{0,1\}^d$. Let aS^*_T be obtained by replacing $aL_{t+1}, \ldots, aL_n, aR_{t+1}, \ldots, aR_n$ by $L^*_{t+1}, \ldots, L^*_n, R^*_{t+1}, \ldots, R^*_n$. Define bS^*_T similarly. By the extractor property and a hybrid argument similar to the one used for the proof of statistical privacy in Section 3.2, we have that

$$\Delta(\mathsf{Ext}(L^*_{t+1}, R^*_{t+1}), \ldots, \mathsf{Ext}(L^*_n, R^*_n)\ ; U_\ell^{n-t} \mid aS^*_T) \le \varepsilon_2 \cdot n\ .$$

Thus, using Lemma 3, we have that

$$\Delta(aS_T\ ;\ aS^*_T) \le \varepsilon_2 \cdot n \cdot 2^{\ell n}\ .$$

Similarly,

$$\Delta(bS_T\ ;\ bS^*_T) \le \varepsilon_2 \cdot n \cdot 2^{\ell n}\ .$$

Also, by Lemma 2, we have that

$$\Delta(aS^*_T\ ;\ bS^*_T) \le \varepsilon_1\ .$$

By applying triangle inequality, we get the desired result.

Leakage rate: We now proceed to study the tradeoff between share-length and leakage rate that we can achieve via the compiler. From Theorem 38 we obtain the following result.

Corollary 39. *Let* $(\mathbf{AShare}, \mathbf{ARec})$ *be an efficient* (n, ε)*-secret-sharing scheme realizing access structure* $\mathcal{A}$ *with no singletons. Furthermore, suppose* $\mathbf{AShare}$ *maps* m*-bit secrets to* n *binary shares of length* ℓ*. Then, there exist efficient* (n, ε', ρ)*-leakage resilient secret-sharing schemes* $(\mathbf{LRShare}, \mathbf{LRRec})$ *realizing access structure* $\mathcal{A}$ *with* $\varepsilon' = \varepsilon + n2^{-\Omega(\ell)}$*, and, assuming the number of parties* n *is constant,*

- *Shares of length* $O(\ell)$ *and leakage rate* $\rho = 1 - c$ *for an arbitrarily small constant* $c > 0$*, or*
- *Shares of length* $\omega(\ell)$ *and leakage rate* $\rho = 1 - o(1)$*.*

Proof. We begin by setting $\varepsilon_2 = 2^{-C_0\ell}$ for a constant C_0. Observe that each share of the compiled scheme has length $(n-1)d + N$, and $d = C(\log N + C_0\ell)$. Recall we can achieve leakage rate ρ satisfying

$$1 - \rho = \frac{2nd}{N}.$$

For the first part of the corollary statement, set $N = C_1\ell$ for a large constant C_1. Then, the share length satisfies $(n-1)d + N = O(\ell)$. Furthermore,

$$1 - \rho = \frac{2nd}{N} \leq \frac{4nC \cdot C_0}{C_1}$$

for ℓ large enough. As a result, it holds that $1-\rho$ can be made an arbitrarily small constant by increasing C_1.

For the second part of the corollary statement, set $N = \omega(\ell)$. Then, the share length is $\omega(\ell)$ and $1 - \rho = o(1)$ since $d = o(N)$. □

5 Threshold Signatures

(n,t)-Threshold signatures, introduced by Desmedt [Des87], allows to distribute the secret key of a signature scheme among n players such that any subset of t players can sign messages. Threshold signatures exist based on the RSA [Sho00] and discrete logarithm [Bol03] based problems.

Definition 40 (Threshold Signature Scheme [Sho00])**.** *A (n,t)-threshold signatures scheme is defined by a tuple of algorithms $(\mathbf{TGen}, \mathbf{TSign}, \mathbf{TRec}, \mathbf{TVerify})$. The key generation algorithm* **TGen** *takes the security parameter 1^λ as input and outputs a verification key vk and secret keys $sk_1, \ldots, sk_n$. The (possibly interactive) signing algorithm* **TSign** *takes a secret key sk_i and a message $m \in \mathcal{M}$ as input and after potentially interacting with the other parties it outputs a signature share σ_i. The reconstruction algorithm* **TRec** *takes the verification key vk, any t signature shares, and outputs a signature σ. The verification algorithm* **TVerify** *takes a signature σ, a message m, and a verification key vk as input and outputs a bit $b \in \{0,1\}$. We call a threshold signature scheme secure if the following holds:*

1. ***Correctness.*** *Any authorized set of parties can generate a valid signature. That is, for any set $T = \{i_1, \ldots, i_t\}$ of size at least t and for any $m \in \mathcal{M}$, it holds that*

   $$\Pr[\mathbf{TVerify}(vk, \mathbf{TRec}(vk, \sigma_{i_1}, \ldots, \sigma_{i_t}), m) = 1] = 1\,,$$

   where $\sigma_i \leftarrow \mathbf{TSign}(sk_i, m)$ and $(vk, sk_1, \ldots, sk_n) \leftarrow \mathbf{TGen}(1^\lambda)$.

2. ***Unforgeability.*** *No collusion of unauthorized parties can forge a signature. More formally, we consider a probabilistic polynomial time adversary A, who can corrupt up to $t-1$ parties to learn their secret keys. The adversary may, on behalf of the corrupt parties, engage in a polynomial number of (possibly interactive) signature share generations with the honest parties for messages of its choice. Let Q be the set of messages that the adversary signs in this fashion. We require that the probability of A outputting a valid message signature pair (m^*, σ^*) with $m^* \notin Q$ is negligible in λ.*

In this work we extend the notion of threshold signatures in two directions. We propose non-malleable as well as leakage-resilient threshold signatures. These two separate notions require that a threshold signature scheme remains secure even if tampering or leakage on the secret keys of each player occurs. Throughout this section we assume a asynchronous communication network with eventual delivery. In such a network each message can be delayed arbitrarily, but it is guaranteed that any sent message eventually arrives at its destination. We also assume that any pair of parties is connected by a secure point-to-point channel.

5.1 Non-Malleable Threshold Signatures

A non-malleable threshold signature scheme requires that even an adversary, who obtains a polynomial number of signature shares under tampered keys for messages of its choice, may not produce a valid forgery. We model this security guarantee as follows:

Definition 41 (Non-Malleable Threshold Signature Scheme)**.** *Let $\mathcal{S} = (\mathbf{NMTGen}, \mathbf{NMTSign}, \mathbf{NMTRec}, \mathbf{NMTVerify})$ be a secure threshold signature scheme according to Definition 40. Let $\mathcal{F}$ be some family of tampering functions. For each $f \in \mathcal{F}$, and any probabilistic polynomial time adversary A, define the tampering experiment*

$$\mathsf{SigTamper}_\lambda^f = \left\{ \begin{array}{l} (vk, sk_1, \ldots, sk_n) \leftarrow \mathbf{NMTGen}(1^\lambda) \\ (\widetilde{sk}_1, \ldots, \widetilde{sk}_n) \leftarrow f(sk_1, \ldots, sk_n) \\ (i_1, \ldots, i_{t-1}) \leftarrow A(1^\lambda) \\ (m^*, \sigma^*) \leftarrow A^{\widetilde{\mathcal{O}}}(vk, \widetilde{sk}_{i_1}, \ldots, \widetilde{sk}_{i_{t-1}}) \\ Output\ (m^*, \sigma^*) \end{array} \right\},$$

where the oracle $\widetilde{\mathcal{O}}(\cdot) = (\mathbf{NMTSign}(\widetilde{sk}_1, \cdot), \ldots, \mathbf{NMTSign}(\widetilde{sk}_n, \cdot))$ allows the adversary to obtain a polynomial number of (honestly generated) signature shares generation for messages of its choice. Let Q be the set of messages that A queries to $\widetilde{\mathcal{O}}$. We say $\mathcal{S}$ is non-malleable w.r.t. $\mathcal{F}$ if for all $f \in \mathcal{F}$

$$\Pr[\mathbf{NMTVerify}(vk, \mathbf{TRec}(vk, \sigma^*, m^*) = 1 \ \wedge\ m^* \notin Q] \leq \mathsf{negl}(\lambda)\,.$$

Our construction follows the same blueprint as our non-malleable secret sharing schemes.

Theorem 42. *For any number of parties $n \geq 2t + 1$ and threshold t, if we have the following primitives :*

1. *A non-interactive[2] secure (n, t)-threshold signatures scheme* $(\mathbf{TGen}, \mathbf{TSign}, \mathbf{TRec}, \mathbf{TVerify})$.
2. *A coding scheme* $(\mathbf{NMEnc}, \mathbf{NMDec})$ *that is ε-non-malleable w.r.t $\mathcal{F}_2^{split}$, where $\varepsilon \leq \mathsf{negl}(\lambda)$.*

then there exists a non-malleable threshold signature scheme w.r.t. $\mathcal{F}_n^{split}$.

We construct a non-malleable threshold signature scheme $\mathcal{S} = (\mathbf{NMTGen}, \mathbf{NMTSign}, \mathbf{NMTRec}, \mathbf{NMTVerify})$ as follows.

NMTGen: Our key generation function takes the security parameter 1^λ as its input and proceeds as follows:

1. $(vk, sk'_1, \ldots, sk'_n) \leftarrow \mathbf{TGen}(1^\lambda)$
2. For each $i \in [n]$, encode the key sk'_i to obtain $(L_i, R_i) \leftarrow \mathbf{NMEnc}(sk'_i)$;
3. For each $i \in [n]$, construct $sk_i = (R_1, \ldots, R_{i-1}, L_i, R_{i+1}, \ldots, R_n)$;
4. Output $(vk, sk_1, \ldots, sk_n)$.

NMTSign: Party i with secret $sk_i = (R_1, \ldots, R_{i-1}, L_i, R_{i+1}, \ldots, R_n)$ constructs its signature share as follows:

1. Request R_i from all other parties and wait for the first $n-t$ responses $(R_i^1, \ldots, R_i^{n-t})$.
2. Check whether $R_i^1 = \cdots = R_i^{n-t}$ and output $\perp$ if not.
3. Reconstruct the secret key $sk' \leftarrow \mathbf{NMDec}(L_i, R_i^1)$ and output $\perp$ if $sk' = \perp$.
4. Compute signature share $\sigma_i \leftarrow \mathbf{TSign}(sk'_i, m)$.
5. Output σ_i.

[2] We call a threshold signature scheme non-interactive if every party can generate a signature share without interacting with the other parties. Many existing schemes are of this form, see for example [Sho00, Bol03]

NMTRec: Given verification key vk and signature shares $\sigma_{i_1}, \ldots, \sigma_{i_t}$, we construct a signature as follows:

1. $\sigma \leftarrow \mathbf{TRec}(vk, \sigma_{i_1}, \ldots, \sigma_{i_t})$.
2. Output σ.

NMTVerify: Given verification key vk, signature σ, and message m, we do the following:

1. $b \leftarrow \mathbf{TVerify}(vk, \sigma, m)$.
2. Output b.

Notice that the way **NMTSign** is formulated now, a single tampered share can make the protocol output $\perp$. If this is undesirable, the two first steps in **NMTSign**: can be replaced by

1. Request R_i from all other parties and collect responses $R_i^1, R_i^2, \ldots$.
2. If and when a subset of the responses of size $n-t$ are all identical to some R_i, use this R_i in the following steps.

In an asynchronous network with eventual delivery, all $n-t$ honest parties will eventually get the request for R_i and send their value. Therefore party i eventually receive all these $n-t$ shares (and possibly some corrupted shares too). Therefore, if there is no tampering, then party i will eventually receive $n-t$ copies of the correct share. In all cases party i will hear from at least one honest party as in the original scheme, so security follows along the lines of the security for the original scheme. Below we will only analyse the original scheme.

Unforgeability: We show that $\mathcal{S}$ is a secure threshold signature scheme according to definition 40. Let $\mathsf{SuccForgery}$ be the event that adversary A successfully outputs a valid forgery in the unforgeability game from Definition 40. We have

$$\Pr[\mathsf{SuccForgery}] = \Pr[\mathsf{SuccForgery}|\mathsf{SuccCheat}] + \Pr[\mathsf{SuccForgery}|\neg\mathsf{SuccCheat}],$$

where $\mathsf{SuccCheat}$ denotes the event that during any of the signature share generations a corrupt party sends a tampered value R_i^*, such that $R_i^* \neq R_i$, to honest party i and this party does not output $\perp$. Recall that any party i waits for $n-t$ responses in step 1 of **NMTSign**. Since $n \geq 2t+1$, it holds that at least one of the $n-t$ parties is honest and thus sends the correct R_i. From step 2 it follows that $\Pr[\mathsf{SuccCheat}] = 0$.

Let us now consider $\Pr[\mathsf{SuccForgery}|\neg\mathsf{SuccCheat}]$. We first observe that each party holds one share of each signing key. This means that the adversary can see one share of each of the $n-(t-1)$ honest party's secret keys. Now consider a hybrid game, which is almost identical to the original unforgeability game with the only difference being that we slightly change the keys of the corrupted parties. Rather than letting them hold one share of each honest party's secret key, we let them hold shares of a random values. By lemma 21 we know that $(\mathbf{NMEnc}, \mathbf{NMDec})$ is a $(2, 2\varepsilon)$-secret sharing scheme and thus any adversary can distinguish the security games with probability at most $2(n-t)\varepsilon$. Since no tampering happens, and since the corrupted parties now hold no information about the honest parties keys, we can conclude that the resulting game is basically identical to the original unforgeability game and thus by the security of the underlying threshold signature scheme we get that

$$\Pr[\mathsf{SuccForgery}] = \Pr[\mathsf{SuccForgery}|\neg\mathsf{SuccCheat}] \leq \mathsf{negl}(\lambda) + 2\varepsilon(n-t+1) \leq (2n-2t+3)\mathsf{negl}(\lambda).$$

Non-Malleability: Assume towards contradiction that the construction described above is not non-malleable according to Definition 41. This means that for some fixed $f \in \mathcal{F}_n^{split}$, there exists an adversary A that successfully outputs a forgery in the experiment $\mathsf{SigTamper}_\lambda^{f}$ with non-negligible probability. We will use A to construct an adversary B that breaks the unforgeability of the underlying threshold signature scheme. Our reduction works as follows:

1. B internally initializes A with fresh random coins.

2. According to $\mathsf{SigTamper}^{f}_{\lambda}$, the adversary A outputs indices $T = (i_1, \ldots, i_{t-1})$.
3. B forwards these indices in his unforgeability game to obtain secret keys $(vk, sk'_{i_1}, \ldots, sk'_{i_{t-1}})$.
4. B picks $P \leftarrow \{0,1\}^\lambda$ and for each $i \in [n]$, if $i \notin T$, then B sets $sk'_i = P$.
5. For each $i \in [n]$, encode the key sk'_i to obtain $(L_i, R_i) \leftarrow \mathbf{NMEnc}(sk'_i)$.
6. For each $i \in [n]$, B constructs $sk_i = (R_1, \ldots, R_{i-1}, L_i, R_{i+1}, \ldots, R_n)$.
7. B computes $(\widetilde{sk}_1, \ldots, \widetilde{sk}_n) \leftarrow f(sk_1, \ldots, sk_n)$ and sends $(vk, \widetilde{sk}_{i_1}, \ldots, \widetilde{sk}_{i_{t-1}})$ to A.
8. B simulates A's queries to oracle $\widetilde{\mathcal{O}}$. Whenever A sends message m, B returns $(\sigma_1, \ldots, \sigma_n)$, which is computed as follows:
   (a) For each $i \in [n]$, B reconstructs $\widetilde{sk'}_i$ as is done in the first three steps of **NMTSign**.
   (b) For each $i \in T$, if $\widetilde{sk'}_i = \perp$, then set $\sigma_i = \perp$, otherwise set $\sigma_i = \mathbf{TSign}(\widetilde{sk'}_i, m)$.
   (c) For each $i \notin T$, if $\widetilde{sk'}_i = \perp$, then set $\sigma_i = \perp$, otherwise if $\widetilde{sk'}_i \neq P$, then $\sigma_i = \mathbf{TSign}(\widetilde{sk'}_i, m)$, otherwise if $\widetilde{sk'}_i = P$, then B queries his oracle to obtain signature share σ_i.
9. At some point A outputs (m^*, σ^*) and B outputs the same.

From the perspective of A, the only difference between our reduction and a real execution $\mathsf{SigTamper}^{f}_{\lambda}$ is the construction of each R_i where $i \notin T$. For this observation we use (in step 8 (c) of the reduction) the fact that the underlying threshold signature scheme is non-interactive, which means that correctness of a signature share σ_i only depends on the correctness of $\widetilde{sk}_i$. By lemma 21, we know that (**NMEnc**, **NMDec**) is a $(2, 2\varepsilon)$-secret sharing scheme and therefore the adversary's success probability can at most differ by an additive factor of $2\varepsilon(n - t + 1)$. It follows that

$$\Pr[B \text{ wins}] \geq \Pr[A \text{ wins}] - 2\varepsilon(n - t + 1) \geq= \Pr[A \text{ wins}] - (2n - 2t + 3)\mathsf{negl}(\lambda),$$

which by assumption on A's success probability is non-negligible.

5.2 Leakage-Resilient Threshold Signatures

In a leakage-resilient threshold signature scheme, the adversary may obtain an unqualified subset of secret keys and a bounded amount of leakage from *all* other secret keys. Even given this information, we require that the adversary may not be able to output a valid forgery.

Definition 43 (Leakage-Resilient Threshold Signature Scheme). *Let $\mathcal{S} = (\mathbf{LTGen}, \mathbf{LTSign}, \mathbf{LTRec}, \mathbf{LTVerify})$ be a tuple of probabilistic polynomial time algorithms. Let $\mathcal{F}$ be a family of leakage functions. For each $f \in \mathcal{F}$, and any probabilistic polynomial time adversary A, define the following experiment*

$$\mathsf{SigLeak}^{f}_{\lambda} = \left\{ \begin{array}{l} (vk, sk_1, \ldots, sk_n) \leftarrow \mathbf{LTGen}(1^\lambda) \\ (i_1, \ldots, i_{t-1}) \leftarrow A(1^\lambda) \\ (\ell_1, \ldots, \ell_n) \leftarrow f(sk_1, \ldots, sk_n) \\ (m^*, \sigma^*) \leftarrow A^{\mathcal{O}}(vk, (sk_{i_1}, \ldots, sk_{i_{t-1}}), (\ell_1, \ldots, \ell_n)) \\ \textit{Output } (m^*, \sigma^*) \end{array} \right\},$$

where the oracle $\mathcal{O}(\cdot)$ allows the adversary, on behalf of the corrupted parties, to engage in a polynomial number of (possibly interactive) signature shares generation for messages of its choice. Let Q be the set of messages that A queries to $\mathcal{O}$. We say $\mathcal{S}$ is leakage-resilient w.r.t. $\mathcal{F}$ if for all $f \in \mathcal{F}$

$$\Pr[\mathbf{NMTVerify}(vk, \mathbf{TRec}(vk, \sigma^*, m^*) = 1 \ \wedge\ m^* \notin Q] \leq \mathsf{negl}(\lambda)\,.$$

Theorem 44. *For any number of parties $n \geq 2t+1$ and threshold t, if we have the following primitives :*

1. *A non-interactive secure (n,t)-threshold signatures scheme* $(\mathbf{TGen}, \mathbf{TSign}, \mathbf{TRec}, \mathbf{TVerify})$.
2. *A two-source $(n - \ell - \log 1/\varepsilon, 2\varepsilon)$-extractor* **nmExt** *with efficient preimage sampling from the space $\mathcal{X} = \{0,1\}^n$, where $\varepsilon \leq \mathsf{negl}(\lambda)$.*

then the construction from Theorem 42, where we replace each call to **NMEnc** *with* $\mathbf{nmExt}^{-1}$ *and each call to* **NMDec** *with* **nmExt**, *is a leakage-resilient threshold signature scheme w.r.t. $\mathcal{F}^{split}_{\ell,n}$, where $\mathcal{F}^{split}_{\ell,n}$ is the set of leakage functions that tamper with each share independently and the output of each tampering function is bounded in size by ℓ bits.*

Assume towards contradiction that it is not. This means that for some fixed $f \in \mathcal{F}^{split}_{\ell,n}$, there exists an adversary A that successfully outputs a forgery in the experiment $\mathsf{SigLeak}^{f}_{\lambda}$ with non-negligible probability. We will use A to construct an adversary B that breaks the unforgeability of the underlying threshold signature scheme. Our reduction works as follows:

1. B internally initializes A with fresh random coins.
2. The adversary A outputs indices $T = (i_1, \ldots, i_{t-1})$.
3. B forwards these indices in his unforgeability game to obtain secret keys $(vk, sk'_{i_1}, \ldots, sk'_{i_{t-1}})$.
4. For each $i \in [n]$, if $i \notin T$, then B sets $sk'_i = 0$.
5. For each $i \in [n]$, encode the key sk'_i to obtain $(L_i, R_i) \leftarrow \mathbf{nmExt}^{-1}(sk'_i)$.
6. For each $i \in [n]$, B constructs $sk_i = (R_1, \ldots, R_{i-1}, L_i, R_{i+1}, \ldots, R_n)$.
7. B computes $(\ell_1, \ldots, \ell_n) \leftarrow f(sk_1, \ldots, sk_n)$ and sends $(vk, (sk_{i_1}, \ldots, sk_{i_{t-1}}), (\ell_1, \ldots, \ell_n))$ to A.
8. Whenever A initiates a signature share generation for honest party i on message m, the honest party would request R_i from A and so does B. A outputs a value R^*_i and we consider two cases here:
   - If $R^*_i = R_i$, where R_i is known to B, then B requests a signature share generation on m from honest party i in his game and returns the result σ_i to A.
   - If $R^*_i \neq R_i$, then we request a scheduler entity[3] to specify an order in which messages at honest party i should arrive. If A's message is among the first $n-t$, then we return $\perp$ to A. If not, then B queries m in his game and again returns σ_i to A.
9. At some point A outputs (m^*, σ^*) and B outputs the same.

From the perspective of A, the only difference between our reduction and a real execution $\mathsf{SigLeak}^{f}_{\lambda}$ is the construction of each R_i as well as the corresponding leakage ℓ_i, where $i \notin T$. Since each honest party's secret sk'_i is encoded with a two-source $(n-\ell-\log 1/\varepsilon, 2\varepsilon)$-extractor, the adversary can distinguish a real execution of the experiment and the reduction with probability at most $4\varepsilon(n-t+1)$. It follows that

$$\Pr[B \ \mathsf{wins}] \geq \Pr[A \ \mathsf{wins}] - 4\varepsilon(n-t+1) \geq= \Pr[A \ \mathsf{wins}] - (4n - 4t + 3)\mathsf{negl}(\lambda),$$

which by assumption on A is non-negligible.

References

[ADKO15] Divesh Aggarwal, Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Leakage-resilient non-malleable codes. In *Theory of Cryptography*, volume 9014 of *Lecture Notes in Computer Science*, pages 398–426. Springer Berlin Heidelberg, 2015.

[3]Recall that we consider a asynchronous network with eventual delivery. This means that we cannot assume a specific order in which messages arrive. Therefore a (potentially malicious) scheduler can specify any order for us.

[BDIR18] Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. In *Annual International Cryptology Conference*, pages 531–561. Springer, 2018.

[Bei11] Amos Beimel. Secret-sharing schemes: a survey. In *International Conference on Coding and Cryptology*, pages 11–46. Springer, 2011.

[BGK14] Elette Boyle, Shafi Goldwasser, and Yael Tauman Kalai. Leakage-resilient coin tossing. *Distributed Computing*, 27(3):147–164, 2014.

[Bol03] Alexandra Boldyreva. Efficient threshold signatures, multisignatures and blind signatures based on the Gap-Diffie-Hellman-group signature scheme. In *PKC 2003, 6th International Workshop on Theory and Practice in Public Key Cryptography*, volume 2567 of *LNCS*. Springer-Verlag, 2003.

[BS18] Saikrishna Badrinarayanan and Akshayaram Srinivasan. Revisiting non-malleable secret sharing. 2018. unpublished.

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

[CGL16] Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. In *Proceedings of the forty-eighth annual ACM symposium on Theory of Computing*, pages 285–298. ACM, 2016.

[CZ14] Eshan Chattopadhyay and David Zuckerman. Non-malleable codes in the constant split-state model. *FOCS*, 2014.

[Des87] Yvo Desmedt. Society and group oriented cryptography: A new concept. In Carl Pomerance, editor, *Advances in Cryptology—CRYPTO '87*, volume 293 of *LNCS*, pages 120–127. Springer-Verlag, 1988, 16–20 August 1987.

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

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS*, pages 434–452. Tsinghua University Press, 2010.

[FMNV14] S. Faust, P. Mukherjee, J. Nielsen, and D. Venturi. Continuous non-malleable codes. In *Theory of Cryptography Conference - TCC*. Springer, 2014.

[GK18a] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing*, pages 685–698. ACM, 2018.

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

[GKP+18] Vipul Goyal, Ashutosh Kumar, Sunoo Park, Silas Richelson, and Akshayaram Srinivasan. Non-malleable commitments from non-malleable extractors. 2018. unpublished.

[GUV09] Venkatesan Guruswami, Christopher Umans, and Salil P. Vadhan. Unbalanced expanders and randomness extractors from parvaresh–vardy codes. *J. ACM*, 56(4), 2009.

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

[Kum18] Ashutosh Kumar. personal communication, 2018.

[Li17] Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pages 1144–1156. ACM, 2017.

[Sho00] Victor Shoup. Practical threshold signatures. In Bart Preneel, editor, *Advances in Cryptology—EUROCRYPT 2000*, volume 1807 of *LNCS*, pages 207–220. Springer-Verlag, 2000.

[SV18] Akshayaram Srinivasan and Prashant Nalini Vasudevan. Leakage resilient secret sharing and applications. 2018. unpublished.