# Non-Malleable Secret Sharing

Vipul Goyal [*] Ashutosh Kumar [†]

**Abstract**

A number of works have focused on the setting where an adversary tampers with the shares of a secret sharing scheme. This includes literature on verifiable secret sharing, algebraic manipulation detection(AMD) codes, and, error correcting or detecting codes in general. In this work, we initiate a systematic study of what we call *non-malleable secret sharing.* Very roughly, the guarantee we seek is the following: the adversary may potentially tamper with all of the shares, and still, *either* the reconstruction procedure outputs the original secret, *or*, the original secret is "destroyed" and the reconstruction outputs a string which is completely "unrelated" to the original secret. Recent exciting work on non-malleable codes in the split-state model led to constructions which can be seen as 2-out-of-2 non-malleable secret sharing schemes. These constructions have already found a number of applications in cryptography. We investigate the natural question of constructing $t$-out-of-$n$ non-malleable secret sharing schemes. Such a secret sharing scheme ensures that only a set consisting of $t$ or more shares can reconstruct the secret, and, additionally guarantees non-malleability under an attack where potentially *every* share maybe tampered with. Techniques used for obtaining split-state non-malleable codes (or 2-out-of-2 non-malleable secret sharing) are (in some form) based on two-source extractors and seem not to generalize to our setting.

- Our first result is the construction of a $t$-out-of-$n$ non-malleable secret sharing scheme against an adversary who arbitrarily tampers each of the shares *independently.* Our construction is unconditional and features statistical non-malleability.
- As our main technical result, we present $t$-out-of-$n$ non-malleable secret sharing scheme in a stronger adversarial model where an adversary may *jointly* tamper multiple shares. Our construction is unconditional and the adversary is allowed to jointly-tamper subsets of up to $(t-1)$ shares. We believe that the techniques introduced in our construction may be of independent interest.

Inspired by the well studied problem of perfectly secure message transmission introduced in the seminal work of Dolev et. al (J. of ACM'93), we also initiate the study of *non-malleable message transmission.* Non-malleable message transmission can be seen as a natural generalization in which the goal is to ensure that the receiver either receives the original message, or, the original message is essentially destroyed and the receiver receives an "unrelated" message, when the network is under the influence of an adversary who can byzantinely corrupt all the nodes in the network. As natural applications of our non-malleable secret sharing schemes, we propose constructions for non-malleable message transmission.

[*]CMU. goyal@cs.cmu.edu . The author is supported by a grant from Northrop Grumman.

[†]UCLA. a@ashutoshk.com A part of this work was done while the second author was at Microsoft Research, India. Work done at UCLA is supported in part from NSF grant 1619348, NSF frontier award 1413955, US-Israel BSF grants 2012366, 2012378, and by the Defense Advanced Research Projects Agency (DAPRA) SAFEWARE program through the ARL under Contract W911NF-15-C-0205 and through a subcontract with Galois, inc. The views expressed are those of the authors and do not reflect the official policy or position of the Department of Defense, the National Science Foundation, or the U.S. Government.

## 1 Introduction

Secret sharing is a fundamental primitive in cryptography which allows a dealer to distribute shares of a secret among several parties, such that only authorized subsets of parties can recover the secret; the secret is "hidden" from all the unauthorized set of parties. Shamir [Sha79] and Blakley [Bla79] initiated the study of secret sharing by constructing threshold secret sharing schemes that only allows at least $t$-out-of-$n$ parties to reconstruct the secret.

Secret sharing schemes as envisioned originally were concerned with the privacy of the secret, however, *what if the primary goal of the adversary is to tamper with the secret instead?* Here one could ask for a number of different guarantees. For example, one could ask that the correct secret is recovered even when some number of shares are arbitrarily corrupted. Concepts from error correcting codes have been useful in obtaining such robust secret sharing schemes. Similar related guarantees can be found in the lines of work on: error detecting codes such as algebraic manipulation detection(AMD) codes [CDF$^+$08], and, verifiable secret sharing [RBO89]. A more detailed overview of the related works can be found later in this section.

**Non-malleable secret sharing.** In this work, we initiate a systematic study of what we call *non-malleable secret sharing.* Very roughly, the guarantee we seek is the following: the adversary may potentially tamper with all of the shares, and still, *either* the reconstruction procedure outputs the original secret, *or*, the original secret is "destroyed" and the reconstruction outputs a string which is completely "unrelated" to the original secret. This is a natural guarantee which is inspired by applications in cryptography. Before giving more details, a discussion of the relationship of this notion with split state non-malleable codes is in order.

**Non-malleable codes in the split state model.** In a beautiful work Dziembowski et al. [DPW10] introduced non-malleable codes and showed the existence of such codes against a broad family of tampering functions. Non-malleable codes guarantee that tampering with the code can only cause the reconstruction procedure to either output the original message or an "unrelated" one. A number of subsequent works have continued to study non-malleable codes in various tampering models. Perhaps the most well-known of these models is the so called split state model. At a high level this means that the codeword has two parts and the adversary is allowed to tamper with both the parts arbitrarily and independently. This model was proposed by Liu and Lysyanskaya [LL12] who also presented a construction based on (strong) cryptographic assumptions.

Constructing split state non-malleable codes without cryptographic assumptions proved to be surprisingly hard. Dziembowski et al. [DKO13] proposed a construction which could encode single bit messages. A subsequent brilliant lines of works resulted in constructions for multi-bit messages as well [ADL14, CGL16, Li17].

Split state non-malleable codes do not explicitly require that the message remain a secret given only one of the two states. However, it is not difficult to see that any 2 split-state non-malleable code is also a 2-out-of-2 secret sharing scheme. Thus, the perspective we follow in our work would be to *view the above constructions as 2-out-of-2 non-malleable secret sharing schemes.* Note that such an implication may not hold if the number of states is more than 2. To see this, consider a contrived example of a 3 split-state non-malleable code where the encoding functions encodes the message using a 2 split-state non-malleable code to obtain the first two states and outputs the message (in the clear) in the third state. The decoding function simply ignores the third state and uses the first two states to decode the message. Such a construction is a valid 3 split-state non-malleable code that is not a 3-out-of-3 secret sharing scheme (in fact, it has no secrecy at all).

Even though non-malleable code is a relatively new primitive, it has already found a number of applications in cryptography including in tamper-resilient cryptography [DPW10], designing multi-prover interactive proof systems [GJK15] and obtaining efficient encryption schemes [CDTV16]. Very recently, non-malleable codes in the split-state model were used as 2-out-of-2 non-malleable secret sharing scheme to obtain 3-round protocol for non-malleable commitments [GPR16] (a problem which was open for a while).

**Our Question.** We study the following natural question in this work:

*Can we get t-out-of-n non-malleable secret sharing schemes in which adversary can tamper (in some form) with all the shares?*

As noted before, split state non-malleable codes provide a positive answer to the above question for the special case of $t = n = 2$. These constructions have already found a number of applications in cryptography. However to our knowledge, a construction for general values of $t$ and $n$ is currently unknown (even from cryptographic assumptions).

*The techniques used for obtaining 2-out-of-2 non-malleable secret sharing schemes are, in some form or the other, based on 2-source (or multi-source) extractors, and, seem not to generalize to our case.* Almost by definition, all the sources are required to compute the extractor output (which, presumably would be used in the reconstruction phase of the non-malleable secret sharing). Such an idea would fail in our setting where all the sources (i.e., shares) may not be available during reconstruction.

**Existing secret-sharing schemes.** Most of the secret sharing schemes known are linear [Bei, chapter 4] and have nice algebraic and geometric properties, which are harnessed to obtain efficient sharing and reconstruction procedures. *Non-malleable secret sharing schemes on the other hand cannot be linear.* To see this, consider a linear secret sharing scheme, in which the secret is a linear combination of the authorized shares. Now if an adversary multiplies each of the authorized shares by 2, the secret, which is a linear combination, also gets multiplied by 2 and non-malleability is lost. In fact, it is easy to see that for any authorized set of shares of linear schemes, the adversary can add an arbitrary value of its choice to the secret by changing only one of the shares. Indeed, the malleability of linear secret sharing schemes, such as polynomials based Shamir's secret sharing scheme [Sha79], forms the basis of secure multi-party computation protocols [BOGW88]. For our purposes, any such alteration is an "attack" and we try to build secret sharing schemes that necessarily prohibit any such attacks.

## 1.1 Our Results

As our first technical contribution, we construct t-out-of-n non-malleable secret sharing scheme that allows a computationally unbounded adversary to arbitrarily tamper with each of the shares *independently.* We note that prior to this work, even for computationally bounded adversaries, no threshold (i.e. t-out-of-n) non-malleable secret sharing schemes were known.

**(Informal) Theorem 1.** *For any threshold $t \geq 2$, any number of parties $n \geq t$, there exists an efficient*[1] *t-out-of-n statistical secret sharing scheme that is statistically non-malleable against a computationally unbounded adversary who tampers each of the shares arbitrarily and independently.*

[1]A statistical secret sharing scheme is efficient if the sharing and reconstruction functions run in $\mathbf{poly}\big(n, k, \log(1/\epsilon)\big)$ time where $k$ is the size of the message and $\epsilon > 0$ is the statistical error. For concrete parameters, please see section 3.

In the above model, the adversary specifies a separate tampering function for each of the $t$ shares that it tampers with. The reconstruction function takes the $t$ tampered shares and outputs a message (which should either be identical or unrelated to the original one). We refer to this as the *individual tampering model*. Next, we consider what we call the *joint tampering model* where the adversary may tamper jointly with several shares. It is easy to see that if the adversary can tamper jointly with $t$ or more shares, it can simply reconstruct the original message, and, replace the original shares with the shares of a related message of its choice. Thus, one could hope to achieve non-malleability only if the adversary can jointly tamper up to $t-1$ shares.

As our main technical contribution, we obtain the following result in the joint tampering model.

**(Informal) Theorem 2.** *For any threshold $t \geq 2$ and number of parties $n \geq t$, there exists an efficient $t$-out-of-$n$ statistical secret sharing scheme that is statistically non-malleable against a computationally unbounded adversary who chooses any arbitrary authorized set of $t$ shares, partitions it into two unequal non-empty subsets*[2] *and then jointly tampers the shares in each of these subsets arbitrarily and independently.*

While theorem 1 is subsumed by theorem 2, it nonetheless serves as a useful starting point for illustrating our ideas. We also remark that our construction for individual tampering features a significantly better rate than the one for joint tampering.

**Leakage Resilient Non-Malleable Codes.** To achieve our results on joint tampering, we need non-malleable codes that satisfy a strong leakage-resilience requirement. Specifically, we need 2 split-state non-malleable codes, where the tampering of first part (i.e. state) not only depends on the first part, but also on bounded information (leakage) from the second part. The non-trivial part is that the bound on the bits of leakage from the second part can be as large as the size of the first part. Unfortunately, all known constructions of split-state non-malleable codes [DKO13, ADL14, CGL16, Li17] (even leakage-resilient ones [ADKO15, GKP+18]) have two parts of equal size and can be trivially broken in our leakage model. We construct such codes and believe that it will be useful in other cryptographic applications. [3]

**(Informal) Theorem 3.** *For any polynomial $p$, there exists an efficient coding scheme that encodes a message into two parts $(l, r)$ and is statistically non-malleable against an adversary who tampers the first share as $f(l, leak(r))$ and the second share as $g(r)$, where $f$ and $g$ are arbitrary tampering functions and $leak(r)$ is an arbitrary function that outputs $p(|l|)$ bits of information about $r$.*

**Non-Malleable Message Transmission.** As an application of non-malleable secret sharing schemes, we initiate the study of non-malleable message transmission. While the existing works on message transmission have been primarily concerned with ensuring reliability and secrecy of the message when a bounded number of nodes are corrupted [DDWY93, SNR04, WD08, KS09, KKVS18], non-malleable message transmission tries to ensure non-malleability against adversaries which can corrupt the entire network. In other words, it ensures that the receiver either receives the original message, or, the original message is essentially destroyed and the receiver receives an "unrelated"

[2] One cannot hope to achieve any non-malleability in case adversary gets all the $t$ shares entirely. Therefore, we make the necessary assumption that subsets are non-empty (proper). Further, we need these subsets to have different number of shares for our techniques to work. It is easy to see that if either the threshold $t$ is odd or the number of subsets is more than two, then such a requirement is trivially satisfied.

[3] Very recently, Goyal et al. [GKP+18] crucially used a weaker version of leakage-resilient non-malleable codes to obtain a three round concurrent non-malleable commitment protocol that is non-malleable w.r.t. replacement.

message, when the network is under the influence of an adversary who can execute arbitrary protocol on each of the nodes in the network (apart from the sender and the receiver).

We even allow the adversary to add a bounded number of arbitrary hidden links which it can use in addition to the original links for communicating amongst corrupt nodes. We now informally define non-malleable paths to help us obtain our protocol which not only ensures non-malleability against such an adversary, but also guarantees that only the receiver learns the secret. For a network represented by an undirected graph $G$, let $G'$ be the induced subgraph of $G$ with sender $S$ and $R$ removed. We define a collection of paths from $S$ to $R$ to be non-malleable if in the induced subgraph $G'$ any node is reachable by nodes present on at most one of these paths.

**(Informal) Lemma 1.** *In any network, with a designated sender $S$ and receiver $R$, if there exists a collection of $n$ non-malleable paths from $S$ to $R$, then non-malleable secure message transmission protocol is possible with respect to an adversary which adds at most $n-3$ arbitrary hidden links in the network and byzantinely corrupts all nodes other than $S$ and $R$* [4].

We have used our $n$-out-of-$n$ scheme to arrive at the above result. We can use our threshold non-malleable secret sharing schemes, and extend the above protocol to additionally ensure that message is correctly recovered in case the adversary only crashes a bounded number of nodes.

## 1.2 Our Techniques

There are a number of constructions of non-malleable codes in the split-state model, all having intricate proofs of non-malleability [DKO13, ADL14, CGL16, Li17]. All of these constructions crucially use techniques from 2-source or multi-source extractors. Almost by definition, all the sources are required to compute the extractor output (which, presumably would be used in the reconstruction phase of the non-malleable code). Such an idea would fail in our setting where all the sources (i.e., shares) may not be available during reconstruction.

**Non-Malleable Secret Sharing with respect to individual tampering.** We highlight the main technical ideas, while trying to build a $t$-out-of-$n$ secret sharing that is non-malleable with respect to individual tampering. As a building block, we use a split-state non-malleable code that encodes a message into two parts, say $l$ and $r$. As these codes are only secure if the two parts are tampered independently, we have to ensure that independent tampering of shares of our scheme can be transformed to an independent tampering of $l$ and $r$. As a first attempt, such independence can be achieved if for a given set of $t$ shares used for reconstruction, the first share has information of $l$, while the other $t-1$ shares has information about $r$. However upon realizing that any set of $t$ shares should allow for reconstruction, such approaches break down. This hints towards a possible approach in which each of $n$ shares of the secret sharing scheme has shares of *both* $l$ and $r$. We describe the two main ideas behind our construction:

1. **Getting non-malleability from secret sharing schemes with different parameters.** We observe that two secret sharing schemes requiring different number of shares for reconstruction have "some" non-malleability with respect to each other. In more detail, suppose the adversary is given shares of a secret $s_1$ under a $t$-out-of-$n$ secret sharing scheme, and, is required to output shares of a secret $s_2$ under $(t-1)$-out-of-$n$ secret sharing scheme, such that each share of $s_2$ is obtained by tampering a single share of $s_1$. Then, we can conclude that $s_2$ is independent of $s_1$. This is because $(t-1)$ shares of $s_2$ can depend only upon $(t-1)$

[4] We in fact allow the adversary to add $2\lceil \frac{n}{2} \rceil - 3$ hidden links, which is tight for odd values of $n$

shares of $s_1$ (which in turn would have no information about $s_1$). This can be also be seen as concluding that polynomials of different degree have "some" non-malleability w.r.t. each other. This is a very natural (and yet powerful) idea, and to our knowledge has not been exploited before. Now we provide further details.
We wish to to build a $t$-out-of-$n$ non-malleable secret sharing scheme. Say $t > 2$. To share $m$, we encode $m$ using the encoder of non-malleable code to obtain $l, r$. We share $l$ using a $t$-out-of-$n$ secret sharing scheme to obtain $n$ shares $l_1, \ldots, l_n$. We share $r$ using a 2-out-of-$n$ secret sharing scheme to obtain $n$ shares $r_1, \ldots, r_n$. We set each of the $n$ shares as $share_i \leftarrow (l_i, r_i)$. For reconstruction, given any $t$ tampered shares of the form $\widetilde{share_i} \leftarrow (\widetilde{l_i}, \widetilde{r_i})$, we reconstruct $\widetilde{l}$ using the $t$ shares. In the reconstruction of $\widetilde{r}$, we use only 2 shares, which is allowed since $r$ was shared using a 2-out-of-$n$ scheme. We notice, that in case of individual tampering, the reconstructed $\widetilde{r}$ can only depend on 2 shares of $l$, and is independent of the value of $l$. This follows from the security of the $t$-out-of-$n$ secret sharing scheme when $t > 2$.

2. **Getting non-malleability from leakage-resilient secret sharing.** Using the previous technique only gives us one direction of independence, that is, $\widetilde{r}$ cannot depend on the value of $l$. We need a new approach to ensure that $\widetilde{l}$, which is reconstructed using all the $t$ shares, cannot depend on the value of $r$. Approaches to use the same technique one more time unfortunately fail. To this end, we share $r$ using a leakage-resilient secret sharing scheme (which we construct), which ensures that the secret $r$ is statistically hidden even when each of the $n$ shares of $r$ leak $|l|$ bits of information. More specifically, we define a set of leakage functions, where each function takes as input a share $r_i$ and $l_i$ to obtain $share_i \leftarrow (l_i, r_i)$, tampers using the tampering function $f_i$ to obtain the tampered $\widetilde{share_i} \leftarrow f_i(share_i)$, parses $\widetilde{share_i}$ as $\widetilde{l_i}, \widetilde{r_i}$, and outputs $\widetilde{l_i}$ as the leakage. The leakage-resilience ensures that $r$ is statistically independent of the joint distribution of $n$ leakages, namely $\widetilde{l_1}, \ldots, \widetilde{l_n}$. We get the other direction of independence, that is, the reconstructed $\widetilde{l}$ is statistically independent of the value $r$, on observing that any reconstruction function for $l$ only uses values from the joint distribution $\widetilde{l_1}, \ldots, \widetilde{l_n}$.
The $k$-out-of-$n$ leakage-resilient secret sharing scheme (used as a building block in the above construction) may be a primitive of independent interest. Such a secret sharing scheme guarantees secrecy of the share even if the adversary partitions the $n$ shares into parts of size at most $k - 1$ and obtains obtains individual leakage from each of the parts. To our knowledge, $k$-out-of-$n$ leakage-resilient secret sharing schemes have not appeared so far in the literature (although the construction for $k = 2$ required for this work follows from standard techniques).

Observe that this approach fails when $t = 2$, as we cannot apply the first technique when both the schemes require the same number of shares for reconstruction. To this end, we separately build a 2-out-of-$n$ non-malleable secret sharing scheme from standard techniques in appendix A.

**Non-Malleable Secret Sharing with respect to Joint Tampering.** Let us try to use our ideas to construct a t-out-of-n secret sharing scheme that is non-malleable against joint tampering. Notice that, in our previous constructions, if we allow the adversary to jointly tamper any two shares then it can simply reconstruct $r$ (which is shared under a 2-out-of-n leakage-resilient secret sharing scheme) and use $r$ to change the shares of $l$ disabling us from obtaining a split-state reduction to the underlying non-malleable code. One way to fix this problem would be to use a k-out-of-n leakage-resilient scheme that ensures secrecy of $r$ against joint-leakage from less than $k$ shares and we might hope to achieve joint tampering of subsets containing less than $k$ shares. Therefore, it appears that the threshold $k$ could be set to a value up to $t-1$ allowing for joint tampering of up to $t-2$ shares.

Unfortunately, apart from the fact that this approach disallows for joint tampering of $t-1$ shares, there is a more fundamental limitation: even when $k$ is less than $t$, the $k$ tampered shares (used for reconstructing $r$) may now depend on all the $t$ shares (of $l$). In more detail, as our reconstruction procedure will always use the first $k$ [5] out of the $t$ shares to reconstruct $r$, and if the threshold $k$ is too high, then the adversary may partition the $t$ shares in a way that allows it to make the first $k$ shares depend on all the $t$ shares of $l$. For example, consider the simple case when $k = t-1$ and the adversary tampers the last two shares together, and all other shares individually. In such a situation the first $t-1$ tampered shares of $r$ may have information of all the $t$ shares of $l$ (in particular, the tampered $\tilde{r}$ may depend on $l$). Towards solving this problem, we take another approach and use the following additional techniques:

1. **Continue using secret-sharing schemes with different parameters for ensuring that tampering of $r$ is independent of $l$.** Without loss of generality we can assume that the adversary partitions the shares into two subsets. Our key idea will be to choose $k$ s.t. one of the two subsets has *full* information about $r$. As $r$ is shared using a $k$-out-of-$n$ secret-sharing scheme, any subset which has at least $k$ shares fixes a value of tampered $\tilde{r}$. Moreover, as no subset can have all the $t$ shares, we can argue that the fixed value of tampered $\tilde{r}$ (for that subset) has to be independent of the value of $l$.
   While we partially get one direction of independence, such an approach inherently breaks the other direction of independence required for non-malleability. In particular, the tampering function for the subset containing $k$ shares can simply reconstruct $r$ even if one uses a leakage-resilient secret-sharing scheme. Thus, the tampered shares of $l$ could now depend on $r$ (unlike the previous construction). Thus, it may seem like we have not made any progress.

2. **Getting non-malleability from leakage-resilient non-malleable codes.** Use traditional $k$-out-of-$n$ secret sharing schemes for sharing $r$ (as opposed to a leakage-resilient one) and instead derive leakage-resilience from the underlying non-malleable code. This allows the tampering of $l$ to potentially depend on $O(|l|)$ bits of information about $r$ (earlier the tampering of $l$ could not depend on even a single bit of $r$). Observe that even if the adversary reconstructs $r$, and uses its value in tampering the shares of $l$, the tampered shares of $l$ can only contain a bounded amount of information about $r$, and we may hope to leverage the leakage-resilience of the underlying non-malleable code that allows for such a bounded amount of leakage from $r$.
   To make the above approach work, the leakage must not be a function of $l$, and tampering of $l$ should only be a function of $l$ and the leakage obtained from $r$ (and not the entire $r$). However note that the tampering function gets the entire $r$ as input (since it can reconstruct and obtain $r$)! We solve this issue by assuming that the two subsets have different sizes and setting $k$ to be $1 + \lfloor \frac{t}{2} \rfloor$. Observe that such a setting ensures that exactly one of the two subsets will have $k$ (or more) shares. In other words, $r$ will be hidden in the smaller subset, and therefore, the tampered shares of $l$ of the smaller subset will be independent of the value of $r$. Now, instead of treating all the $t$ tampered shares of $l$ as leakage from $r$, we only consider the tampered shares of $l$ present in the larger subset as leakage from $r$. Notice, as the larger subset has less than $t$ shares, we have ensured that this leakage will be independent of the

[5]The reconstruction function for our t-out-of-n non-malleable secret-sharing scheme fixes which k-out-of-t shares will be used for calculating $r$. Recall that our previous construction only used the first two shares of $r$.

value of $l$.
The above approach relies on a leakage resilient non-malleable code in the split state model (where the two parts are necessarily of unequal length). All existing non-malleable codes break down in this setting, and, natural approaches to extend them to our setting seem not to work. Towards that end, we propose a new construction based on [CGL16], and outline our ideas behind it later in this subsection.

3. **The "selective bot" problem.** One remaining problem that we ignored in the description of above techniques is the following. Suppose the reconstruction function only uses some fixed $k$ (out of $t$) shares to reconstruct $r$. Then what if the adversary partitions in such a way that these $k$ shares are spread across both the subsets? Thus, potentially the tampered versions of these shares (and hence tampered $\tilde{r}$) could depend on all the $t$ shares of $l$. To solve this problem, our new reconstruction will make use of *all* the $t$ shares of $r$ and check that they are consistent under a $k$-out-of-$n$ scheme. The reconstruction procedure outputs $\perp$ if the check fails.
Trying to make this idea work, we run into the following "selective bot" problem: to know if our reconstruction outputs $\perp$ (bot) or not, all the $t$ tampered shares of $r$ must be used (which in turn may depend overall on all $t$ shares of $l$). This leads to the following possibility: the probability that the tampered shares reconstruct to $\perp$ might depend upon what the original message was. This breaks the non-malleability guarantee of our scheme. Our ideas to solve the problem are as follows:
   - **Further rely on leakage of the non-malleable codes.** Interestingly, we solve this issue by relying on the leakage resilience of the underlying non-malleable code one more time. In particular, we design an efficient randomized protocol for detection of $\perp$ with communication complexity much lower than the size of $r$ (if fact, lower than $|l|$ bits). Moreover our protocol only requires sending a single string from the larger subset to the smaller subset, and this communication can be modeled as additional leakage from the $r$ to $l$. More details follow.
   - **Interpolate the values of consistent shares.** Recall that our sharing procedure shared $r$ using a $k$-out-of-$n$ Shamir's secret sharing scheme [Sha79]. As the larger subset has at least $k$ tampered shares of $r$, using the properties of Shamir's secret sharing scheme, it is possible to interpolate the unique value of each of the $t$ shares that will be consistent with these $k$ shares. In particular, even if the larger subset has no idea of shares of $l$ in the smaller subset, it knows what the tampered shares of $r$ in the smaller subset would like in case $\perp$ does not happen. Lets concatenate these shares (corresponding to the smaller subset) to obtain an "interpolated string". On the other hand, the smaller subset concatenates its share of $r$ (in the same order) to obtain the "real string". Notice that we have reduced the problem of detecting $\perp$ to an "easier" problem of detecting inequality in between the two strings.
   - **Scatter the error and perform random-sampling.** Now each of the two subsets encodes its string using some high-distance error-correcting code to scatter the error and guarantee that even if the inputs differed in a single position, the output binary strings will differ in at least a constant fraction of bits. Now notice that comparing the two encodings at a random location will detect any difference (in inputs) with at least a constant probability. In particular, for any $\epsilon$, by executing $O(\log(1/\epsilon))$ such checks in parallel, we can reduce the chance of not detecting the $\perp$ to lower than $\epsilon$. That is, we can use uniform randomness to sample random locations from the encoded "interpolated string" and send the sampled bits as an additional leakage from (only) $r$.

The other subset can also use the same uniform randomness to sample at exactly the same locations from the encoded "real string" and can detect any $\perp$ with probability at least $1 - \epsilon$, allowing us the invoke the non-malleability of the underlying non-malleable code.

**Leakage Resilient Non-Malleable Codes.** Unfortunately, all known constructions of 2 split-state statistical non-malleable codes can be trivially broken in our leakage model. All existing constructions use inner-product as a two-source extractor, and further rely on the linearity of inner-product for obtaining an efficient encoding procedure for the non-malleable code. In our leakage-model, the adversary can fully leak one of the sources of the inner-product, and there will be no randomness left to invoke the properties of the two-source extractor. Therefore, in our constructions we need to use some two-source extractor for which the two sources have unequal lengths. While the literature on extractors is very rich, most of the works focus on the improving the parameters of seeded-extractors (weaker objects than two-source extractors) or improving the parameters for multi-source extractors (which generally have sources of equal length). One notable exception is the beautiful construction of Raz [Raz05] which constructs two-source extractors for uneven length sources. Unfortunately, it does not have the linearity property that has been used for efficient encoding. Another possible way of generating two-source extractors is via generalized left-over hash lemma of Lee et al. [LLTT05]. This lemma does not seem to help in obtaining the parameters we seek .

To construct the required leakage-resilient non-malleable code, our starting point would be the non-malleable extractor of Chattopadhyay et al. [CGL16] hereby referred to as the CGL construction. We modify their construction to build the first 2-source non-malleable extractor with uneven length sources. We use the following additional ideas [6]

- **Redesign the advice generator.** The CGL construction is based on alternating extraction from 2 sources (with each source representing one part of the code in case of split state of non-mallaeble code). A crucial step in this construction requires constructing an advice generator by "slicing" from each of the two sources. However now one of the "slices" would need to be much longer than the other. This would lead to an unacceptable blow up in the size of the generated advice. To solve this problem, we redesign the construction of the advice generator. The advice that we generate no longer contains the "slices" explicitly. In our advice generator we use Raz's two-source extractor [Raz05] (instead of inner-product), and rely on its "strongness" (and ideas from the beautiful work of Cohen [Coh16]) for the new analysis.
- **Inject leakage-resilience in each step to obtain overall leakage-resilience.** After the advice is generated, CGL construction can be seen as a sequential invocations of seeded extractors, where the first seed is generated using another two-source extractor. As extractors are naturally leakage-resilient (leakage from the source can be seen as loss of min-entropy from the source, and extractors work with smaller min-entropy as well, as observed in [DP07, DORS08]), we adjust the parameters of linear seeded extractors to handle the leakage, and use Raz's extractor to generate the first seed. In this way we achieve global leakage-resilience by achieving leakage-resilience in each of the intermediate steps.
- **Redesign the encoding procedure.** The encoding procedure of CGL proceeds by randomly sampling all the seeds corresponding to each of the seeded extractors. It then uses the linearity of seeded-extractor to uniformly sample the value of the source given the output and seed . After which, it uses the linearity of inner-product (two-source extractor), to uni-

[6]We give more intuition behind the construction of [CGL16] in section 5.

formly sample two "slices" which encode the first seed. Unfortunately, we do not know how to uniformly sample from the pre-image of the output of Raz's extractor, and this approach of encoding breaks down. Instead of first sampling all the seeds and then the value of sources, we interleave this process. In particular, we randomly sample the value of these two "slices" (which fixes the first and third seed), and carefully sample (adjust) other sources based on fixed output and randomly chosen value of other seeds. At the same time, using properties of Reed-Solomon codes (extending the technique of [CGL16]), we show that everything is consistent with the new advice generator.

**Open problems.** We see this as the beginning of a rich line of research inspired by non-malleable codes. We mention some natural research directions :

- **Handing general access structures.** An interesting future direction is to consider even more general access structures such as those represented by monotone span programs or even monotone polynomial size circuits.
- **More advanced joint-tampering.** An even stronger model of joint tampering would be one where the different subset of shares (being jointly tampered) need not be disjoint. In particular, what if the tampering of any share can depend on any unauthorized set of shares? For the case of t-out-of-n schemes, it allows the tampering of any share to depend on any t-1 (out of n) shares. A construction of even n-out-of-n non-malleable secret sharing scheme in this stronger tampering model would be interesting. Another open problem would be to construct t-out-of-n schemes where the adversary can tamper with two subsets of size exactly $\frac{t}{2}$ [7].
- **Construction with improved rate.** While our focus has not been to optimize the rate of our constructions, we believe it is an interesting problem to improve the rate of non-malleable secrets sharing schemes. As our results use natural primitives in a black-box way, an improvement in the construction of these primitives directly improves the rate of the final secret-sharing scheme. It may also be possible to further improve the rate by designing computational non-malleable secret-sharing schemes.
- **Explore newer applications.** It would be interesting to explore further applications of non-malleable secret sharing. In particular for non-malleable message transmission, an interesting direction is to assume that the network is synchronous and we can leverage the global clock to achieve non-malleability in a much larger class of graphs (and natural networks).

**Paper organization.** We formally define non-malleable codes and secret sharing schemes in section 2. We give our construction for the individual tampering model in section 3. As our main result, we construct non-malleable secret sharing schemes against joint-tampering in section 4. We construct our leakage-resilient non-malleable code in section 5. As an application of our schemes, we introduce non-malleable message transmission in section 6. We construct 2-out-of-n non-malleable (resp. leakage-resilient) secret sharing schemes in appendix A. We also mentions why some simple approaches do not work in appendix B.

**Related Works.** A number of works in the literature ensure that the correct secret is recovered even when some number of shares are arbitrarily corrupted. Concepts from error correcting codes have been useful in obtaining such schemes. McEliece and Sarwate [MS81] noticed that, the polynomial based Shamir's secret sharing scheme [Sha79] is very closely related to Reed-Solomon

[7] We have crucially used unequal sized subsets in our techniques. In more detail, while we needed the larger subset to have full information of $r$, we also crucially relied on $r$ being statistically hidden in the smaller subset

codes, and allows for correct reconstruction of the secret even if less than one third of the shares are altered.

In a seminal work [RBO89], Rabin and Ben-Or introduced verifiable secret sharing, which allowed the adversary to tamper almost half the shares, and still ensured that the adversary cannot cause the reconstruction procedure to output an incorrect message (except with exponentially small error probability). Cramer et al. [CDF$^+$08], in a beautiful work introduced algebraic manipulation detection(AMD) codes and gave almost optimal constructions for them. These codes allow the adversary to "blindly" add any value to the codeword, and ensure that any such algebraic tampering will be detected with high probability. Moreover, they elegantly used these codes to arrive at robust linear secret sharing schemes, that allowed the adversary to arbitrarily tamper with any unauthorized set of shares, with a guarantee that any such tampering will be detected with high probability. As an example, consider the $n$-out-of-$n$ linear secret sharing scheme constructed using AMD codes that allows the adversary to tamper $n-1$ shares arbitrarily, and ensures that any such tampering will be detected with high probability. The notion of non-malleability was introduced in the seminal work of Dolev et al. [DDN91].

## 2 Definitions

We use capital letters to denote distributions and their support, and corresponding small letters to denote a sample from the distribution. Let $[m]$ denote the set $\{1, 2, \ldots, m\}$, and $\mathbf{U_r}$ denote the uniform distribution over $\{0,1\}^r$. For any set $B \in [n]$, let $\otimes_{i \in B} S_i$ denote the Cartesian product $S_{i_1} \times S_{i_2} \times \ldots \times S_{i_{|B|}}$, where $i_1, i_2 \ldots i_{|B|}$ are ordered elements of $B$, such that $i_j < i_{j+1}$.

**Definition 1.** *(**Statistical Distance**) Let $\mathbf{D_1}$ and $\mathbf{D_2}$ be two distributions on a set $S$. The statistical distance between $\mathbf{D_1}$ and $\mathbf{D_2}$ is defined to be :*

$$|\mathbf{D_1} - \mathbf{D_2}| = \max_{T \subseteq S} |\mathbf{D_1}(T) - \mathbf{D_2}(T)| = \frac{1}{2} \sum_{s \in S} |Pr[\mathbf{D_1} = s] - Pr[\mathbf{D_2} = s]|$$

*We say $\mathbf{D_1}$ is $\epsilon$-close to $\mathbf{D_2}$ if $|\mathbf{D_1} - \mathbf{D_2}| \leq \epsilon$. Sometimes we represent the same using $\mathbf{D_1} \approx_\epsilon \mathbf{D_2}$.*

### 2.1 Non-Malleable Codes

**Definition 2.** *(**Coding Schemes**)( [DPW10]). A coding scheme consists of two functions : an encoding function (possibly randomized) $\mathbf{Enc} : \mathcal{M} \to \mathcal{C}$, and a deterministic decoding function $\mathbf{Dec} : \mathcal{C} \to \mathcal{M} \cup \{\perp\}$ such that , for each $m \in \mathcal{M}$, $Pr(\mathbf{Dec}(Enc(m)) = m) = 1$ (over the randomness of the encoding function).*

**Definition 3.** *(**Non-Malleable Codes**) ( [DPW10]). Let $\mathcal{F}$ be some family of tampering functions. For each $\mathbf{f} \in \mathcal{F}$, and $m \in \mathcal{M}$, define the tampering experiment*

$$\mathbf{Tamper_m^f} = \left\{ \begin{array}{c} c \leftarrow \mathbf{Enc}(m) \\ \tilde{c} \leftarrow \mathbf{f}(c) \\ \tilde{m} \leftarrow \mathbf{Dec}(\tilde{c}) \\ Output : \tilde{m} \end{array} \right\}$$

*which is random variable over the randomness of the encoding function $\mathbf{Enc}$. We say a coding scheme $(\mathbf{Enc}, \mathbf{Dec})$ is **$\epsilon$-non-malleable** w.r.t $\mathcal{F}$ if for each $\mathbf{f} \in \mathcal{F}$, there exists a distribution $\mathbf{D^f}$*

*(corresponding to the simulator) over $\mathcal{M} \cup \{same^*, \perp\}$ such that, for all $m \in \mathcal{M}$, we have that the statistical distance between $\mathbf{Tamper_m^f}$ and*

$$\mathbf{Sim_m^f} = \left\{ \begin{array}{c} \tilde{m} \leftarrow \mathbf{D^f} \\ Output : m \text{ } if \tilde{m} = same^*, or \text{ } \tilde{m}, otherwise \end{array} \right\}$$

*is at most $\epsilon$. Additionally, $\mathbf{D^f}$ should be efficiently samplable given oracle access to $\mathbf{f}(.)$.*

## 2.2 Secret Sharing Schemes

The following definitions are inspired from the survey [Bei11].

**Definition 4.** *(**Sharing function** ) Let $[n] = \{1, 2, \ldots, n\}$ be a set of identities of $n$ parties. Let $\mathcal{M}$ be the domain of secrets. A **sharing function** $\mathbf{Share}$ is a randomized mapping from $\mathcal{M}$ to $S_1 \times S_2 \times \ldots \times S_n$, where $S_i$ is called the domain of shares of party with identity $j$. A dealer distributes a secret $m \in \mathcal{M}$ by computing the vector $\mathbf{Share}(m) = (s_1, \ldots, s_n)$, and privately communicating each share $s_j$ to the party $j$. For a set $S \subseteq [n]$, we denote $\mathbf{Share}(m)_S$ to be a restriction of $\mathbf{Share}(m)$ to its $S$ entries.*

**Definition 5.** *($(t, n, \epsilon)$**-Secret Sharing Scheme** [Bei11] ). Let $\mathcal{M}$ be a finite set of secrets, where $|\mathcal{M}| \geq 2$. Let $[n] = \{1, 2, \ldots, n\}$ be a set of identities (indices) of $n$ parties. A sharing function $\mathbf{Share}$ with domain of secrets $\mathcal{M}$ is a $(t, n, \epsilon)$**-Secret Sharing Scheme** if the following two properties hold :*

1. ***Correctness**. The secret can be reconstructed by any $t$-out-of-$n$ parties. That is, for any set $T \subseteq [n]$ such that $|T| \geq t$, there exists a deterministic reconstruction function $\mathbf{Rec} : \otimes_{i \in T} S_i \to \mathcal{M}$ such that for every $m \in \mathcal{M}$,*

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

   *(over the randomness of the Sharing function)*
2. ***Statistical Privacy**. Any collusion of less than $t$ parties should have "almost" no information about the underlying secret. More formally, for any unauthorized set $U \subseteq [n]$ such that $|U| < t$, and for every pair of secrets $a, b \in \mathcal{M}$, for any distinguisher $\mathbf{D}$ with output in $\{0, 1\}$, the following holds :*

$$|Pr_{shares \leftarrow \mathbf{Share}(a)}[\mathbf{D}(shares_U) = 1] - Pr_{shares \leftarrow \mathbf{Share}(b)}[\mathbf{D}(shares_U) = 1]| \leq \epsilon$$

   *The special case of $\epsilon = 0$, is known as Perfect Privacy. We also state $(\mathbf{Share}, \mathbf{Rec})$ as a $(t, n, \epsilon)$-secret sharing scheme.*

**Definition 6.** *(**Non-Malleable Secret Sharing Schemes**) Let $(\mathbf{Share}, \mathbf{Rec})$ be a $(t, n, \epsilon)$-secret sharing scheme for message space $\mathcal{M}$. Let $\mathcal{F}$ be some family of tampering functions. For each $\mathbf{f} \in \mathcal{F}$, $m \in \mathcal{M}$ and authorized set $T \subseteq [n]$ containing $t$ indices, define the tampering experiment*

$$\mathbf{STamper_m^{f,T}} = \left\{ \begin{array}{c} shares \leftarrow \mathbf{Share}(m) \\ \widetilde{shares} \leftarrow \mathbf{f}(shares) \\ \tilde{m} \leftarrow \mathbf{Rec}(\widetilde{shares_T}) \\ Output : \tilde{m} \end{array} \right\}$$

*which is a random variable over the randomness of the sharing function $\mathbf{Share}$. We say that the $(t, n, \epsilon)$-secret sharing scheme, $(\mathbf{Share}, \mathbf{Rec})$ is $\epsilon'$**-non-malleable** w.r.t $\mathcal{F}$ if for each $\mathbf{f} \in \mathcal{F}$*

*and authorized set $T$ containing $t$ indices, there exists a distribution* $\mathbf{SD^{f,T}}$ *(corresponding to the simulator) over* $\mathcal{M} \cup \{same^*, \perp\}$ *such that, for all* $m \in \mathcal{M}$ *, we have that the statistical distance between* $\mathbf{STamper_m^{f,T}}$ *and*

$$\mathbf{SSim_m^{f,T}} = \left\{ \begin{array}{c} \tilde{m} \leftarrow \mathbf{SD^{f,T}} \\ Output : m \text{ } if \tilde{m} = same^*, or \text{ } \tilde{m}, otherwise \end{array} \right\}$$

*is at most* $\epsilon'$*. Additionally,* $\mathbf{SD^{f,T}}$ *should be efficiently samplable given oracle access to* $\mathbf{f}$*(.)*

## 3 Non-Malleable Secret Sharing against Individual Tampering

In this section we show how to convert any $t$-out-of-$n$ secret sharing scheme into a non-malleable one against an adversary who arbitrarily tampers each of the shares independently. We have tried to keep the construction as modular as possible, and have used various components in a black-box way. This approach not only makes the construction more accessible, but also enables one to improve the final construction by improving one or more of the sub-components. We begin with formal definition of the split-state (individual) tampering family:

### Split-State Tampering Family $\mathcal{F}_n^{split}$

Let **Share** be a sharing function that takes as input a message $m \in \mathcal{M}$ and outputs shares $shares \in \otimes_{i\in[n]}\mathcal{S}_i$. Parse the output $shares$ into $n$ blocks, namely $share_1, share_2, \ldots, share_n$ where each $share_i \in \mathcal{S}_i$. For each $i \in [n]$, let $\mathbf{f_i} : \mathcal{S}_i \to \mathcal{S}_i$ be an arbitrary tampering function, that takes as input $share_i$, the $i^{th}$ share. Let $\mathcal{F}_n^{split}$ be a family of such $n$ functions $(\mathbf{f_1}, \mathbf{f_2}, \ldots, \mathbf{f_n})$.

Note that above definition is written with respect to a sharing function. It is just for ease of presentation, we can use this family of tampering functions with respect to a coding scheme, by treating the encoding procedure as a sharing function. We also recall a lemma, which can be used to show that every 2 split-state non-malleable code is a 2-out-of-2 non-malleable secret sharing scheme.

**Lemma 1.** *( [ADKO15, Lemma 6.1]) Let* $\mathbf{Enc} : \mathcal{M} \to \mathcal{C}^2$ *be the encoding function, and* $\mathbf{Dec} : \mathcal{C}^2 \to \mathcal{M} \cup \{\perp\}$ *be a deterministic decoding function. If a coding scheme* $(\mathbf{Enc}, \mathbf{Dec})$ *is* $\epsilon$*-non-malleable w.r.t.* $\mathcal{F}_2^{split}$ *then* $(\mathbf{Enc}, \mathbf{Dec})$ *is also a* $(2, 2\epsilon)$*-secret sharing scheme that is* $\epsilon$*-non-malleable w.r.t.* $\mathcal{F}_2^{split}$*, where* **Enc** *acts as a sharing function.*

**Leakage Family.** As one of our building blocks, we use a 2-out-of-n leakage-resilient secret-sharing scheme, we begin with the formal definition of such schemes and then formally state our split-state leakage family.

**Definition 7.** *(**Leakage-Resilient Secret Sharing Schemes**) Let* $\mathcal{M}$ *be any message space. Let* $\mathcal{L}$ *be some family of leakage functions. We say that the* $(t, n, \epsilon)$*-secret sharing scheme, (***Share***,* **Rec***) is* $\epsilon'$***-leakage-resilient*** *w.r.t.* $\mathcal{L}$ *if for each* $\mathbf{f} \in \mathcal{L}$*, and for any two messages* $a, b \in \mathcal{M}$*, any distinguisher* $\mathbf{D}$ *with output in* $\{0, 1\}$*, the following holds :*

$$|Pr_{shares \leftarrow \mathbf{Share}(a)}[\mathbf{D}(f(shares)) = 1] - Pr_{shares \leftarrow \mathbf{Share}(b)}[\mathbf{D}(f(shares)) = 1]| \leq \epsilon'$$

**Leakage Family $\mathcal{L}_\mu^{split}$**

$\mathcal{L}_\mu^{split}$consists of the family of all leakage functions of the form $\mathbf{f} = (\mathbf{f_1}, \mathbf{f_2}, \ldots, \mathbf{f_n})$ and any index $j \in [n]$. Specifically, for the chosen $j \in [n]$, the function $\mathbf{f_j}$ on input $share_j$ outputs the whole $share_j$. While for each $i \in [n] \setminus \{j\}$, function $\mathbf{f_i}$ computes an arbitrary function of input $share_i$ and outputs at most $\mu$ bits.

**Main result for individual tampering.**

**Theorem 1.** *For any message space $\mathcal{M}$, any threshold $t \geq 2$, any number of parties $n \geq t$, if we have the following primitives :*

1. *For any $\epsilon_1 \geq 0$, let $(\mathbf{NMEnc}, \mathbf{NMDec})$ be any coding scheme that is $\epsilon_1$-non-malleable w.r.t. $\mathcal{F}_2^{split}$, which encodes an element of $\mathcal{M}$ into two elements of $\mathbb{F}_1$.*
2. *For any $\epsilon_2 \geq 0$, let $(\mathbf{TShare_n^t}, \mathbf{TRec_n^t})$ be any $(t, n, \epsilon_2)$-secret sharing scheme that shares an element of $\mathbb{F}_1$ into $n$ elements of $\mathbb{F}_2$.*
3. *Let $\mu \leftarrow \log|\mathbb{F}_2|$. For any $\epsilon_3 \geq 0$, let $(\mathbf{LRShare_n^2}, \mathbf{LRRec_n^2})$, be any $(2, n, \epsilon_3)$-secret sharing scheme that is $\epsilon_3$-leakage-resilient w.r.t. $\mathcal{L}_\mu^{split}$, that shares an element of $\mathbb{F}_1$ into $n$ elements of $\mathbb{F}_3$.*

*then there exists $(t, n, 2\epsilon_1 + \epsilon_2)$-secret sharing scheme that is $(2\epsilon_1 + \epsilon_2 + \epsilon_3)$-non-malleable w.r.t. $\mathcal{F}_n^{split}$. The resulting scheme, $(\mathbf{NMShare_n^t}, \mathbf{NMRec_n^t})$, shares an element of $\mathcal{M}$ into $n$ shares where each share is an element of $(\mathbb{F}_2 \times \mathbb{F}_3)$. Furthermore, if the three primitives have efficient construction, then the constructed scheme is also efficient.*

*Proof.* In case threshold $t$ is equal to two, we use the scheme constructed in theorem 7 in appendix A. Otherwise, we give our construction of $(\mathbf{NMShare_n^t}, \mathbf{NMRec_n^t})$ :

1. **Sharing Function**$(\mathbf{NMShare_n^t})$
   On input a secret $m \in \mathcal{M}$, encode $m$ using the encoding procedure of the non-malleable code. Let $l, r \leftarrow \mathbf{NMEnc}(m)$. Use the sharing procedure $\mathbf{TShare_n^t}$ to share $l$. Let $((l_1, \ldots, l_n) \leftarrow \mathbf{TShare_n^t}(l)$. Use the sharing function of the leakage-resilient secret sharing scheme to share $r$. Let $(r_1, \ldots, r_n) \leftarrow \mathbf{LRShare_n^2}(r)$. For each $i \in [n]$, construct the $i^{th}$ share of our scheme as follows : $share_i = (l_i, r_i)$. Output $(share_1, \ldots, share_n)$

2. **Reconstruction Function**$(\mathbf{NMRec_n^t})$
   Without loss of generality assume that the authorized set $T$ has exactly $t$ elements, as we only use the first $t$ elements of $T$ and can ignore all other shares. Let $\{i_1, i_2, \ldots, i_t\}$ be ordered indices of $T$ such that $i_j < i_{j+1}$ for each $j \in [t-1]$. On input the shares $\otimes_{i \in T} share_i$, for each $i \in T$, parse $share_i$ as $(l_i, r_i)$. Run the reconstruction procedure $\mathbf{TRec_n^t}$ on $t$ shares of $l$, to obtain $l \leftarrow \mathbf{TRec_n^t}(\otimes_{i \in T} l_i)$. Run the reconstruction procedure of the leakage-resilient secret sharing scheme on the first two shares of $r$, to obtain $r \leftarrow \mathbf{LRRec_n^2}(r_{i_1}, r_{i_2})$. Decode $l$ and $r$ using decoding process of underlying non-malleable code to obtain : $m \leftarrow \mathbf{NMDec}(l, r)$. Output $m$.

Let us show that $(\mathbf{NMShare_n^t}, \mathbf{NMRec_n^t})$ as constructed above satisfies the three properties required for a non-malleable secret sharing scheme :

**Correctness and Efficiency** : Trivially follows from the construction.

**Statistical Privacy** : We prove statistical privacy using hybrid argument. For ease of understanding, let $share_i$ be of the form $al_i, ar_i$ when the secret $a$ is encoded by the sharing procedure

$\mathbf{NMShare_n^t}$. Similarly, let $share_i$ be of the form $bl_i, br_i$ when the secret $b$ is encoded. Let $U$ be an unauthorized set containing less than $t$ indices. We describe the hybrids below :

1. $\mathbf{Hybrid_1}$ : for each $i \in U$, $share_i$ is of the form $al_i, ar_i$. The distribution of these shares is identical to distribution obtained on running the $\mathbf{NMShare_n^t}$ on input $a$. Output $\otimes_{i \in U} share_i$.
2. $\mathbf{Hybrid_2}$ : Sample the shares as in $\mathbf{Hybrid_1}$, the previous hybrid. For each $i \in U$, replace $al_i$ with $bl_i$ to obtain share of the form $bl_i, ar_i$. Output $\otimes_{i \in U} share_i$.
3. $\mathbf{Hybrid_3}$ : Sample the shares as in $\mathbf{Hybrid_2}$, the previous hybrid. For each $i \in U$, replace $ar_i$ with $br_i$ to obtain share of the form $bl_i, br_i$. The distribution of these shares is identical to distribution obtained on running the $\mathbf{NMShare_n^t}$ on input $b$. Output $\otimes_{i \in U} share_i$.

Claim: For any pair of secrets $a, b \in \mathcal{M}$, any unauthorized set $U$, the statistical distance between $\mathbf{Hybrid_1}$ and $\mathbf{Hybrid_2}$ is at most $\epsilon_2$.
Proof: The two hybrids only differ in the shares of $l$. The claim follows from the statistical privacy of the scheme ($\mathbf{TShare_n^t}, \mathbf{TRec_n^t}$). That is for any $c, d \in \mathbb{F}_1$, any $U$ containing less than $t$ indices, any distinguisher $\mathbf{D}$ with output $\in \{0, 1\}$,

$$|Pr_{sh \leftarrow \mathbf{TShare_n^t}(c)}[\mathbf{D}(sh_U) = 1] - Pr_{sh \leftarrow \mathbf{TShare_n^t}(d)}[\mathbf{D}(sh_U) = 1]| \leq \epsilon_2$$

■

Claim: For any pair of secrets $a, b \in \mathcal{M}$, any unauthorized $U$, the statistical distance between $\mathbf{Hybrid_2}$ and $\mathbf{Hybrid_3}$ is at most $2\epsilon_1$.
Proof: Assume towards contradiction that there exists $a, b \in \mathcal{M}$, an unauthorized set $U$, and a distinguisher $\mathbf{D}$ that is successful in distinguishing the hybrids with probability greater than $2\epsilon_1$. We use distinguisher $\mathbf{D}$ to construct another distinguisher $\mathbf{D_1}$ which violates the property of statistical privacy satisfied by the underlying non-malleable code for the message pair $a$ and $b$, as proved in Lemma 1.

The distinguisher $\mathbf{D_1}$ is defined as follows : On input a share $r$, run the sharing function of the leakage-resilient scheme to obtain $r_1, r_2, \ldots, r_n \leftarrow \mathbf{LRShare_n^2}(r)$. Sample $\otimes_{i \in U} share_i$ according to $\mathbf{Hybrid_2}$. For each $i \in U$, replace $ar_i$ with $r_i$ in $share_i$ to obtain a share of form $al_i, r_i, bp_i$. Invoke the distinguisher $\mathbf{D}$ with the modified $\otimes_{i \in U} share_i$ and output its output.

Notice, in the case $r$ was sampled while encoding $a$, then $\mathbf{D}$ will be invoked with input distributed according to $\mathbf{Hybrid_2}$. Otherwise, it will be invoked with distributed according to $\mathbf{Hybrid_3}$. Therefore the success probability of $\mathbf{D_1}$ will be equal to the advantage of $\mathbf{D}$ in distinguishing these two hybrids, which is greater than $2\epsilon_1$. We have arrived at a contradiction to $(2, 2\epsilon_1)$-secret sharing of ($\mathbf{NMEnc}, \mathbf{NMDec}$).

■

By repeated application of triangle inequality, we get that for any $a, b \in \mathcal{M}$, any unauthorized $U$ containing less than $t$ shares, the statistical distance between $\mathbf{Hybrid_1}$ and $\mathbf{Hybrid_3}$ is at most $2\epsilon_1 + \epsilon_2$. This proves the statistical privacy of our scheme.

**Statistical Non Malleability** : To prove non malleability of the current secret sharing scheme, we need to give a simulator for every admissible tampering attack on the scheme. To this end, we transform an admissible tampering attack on the current scheme into an equivalent split-state attack on the underlying non-malleable code. We know by the properties of the non-malleable code, that for every split-state tampering attack there exists a simulator whose output distribution is similar to the one obtained in real tampering experiment. We use the simulator of the underlying non-malleable code as the simulator of our scheme, and the non malleability of our scheme follows from the equivalence of the tampering attacks. Now we present the formal proof.

As the adversary belongs to $\mathcal{F}_n^{split}$, it specifies a set of $n$ tampering functions $\{\mathbf{f_i} : i \in [n]\}$. All these functions act on their respective shares independently of the other shares, i.e. every $\mathbf{f_i}$ takes $share_i$ as input and outputs the tampered $\widetilde{share_i}$. We can also assume without loss of generality that all these tampering functions are deterministic, as the computationally unbounded adversary can compute the optimal randomness. Further, we can also assume that adversary chooses an authorized set $T$ that contains exactly $t$ indices, as we will only use the first $t$ shares for reconstruction. Let $T = \{i_1, i_2 \dots i_t\}$ be an ordered set of $t$ indices, such that $i_j < i_{j+1}$. We now give the formal reduction, in which we use the tampering functions $\{\mathbf{f_i} : i \in T\}$ that tamper with the shares of our scheme $(\mathbf{NMShare_n^t}, \mathbf{NMRec_n^t})$ to create explicit tampering functions $\mathbf{F}$ and $\mathbf{G}$ that tamper with the two parts of non-malleable code. Recall that as $\mathcal{F}_2^{split}$ allows arbitrary computation, the functions $\mathbf{F}$ and $\mathbf{G}$ are allowed to brute force over any finite subset.

1. **(Initial Setup)** : Fix $l_\$$ and $r_\$$ encoding any $m_\$ \leftarrow \mathbf{NMDec}(l_\$, r_\$)$. Run the sharing function $\mathbf{TShare_n^t}$ with input $l_\$$ to obtain $\otimes_{i\in[n]} tl_i$. Run the sharing function $\mathbf{LRShare_n^2}(r_\$)$ to obtain $\otimes_{i\in[n]} tr_i$. For each $i \in [n]$, create $tShare_i$ as $tl_i, tr_i$. For each $i \in T \setminus \{i_t\}$, run $\mathbf{f_i}$ on $tShare_i$ to obtain $\widetilde{tShare_i} \leftarrow \mathbf{f_i}(tShare_i)$. Parse $\widetilde{tShare_i}$ as $\widetilde{tl_i}, \widetilde{tr_i}$. **Fix** $l_i \leftarrow tl_i$ and $\widetilde{l_i} \leftarrow \widetilde{tl_i}$. For the last share corresponding to index $i_t$, **fix** $r_{i_t} \leftarrow tr_{i_t}$.

2. The **tampering function** $\mathbf{F}(l)$ is defined as follows : On input $l$, sample the value of $l_{i_t}$ such that the shares $\{l_i : i \in T\}$ hide the secret $l$ and the distribution of sampled $l_{i_t}$ is identical to the distribution of $l_{i_t}$ produced on running $\mathbf{TShare_n^t}$ with input $l$ conditioned on fixing $\{l_i : i \in T \setminus \{i_t\}\}$. In case such a sampling is not possible, then `abort`. Otherwise, construct $share_{i_t}$ as $l_{i_t}, r_{i_t}$ using the fixed value of $r_{i_t}$. Run the tampering function $\mathbf{f_{i_t}}$ on $share_{i_t}$ to obtain tampered $\widetilde{share_{i_t}}$. Parse $\widetilde{share_{i_t}}$ as $\widetilde{l_{i_t}}, \widetilde{r_{i_t}}$. Run the reconstruction function $\mathbf{TRec_n^t}$ with input $\otimes_{i\in T}\widetilde{l_i}$ to obtain $\widetilde{l}$. Output $\widetilde{l}$.

3. The **tampering function** $\mathbf{G}(r)$ is defined as follows : On input $r$, sample the values of $t-1$ shares of $r$, namely $\{r_i : i \in T \setminus \{i_t\}\}$ satisfying the following constraints :-
   - The $t$ shares, namely, $\{r_i : i \in T\}$ should be distributed according to the distribution of output of $\mathbf{LRShare_n^2}$ on input $r$. In other words, these shares should correspond to the $t$ shares of the scheme $(\mathbf{LRShare_n^2}, \mathbf{LRRec_n^2})$, in which any two of these shares can be used to reconstruct $r$.
   - For each $i \in T \setminus \{i_t\}$, let $share_i$ be $l_i, r_i$, run $\mathbf{f_i}$ on $share_i$ to obtain $\widetilde{share_i}$. Parse $\widetilde{share_i}$ as $\widetilde{nl_i}, \widetilde{nr_i}$. The value of $\widetilde{nl_i}$ should be equal to $\widetilde{l_i}$ (the value that was fixed in the initial step of reduction).

   In case such a sampling is not possible, then `abort`. Otherwise, run the reconstruction procedure of the leakage-resilient scheme to obtain $\widetilde{r}$, using the tampered values of first 2 shares of $r$. That is $\widetilde{r} \leftarrow \mathbf{LRRec_n^2}^{\{i_1,i_2\}}(\widetilde{nr_{i_1}}, \widetilde{nr_{i_2}})$. Output $\widetilde{r}$.

The reduction given above creates $t$ shares corresponding to indices in $T$. To ensure the sanity of $t$ tampering functions, we need to prove that for a given $l$ and $r$, the distribution of the $t$ shares created by the reduction is statistically close to the distribution of $t$ shares which are obtained on running $\mathbf{NMShare_n^t}$ conditioned on output of $\mathbf{NMEnc}(m)$ being $l, r$. Otherwise, the tampering functions may detect a change in distribution and stop working.

We achieve this using a hybrid argument described below. For ease of understanding, let $share_i$ be of the form $al_i, ar_i$ when the shares are produced by the reduction on input $l$ and $r$, with the fixing of $l_\$$ and $r_\$$. Similarly, let $share_i$ be of the form $bl_i, br_i$ when the secret $m$ is encoded by the sharing procedure $\mathbf{NMShare^t_n}$ conditioned on output of $\mathbf{NMEnc}(m)$ being $l, r$.

1. $\mathbf{Hybrid_1}$ : for each $i \in T$, $share_i$ is of the form $al_i, ar_i$. The distribution of these $t$ shares is identical to distribution of the shares produced by the reduction on input $l$ and $r$, with the fixing of $l_\$$ and $r_\$$. Output $\otimes_{i\in T} share_i$.
2. $\mathbf{Hybrid_2}$ : Fix $l_\$ \leftarrow l$ in the initial setup phase. Proceed with the reduction to create the $t$ shares. Output $\otimes_{i\in T} share_i$.
3. $\mathbf{Hybrid_3}$ : Fix $l_\$ \leftarrow l$ and fix $r_\$ \leftarrow r$ in the initial setup phase. Proceed with the reduction to create the $t$ shares. Output $\otimes_{i\in T} share_i$.
4. $\mathbf{Hybrid_4}$ : For each $i \in [n]$, let $share_i$ be of the form $bl_i, br_i$. The distribution of these $t$ shares is identical to distribution obtained on running the $\mathbf{NMShare^t_n}$ conditioned on output of $\mathbf{NMEnc}(m)$ being $l, r$. Output $\otimes_{i\in T} share_i$.

Claim: For any $l, l_\$$, any authorized set $T$ containing $t$ indices, the statistical distance between $\mathbf{Hybrid_1}$ and $\mathbf{Hybrid_2}$ is at most $\epsilon_2$.
Proof: The two hybrids differ in the initial setup phase. In $\mathbf{Hybrid_1}$, $t-1$ shares of $l_\$$ are fixed, while in $\mathbf{Hybrid_2}$ $t-1$ shares of $l$ are fixed. By the statistical secrecy of $\mathbf{TShare^t_n}$, for any $l, l_\$ \in \mathbb{F}_1$, any $T \setminus \{i_t\}$, any distinguisher $\mathbf{D}$ with binary output, we know that

$$|Pr_{sh\leftarrow \mathbf{TShare^t_n}(l)}[\mathbf{D}(sh_{T\setminus\{i_t\}}) = 1] - Pr_{sh\leftarrow \mathbf{TShare^t_n}(l_\$)}[\mathbf{D}(sh_{T\setminus\{i_t\}}) = 1]| \leq \epsilon_2$$

Now, assume towards contradiction that the $t$ shares created by our reduction in the two hybrids are not statistically close, and there exists a $l, l_\$$ and there exists a distinguisher $\mathbf{D}$ which differentiates the two distributions with probability greater than $\epsilon_2$. In this case, we create a distinguisher $\mathbf{D_1}$ which violates the statistical privacy property of $\mathbf{TShare^t_n}$ mentioned above. More formally, given a set of $t-1$ shares of $l$ or $l_\$$, namely $l_{i_1}, \dots, l_{i_{t-1}}$, the new distinguisher $\mathbf{D_1}$ uses these $l_{i_1}, \dots, l_{i_{t-1}}$, instead of running $\mathbf{TShare^t_n}$ on $l_\$$ in the initial setup phase and proceeds in a manner similar to $\mathbf{Hybrid_1}$. Then it invokes the distinguisher $\mathbf{D}$ with the sampled shares and outputs it output. Notice, in case $l_{i_1}, \dots, l_{i_{t-1}}$ encoded the message $l_\$$, we are in a distribution identical to $\mathbf{Hybrid_1}$. Otherwise, if they encoded the secret $l$, we are in $\mathbf{Hybrid_2}$. Therefore, the distinguishing advantanges are of both the distinguishers is the same, and we have arrived at the contradiction of statistical privacy of $(t, n, \epsilon_2)$-secret sharing scheme $(\mathbf{TShare^t_n}, \mathbf{TRec^t_n})$. ■

The above also shows that the function $\mathbf{F}$ in the reduction aborts with probability less than $\epsilon_2$.
Claim: For any $r, r_\$$, any authorized set $T$ containing $t$ indices, the statistical distance between $\mathbf{Hybrid_2}$ and $\mathbf{Hybrid_3}$ is at most $\epsilon_3$.
Proof: Assume towards contradiction that there exists $r, r_\$ \in \mathbb{F}_1$, and a distinguisher $\mathbf{D}$ that is successful in distinguishing $\mathbf{Hybrid_2}$ and $\mathbf{Hybrid_3}$ with probability greater than $\epsilon_3$. We use distinguisher $\mathbf{D}$ to construct another distinguisher $\mathbf{D_1}$ and a leak function $\mathbf{g} \in \mathcal{L}_\mu^{split}$ which violates the property of leakage-resilience satisfied by the scheme $(\mathbf{LRShare^2_n}, \mathbf{LRRec^2_n})$ for the secrets $r, r_\$$. The reduction is described below :

1. (**Initial Setup**) : Run the sharing function $\mathbf{TShare^t_n}$ with input $l$ to obtain $\otimes_{i\in[n]} tl_i$. For each $i \in T \setminus \{i_t\}$, **fix** $l_i \leftarrow tl_i$.

2. (**Leak function g**) : We define a specific leakage function $\mathbf{g} = (\mathbf{g_1}, \dots, \mathbf{g_n})$ which leaks independently from the $n$ shares.
   - For each $i \in [n] \setminus T$, define $\mathbf{g_i}$ as a vacuous function that takes as input $r_i$ and outputs 0 bits.
   - Define $\mathbf{g_{i_t}}$ as identity function, it outputs the whole $r_{i_t}$.

- For each $i \in T \setminus \{i_t\}$, define $\mathbf{g_i}$ as the following function. Create $tShare_i$ as $l_i, r_i$. Run $\mathbf{f_i}$ on $tShare_i$ to obtain $\widetilde{tShare_i} \leftarrow \mathbf{f_i}(tShare_i)$. Parse $\widetilde{tShare_i}$ as $\widetilde{tl_i}, \widetilde{tr_i}$. Output $\widetilde{tl_i}$.

As $\widetilde{tl_i}$ is an element of $\mathbb{F}_2$, it can be represented by at most $\log |\mathbb{F}_2|$ bits, which is equal to $\mu$. This shows that the above leak function $\mathbf{g}$ belongs to the class $\mathcal{L}_{\mu}^{split}$ which allows one share to be fully leaked and $\mu$ bits of leakage from each of the other shares.

3. (**Distinguisher $\mathbf{D_1}$**) : The distinguisher $\mathbf{D_1}$ is defined as follows : On input $\mathbf{g}(r_1, \ldots, r_n)$, parse it as $\widetilde{tl_{i_1}}, \widetilde{tl_{i_2}}, \ldots, \widetilde{tl_{i_{t-1}}}, r_{i_t}$. For each $i \in T \setminus \{i_t\}$, **fix** $\widetilde{l_i} \leftarrow \widetilde{tl_i}$. For the last share corresponding to index $i_t$, **fix** $r_{i_t}$. This completes the initial setup of our original reduction. Now invoke $\mathbf{F}$ with input $l$ and $\mathbf{G}$ with input $r$ to create the set of $t$ shares, namely $\{share_i : i \in T\}$. Invoke the distinguisher $\mathbf{D}$ with these $t$ shares and output its output.

Notice, in the case the secret hidden by the leakage-resilient scheme was $r_\$$, $\mathbf{D}$ will be invoked with input distributed according to $\mathbf{Hybrid_2}$. In the other case, in which $r$ was hidden, $\mathbf{D}$ will be invoked with distributed according to $\mathbf{Hybrid_3}$. Therefore the success probability of $\mathbf{D_1}$ will be equal to the advantage of $\mathbf{D}$ in distinguishing these two hybrids, which is greater than $\epsilon_3$ by assumption. Hence, we have arrived at a contradiction to statistical leakage-resilience property of the scheme $(\mathbf{LRShare_n^2}, \mathbf{LRRec_n^2})$. ■

The above also shows that the function $\mathbf{G}$ in the reduction aborts with probability less than $\epsilon_3$.
Claim: For any $l, r$, $\mathbf{Hybrid_3}$ is identical to $\mathbf{Hybrid_4}$.
Proof: In $\mathbf{Hybrid_3}$, the shares of $r_\$$ that are sampled in the initial setup already encode the value $r$ (as $r_\$ = r$). Recall that these shares are used for fixing the tampered shares of $l$, but now as the unfixed shares of $r_\$$ are already from the correct distribution, the tampering function $\mathbf{G}(r)$ can trivially use the same shares, and all the leakages (deterministic function computing tampered shares of $\tilde{l}$) will be trivially satisfied. Similarly, $\mathbf{F}(l)$ can use the value of unfixed share $tl_{i_t}$, because the $t$ shares of $l$ already encode $l$ ( as $l_\$ = l$ in $\mathbf{Hybrid_3}$). (Basically, $(\mathbf{F}, \mathbf{G})$ will be always successful in finding such satisfying shares by brute-force.) Therefore, all the $t$ shares created in $\mathbf{Hybrid_3}$ will be identically distributed to the ones produced while executing $\mathbf{NMShare_n^t}$ with the output of $\mathbf{NMEnc}$ being $(l, r)$. ■

By repeated application of triangle inequality, we get that for any $a, b \in \mathcal{M}$, the statistical distance between $\mathbf{Hybrid_1}$ and $\mathbf{Hybrid_4}$ is at most $\epsilon_2 + \epsilon_3$. This proves that the set of shares created by our reduction is statistically close the set of shares created during the real sharing by the scheme, and thus the tampering functions $\mathbf{f} = \{\mathbf{f_i} : i \in T\}$ can be successfully invoked.

From our construction of $\mathbf{F}$ and $\mathbf{G}$, it is clear that for any $l$ and $r$, if the reduction is successful in creating the $t$ shares, then the secret hidden is these $t$ shares is the same as the message encoded by $l$ and $r$ (under non-malleable code). That is,

$$\mathbf{NMRec_n^t}(\{share_i : i \in T\}) = \mathbf{NMDec}(l, r)$$

Similarly, we can say that the secret hidden is the $t$ tampered shares is the same as the message encoded by tampered $\tilde{l}$ and tampered $\tilde{r}$. That is,

$$\mathbf{NMRec_n^t}(\{\mathbf{f_i}(share_i) : i \in T\}) = \mathbf{NMDec}(\mathbf{F}(l), \mathbf{G}(r))$$

Therefore, the tampering experiments of non-malleable codes (see definition 3) and non-malleable secret-sharing schemes (see definition 6) are statistically indistinguishable, specifically,

$$\mathbf{STamper_m^{f,T}} \approx_{\epsilon_2+\epsilon_3} \mathbf{Tamper_m^{F,G}}$$

By the $\epsilon_1$-non malleability of the scheme $(\mathbf{NMEnc}, \mathbf{NMDec})$, we know that there exists a distribution $\mathbf{D}^{\mathbf{F},\mathbf{G}}$ such that

$$\mathbf{Tamper}_{\mathbf{m}}^{\mathbf{F},\mathbf{G}} \approx_{\epsilon_1} \mathbf{Sim}_{\mathbf{m}}^{\mathbf{F},\mathbf{G}}$$

Using the underlying distribution $\mathbf{D}^{\mathbf{F},\mathbf{G}}$ as our distribution $\mathbf{SD}^{\mathbf{f},\mathbf{T}}$ we get our simulator.

$$\mathbf{Sim}_{\mathbf{m}}^{\mathbf{F},\mathbf{G}} \equiv \mathbf{SSim}_{\mathbf{m}}^{\mathbf{f},\mathbf{T}}$$

Applying triangle inequality to the above relations we prove the statistical non malleability of our scheme.

$$\mathbf{STamper}_{\mathbf{m}}^{\mathbf{f},\mathbf{T}} \approx_{\epsilon_1+\epsilon_2+\epsilon_3} \mathbf{SSim}_{\mathbf{m}}^{\mathbf{f},\mathbf{T}}$$

□

**Corollary 1.** *For any threshold $t \geq 2$, any number of parties $n \geq t$, any $\epsilon > 0$, there exists an efficient $(t, n, \epsilon)$-secret sharing scheme that is $\epsilon$-non-malleable w.r.t. $\mathcal{F}_n^{split}$. The sharing function of the constructed scheme shares a m bit secret into n shares such that the length of each share is $O(nm \log m)$ bits with error $\epsilon = n2^{-\Omega(m)}$.*

*Proof.* If threshold $t$ is greater than two, we instantiate the primitives in theorem 1 using the following constructions :

1. We use split-state non-malleable code constructed by Li [Li17] $(\mathbf{LiNMEnc}, \mathbf{LiNMDec})$. **LiEnc** encodes a $m$ bit message to two shares each of length $k \leftarrow O(m \log m)$ bits and has error $\epsilon_1 = 2^{-\Omega(m)}$. We set $\epsilon_1 = \frac{\epsilon}{2}$.
2. We use the t-out-of-n Shamir's secret sharing scheme [Sha79] $(\mathbf{TShare}_{\mathbf{n}}^{\mathbf{t}}, \mathbf{TRec}_{\mathbf{n}}^{\mathbf{t}})$. The sharing function $\mathbf{TShare}_{\mathbf{n}}^{\mathbf{t}}$ shares a $k$ bit secret into $n$ shares each of size $k$ bits.
3. We use the 2-out-of-n split-state leakage-resilient secret sharing scheme created in appendix A. Specifically, using lemma 14, and leakage-parameter $\mu \leftarrow k$, we get an efficient construction of a 2-out-of-2 secret sharing scheme that $\frac{\epsilon_3}{n^2}$-leakage-resilient w.r.t. $\mathcal{L}_{\mu}^{split}$. We use this scheme in theorem 6, to construct our 2-out-of-n secret sharing scheme that is $\epsilon_3$-leakage-reslient w.r.t. to $\mathcal{L}_{\mu}^{split}$ and encodes a $k$ bit secret into $n$ shares, where each share has length $O(nk)$. We set $\epsilon_3 = \frac{\epsilon}{2}$

In case threshold $t$ is equal to two, we use the scheme constructed in theorem 7 in appendix A with the split-state non-malleable code constructed by Li [Li17] $(\mathbf{LiNMEnc}, \mathbf{LiNMDec})$ to arrive at our t-out-of-n non-malleable secret-sharing schemes. □

## 4 Non-Malleable Secret Sharing against Joint Tampering

While our previous constructions allowed the adversary to tamper each share independently, in this section, we relax this requirement and construct t-out-of-n non-malleable secret sharing sharing schemes that allow the tampering of each share to depend on the values of multiple shares. In more detail, we want our non-malleable secret-sharing schemes to handle the following class of tampering functions.

### Joint Tampering Family $\mathcal{F}_{t,n}^{joint}$

For any t-out-of-n secret sharing scheme, the adversary chooses any t-out-of-n shares to obtain an authorized set $T$, partitions the set $T$ into any two non-empty subsets which have different cardinalities [8], and jointly tampers with shares in each of the two subsets arbitrarily and independently.

To construct such schemes we make black-box use of non-malleable codes that have a strong leakage-resilience property. We formally define the property that we need below, and construct non-malleable codes satisfying it in section 5 [9].

**Definition 8.** *(**Leakage-Resilient Non-Malleable Codes**) Let $\mathcal{M}$ be any message space. Let ($\mathbf{Enc}, \mathbf{Dec}$) be any coding scheme, such that $\mathbf{Enc} : \mathcal{M} \to \mathcal{L} \times \mathcal{R}$ encodes any message into two parts. Let $\mathbf{Leak} : \mathcal{R} \to \{0,1\}^\mu$ be any leakage function that outputs $\mu$ bits of arbitrary information about its input. Similarly, let $\mathbf{F} : \mathcal{L} \times \{0,1\}^\mu \to \mathcal{L}$ and $\mathbf{G} : \mathcal{R} \to \mathcal{R}$ be any two tampering functions that arbitrarily uses its input to output the tampered parts (Without loss of generality, we can assume that all these functions specified by the adversary are deterministic, as the adversary being computationally unbounded can compute the optimal randomness). For any message $m \in \mathcal{M}$, we define*

$$\mathbf{LTamper}_{\mathbf{m}}^{\mathbf{F},\mathbf{G},\mathbf{Leak}} = \left\{ \begin{array}{c} l, r \leftarrow \mathbf{Enc}(m) \\ a \leftarrow \mathbf{Leak}(r) \\ \widetilde{l} \leftarrow \mathbf{F}(l, a) \\ \widetilde{r} \leftarrow \mathbf{G}(r) \\ \widetilde{m} \leftarrow \mathbf{Dec}(\widetilde{l}, \widetilde{r}) \\ Output : \widetilde{m} \end{array} \right\}$$

*which is random variable over the randomness of the encoding function $\mathbf{Enc}$. We say a coding scheme ($\mathbf{Enc}, \mathbf{Dec}$) is $\mu$-**leakage resilient** $\epsilon$-**non-malleable** if for any such $\mathbf{F}, \mathbf{G}, \mathbf{Leak}$, there exists a distribution $\mathbf{D}^{\mathbf{F},\mathbf{G},\mathbf{Leak}}$ (corresponding to the simulator) over $\mathcal{M} \cup \{same^*, \perp\}$ such that, for all $m \in \mathcal{M}$, we have that the statistical distance between $\mathbf{LTamper}_{\mathbf{m}}^{\mathbf{F},\mathbf{G},\mathbf{Leak}}$ and*

$$\mathbf{LSim}_{\mathbf{m}}^{\mathbf{F},\mathbf{G},\mathbf{Leak}} = \left\{ \begin{array}{c} \widetilde{m} \leftarrow \mathbf{D}^{\mathbf{F},\mathbf{G},\mathbf{Leak}} \\ Output : m \text{ if } \widetilde{m} = same^*, \text{or } \widetilde{m}, otherwise \end{array} \right\}$$

*is at most $\epsilon$. Additionally, $\mathbf{D}^{\mathbf{F},\mathbf{G},\mathbf{Leak}}$ should be efficiently samplable given oracle access to $\mathbf{F}, \mathbf{G}$, $\mathbf{Leak}$.*

We now give the main result of this section.

**Theorem 2.** *Assume that for any message space $\mathcal{M}$, any threshold $t \geq 2$ , any number of parties $n \geq t$, any $\epsilon > 0$, there exists an efficient coding scheme (**NMEnc**,**NMDec**) that encodes a message $m \in \mathcal{M}$ into two shares in $\mathcal{L} \times \mathcal{R}$ and is $(\log(|\mathcal{L}|) + \log(\frac{1}{\epsilon}))$-leakage resilient $\epsilon$-non-malleable (as defined in def 8).*

*Then there exists an efficient $(t, n, 2\epsilon)$-secret sharing scheme that is $2\epsilon$-non-malleable w.r.t $\mathcal{F}_{t,n}^{joint}$. The resulting scheme, ($\mathbf{JNMShare}_{\mathbf{n}}^{\mathbf{t}}$, $\mathbf{JNMRec}_{\mathbf{n}}^{\mathbf{t}}$), shares an element of $\mathcal{M}$ into $n$ shares, where each share is an element of $\mathcal{L} \times \mathcal{R}$.*

[8]We need the subsets to have different number of shares for our techniques to work. It is easy to see that if the threshold $t$ is odd, or number of subsets is more than two, then such a requirement is trivially satisfied.

[9]More general definition of multi-round leakage-resilient non-malleable codes can be found in the beautiful work of Aggarwal et al. [ADKO15]. Unfortunately, we cannot use their codes because of our leakage parameters. While it is possible to extend the number of rounds of leakage in our constructions (and definitions) (using known techniques of [DP07, DP08, ADKO15]), for the sake of clarity of exposition, we restrict our model to only include one-round of leakage, as it suffices for arriving at goal of achieving joint-tampering.

*Proof.* The case of threshold $t = 2$ follows from the previous theorem 1. Otherwise, we begin with defining our building blocks :

- Let ($\mathbf{TShare_n^t}$,$\mathbf{TRec_n^t}$) denote $t$-out-of-$n$ Shamir's secret sharing scheme [Sha79] that shares an element of field $\mathcal{L}$ into $n$ elements of $\mathcal{L}$ ( We can choose our non-malleable code such that $\mathcal{L}$ and $\mathcal{R}$ are fields and $|\mathcal{L}| > n$ as Shamir's secret-sharing scheme needs it).
- Let $k \leftarrow 1 + \lfloor t/2 \rfloor$. Let ($\mathbf{TShare_n^k}$,$\mathbf{TRec_n^k}$) denote $k$-out-of-$n$ Shamir's secret sharing scheme that shares an element of field $\mathcal{R}$ into $n$ elements of $\mathcal{R}$ .

The construction of our scheme is given below :

1. **Sharing Function**($\mathbf{JNMShare_n^t}$)
   On input a secret $m \in \mathcal{M}$, encode $m$ using the encoding procedure of the non-malleable code. Let $l, r \leftarrow \mathbf{NMEnc}(m)$. Use the sharing function of the $t$-out-of-$n$ threshold secret sharing scheme to share $l$. Let $(l_1, \ldots, l_n) \leftarrow \mathbf{TShare_n^t(l)}$. Use the sharing function of the $k$-out-of-$n$ threshold secret sharing scheme to share $r$. Let $(r_1, \ldots, r_n) \leftarrow \mathbf{TShare_n^k(r)}$. For each $i \in [n]$, construct the $i^{th}$ share of our scheme as follows : $share_i = (l_i, r_i)$. Output $(share_1, \ldots, share_n)$

2. **Reconstruction Function**($\mathbf{JNMRec_n^t}$)
   Without loss of generality assume that the authorized set $T$ has exactly $t$ elements, as we only use the first $t$ elements of $T$ and can ignore all other shares. On input $(share_{i_1}, \ldots, share_{i_t})$, for each $i \in T$, parse $share_i$ as $(l_i, r_i)$. Verify that all the $t$ shares of $r$, namely $(r_{i_1}, \ldots, r_{i_t})$ are consistent under $k$-out-of-$n$ Shamir's secret sharing scheme. In case this verification fails, output $\perp$. Otherwise, reconstruct $l$ by running the reconstruction procedure of the $t$-out-of-$n$ threshold secret sharing scheme, $\mathbf{TRec_n^t}$ on the $t$ shares of $l$, namely, $(l_{i_1}, \ldots, l_{i_t})$. Reconstruct $r$ by running the reconstruction procedure of the $k$-out-of-$n$ threshold secret sharing scheme, $\mathbf{TRec_n^k}$ on the $t$ shares of $r$, namely, $(r_{i_1}, \ldots, r_{i_t})$. Then decode $l$ and $r$ using decoding process of underlying non-malleable code to obtain : $m \leftarrow \mathbf{NMDec(l, r)}$ and output $m$.

**Correctness, efficiency and statistical privacy** : Correctness and efficiency trivially follows. A proof similar to one given for theorem 1 shows statistical privacy.

**Statistical Non Malleability** : The adversary specifies an authorized set $T$ of cardinality $t$. As the adversary is in $\mathcal{F}_{t,n}^{joint}$, it also specifies a partition $A$ and $B$ of $T$. Without loss of generality, let $A$ be the set containing larger number of shares. In fact, by our choice of $k$, only $A$ has at least $k$ shares. The adversary also specifies the two tampering functions, namely $\mathbf{F_A}$ and $\mathbf{F_B}$, corresponding to two subsets. That is, $\mathbf{F_A}$ (resp. $\mathbf{F_B}$) takes shares corresponding to set $A$ (resp. $B$) and outputs their tampered versions.

We follow a similar proof strategy as of theorem 1, and use the tampering functions provided by the adversary to create explicit tampering functions of the underlying non-malleable code. While in the previous proofs, the tampering of the each part was independent of the value of the other part, it is no longer the case in our current construction. In fact, to create a tampering function for one of the parts, we need information about the other part. We obtain the required information, by giving explicit leakage-function which leaks from the other part, and use the output of leakage-function in giving explicit tampering function. The underlying non-malleable code has been strengthened to handle such leakages. We then argue that any attack on our non-malleable secret-sharing scheme corresponds to an equivalent tampering attack on the underlying leakage-resilient non-malleable code, and can use the latter's simulator in designing our simulator. Now we give the formal reduction which explicitly defines $\mathbf{F}, \mathbf{G}, \mathbf{Leak}$.

1. (**Initial Setup**): For each $i \in A$, fix $l_i$ randomly (this is okay, since $A$ has less than $t$ shares). For each, $i \in B$, fix $r_i$ randomly (this is okay, since $B$ has less than $k$ shares). Choose and fix randomness $\$_A, \$_B$ and $\$_C$ uniformly and independently.

2. (**Sub-routine Error-Correcting Codes ECC**): Let $\mathbf{ECC} : \mathcal{R}^{|B|} \rightarrow \{0,1\}^p$ be any error-correcting code having distance $\frac{p}{2}$. We will use randomness $\$_C$ to sample $\log(\frac{1}{\epsilon})$ random locations from the encoded string of length $p$. (Note that **ECC** need not be efficient as the reduction is allowed to perform arbitrary computations. Also, any constant distance would also work, we have chosen these parameter for the ease of exposition. One could even use Hadamard encoding.)

3. (**Leak Function** : $\mathbf{Leak}(r)$) : On input $r$, use randomness $\$_A$ and the value of the fixed shares of $r$, namely $\{r_i : i \in B\}$, to interpolate $\{r_i : i \in A\}$. For each $i \in A$, create $share_i \leftarrow (l_i, r_i)$ using the fixed $l_i$. Run the tampering function $\mathbf{F_A}$ on the set of shares $\{share_i : i \in A\}$ to obtain tampered shares $\{\widetilde{share_i} : i \in A\}$. For each $i \in A$, parse $\widetilde{share_j}$ as $\widetilde{l_j}, \widetilde{r_j}$. Using the tampered shares of $l$ compute the following :

$$sum \leftarrow \sum_{i \in A} \alpha_i \times \widetilde{l_i}$$

where each $\alpha_i$ is the coefficient for Lagrangian interpolation at 0 of the unique $t-1$ degree polynomial passing through the points in set $T$ (this can be seen as partial reconstruction of the $t$-out-of-$n$ Shamir's secret sharing scheme [Sha79] where one obtains the secret as $\widetilde{l} \leftarrow \sum_{i \in A \cup B} \alpha_i \times \widetilde{l_i}$).

If the tampered shares of $r$, namely $\{\widetilde{r_i} : i \in A\}$, are inconsistent under a k-out-of-n Shamir's secret sharing scheme, then let $samples \leftarrow \perp$, otherwise use Shamir's secret sharing scheme interpolation operation to obtain the consistent values of the other tampered shares of $\widetilde{r}$, namely $\{\widetilde{cr_i} : i \in B\}$. Now concatenate all these (interpolated) values to obtain a string $\widetilde{cr} \leftarrow \widetilde{cr_{i_1}} \circ \widetilde{cr_{i_2}} \circ \ldots \circ \widetilde{cr_{i_{|B|}}}$, where $(i_1, \ldots, i_{|B|})$ are ordered indices present in the set $B$. Encode the string $\widetilde{cr} \in \mathcal{R}^{|B|}$ using **ECC** to obtain $\overline{cr} \leftarrow \mathbf{ECC}(\widetilde{cr})$. Use randomness $\$_C$ to sample $\log(\frac{1}{\epsilon})$ locations from the binary string $\overline{cr}$ to obtain

$$samples \leftarrow \overline{cr}\big|_{\$_C}$$

Output $(sum \circ samples)$.

4. (**Tampering function** $\mathbf{F}(l, \mathbf{Leak}(r))$): On input $l$ and $\mathbf{Leak}(r)$, interpolate the values of $\{l_i : i \in B\}$ using the input $l$, the fixed values of $\{l_i : i \in A\}$ and the fixed randomness $\$_B$. Using the fixed value of $\{r_i : i \in B\}$ construct the shares $\{share_i = (l_i, r_i) : i \in B\}$. Run the tampering function $\mathbf{F_B}$ on this set of shares to obtain $\{\widetilde{share_j} : j \in B\}$. For each $\{i \in B\}$, parse $\widetilde{share_j}$ as $\widetilde{l_j}, \widetilde{r_j}$.
Concatenate all the tampered shares of $r$, namely $\{\widetilde{r_i} : i \in B\}$, to obtain a string $\widetilde{r_B} \leftarrow \widetilde{r_{i_1}} \circ \widetilde{r_{i_2}} \circ \ldots \circ \widetilde{r_{i_{|B|}}}$, where $(i_1, \ldots, i_{|B|})$ are ordered indices present in the set $B$. Encode the string $\widetilde{r_B} \in \mathcal{R}^{|B|}$ using **ECC** to obtain $\overline{r_B} \leftarrow \mathbf{RS}(\widetilde{r_B})$. Use randomness $\$_C$ to sample $\log(\frac{1}{\epsilon})$ locations from the binary string $\overline{r_B}$ to obtain

$$samples_B \leftarrow \overline{r_B}\big|_{\$_C}$$

Parse $\mathbf{Leak}(r)$ as $(sum \circ samples)$. If the string $samples$ does not match the string $samples_B$, then output $\perp$, otherwise use the tampered shares of $l$ to compute (and output) the following :

$$\widetilde{l} \leftarrow sum + \sum_{i \in B} \alpha_i \times \widetilde{l_i}$$

where $\alpha_i$ is the Lagrangian's coefficient (as in the $\mathbf{Leak}(r)$ function above).

5. (**Tampering function $\mathbf{G}(r)$**) : On input $r$, proceed as in leakage function $\mathbf{Leak}(r)$ to compute tampered shares of $r$, namely $\{\widetilde{r_i} : i \in A\}$. Verify that all these tampered shares of $r$ are consistent under $k$-out-of-$n$ Shamir's secret sharing scheme. In case this verification fails, output $\perp$, otherwise reconstruct the tampered $\widetilde{r}$ by running $\mathbf{TRec^k_n}$ on $\{\widetilde{r_i} \in A\}$. Output $\widetilde{r}$.

This completes the description of our reduction. Now we show that the reduction creates shares from the correct distribution. As the number of fixed shares of $l$ and $r$ are less than the respective thresholds of $t$ and $k$, on receiving the actual values of $l$ and $r$, the remaining shares can be sampled from a distribution identical to the real sharing experiment (from perfect secrecy of Shamir's secret sharing scheme). It is also clear that if some message is hidden in the challenge $(l, r)$ (under the non-malleable code) then the same message is hidden in the $t$ shares created by our reduction (under our non-malleable secret sharing scheme). As the distribution of the $t$ shares created in the reduction is identical to the distribution in real-experiment, we can successfully invoke the tampering functions, namely $\mathbf{F_A}$ and $\mathbf{F_B}$, on these $t$ shares.

Now let us argue that the secret hidden in our $t$ tampered shares (under our secret-sharing scheme) will almost always be equal to the message hidden by $\widetilde{l} \leftarrow \mathbf{F}(l, \mathbf{Leak}(r))$ and $\widetilde{r} \leftarrow \mathbf{G}(r)$ (under non-malleable code). This will establish that the two tampering attacks are equivalent, and will allow us to use the simulator of the non-malleable code while creating our simulator. Notice that we used the same randomness $\$_A$ to sample the shares of $r$ to ensure consistency in between the functions **Leak** and **G**.

In case the $t$ tampered shares of $r$, namely $\{\widetilde{r_i} : i \in T\}$ , are consistent under a k-out-of-n Shamir's secret sharing scheme, then any $k$ shares can be used for reconstructing $\widetilde{r}$. In particular, in this case, the value of $\widetilde{r} \leftarrow \mathbf{G}(r)$ will be equal to the value of $\widetilde{r}$ calculated while executing $\mathbf{JNMRec^t_n}$, the reconstruction procedure of our non-malleable secret-sharing scheme. Moreover, in this case, as $sum = \sum_{i \in A} \alpha_i \times \widetilde{l_i}$, the value calculated by $\mathbf{F}(l, \mathbf{Leak}(r)))$ can be interpreted as $\widetilde{l} \leftarrow \sum_{i \in A} \alpha_i \times \widetilde{l_i} + \sum_{i \in B} \alpha_i \times \widetilde{l_i}$. On observing that the sets A and B are disjoint, we get that $\sum_{i \in A} \alpha_i \times \widetilde{l_i} + \sum_{i \in B} \alpha_i \times \widetilde{l_i} = \sum_{i \in A \cup B} \alpha_i \times \widetilde{l_i}$. The latter expression is identical to the reconstruction formula for t-out-of Shamir's secret sharing scheme, which is used in $\mathbf{JNMRec^t_n}$. Therefore, in this case, both $\widetilde{l}$ and $\widetilde{r}$ calculated by our reduction equals the values obtained while executing $\mathbf{JNMRec^t_n}$ on the $t$ tampered shares.

Now we argue the trickier case, in which the $t$ tampered shares of $r$ are inconsistent and therefore $\mathbf{JNMRec^t_n}$ always outputs $\perp$. If the inconsistency is in the shares of subset $A$, then the function $\mathbf{G}$ trivially detects it and outputs $\perp$. Otherwise, the inconsistency is introduced because of the tampered shares of $r$ in set $B$. In such a case, we use properties of Error-correcting coding scheme **ECC** to argue that the tampering function $\mathbf{F}$ outputs $\perp$ with probability at least $1 - \epsilon$. As **ECC** has distance $\frac{p}{2}$, for any two different inputs, the encoded outputs will differ in at least $\frac{p}{2}$ out of $p$ locations. Now if we randomly sample one of these $p$ locations, we will hit a location for which the two encodings differ with probability at least $\frac{1}{2}$. By repeating with uniform randomness ($\$_C$), the probability that we would pick at least one differing location in $\log(\frac{1}{\epsilon})$ independent repetitions is at least $1-\epsilon$. Therefore, if the strings $\widetilde{cr}$ and $\widetilde{r_B}$ differ, then with probability at least $1-\epsilon$, the strings *samples* and $samples_B$ will not match and the tampering function $\mathbf{F}$ will output $\perp$. Otherwise, with probability at most $\epsilon$, our reduction fails to detect a $\perp$ and outputs valid values of tampered $\widetilde{l}$ and $\widetilde{r}$. Therefore, overall the statistical distance in between message encoded between $\widetilde{l}$ and $\widetilde{r}$ constructed by reduction and the secret computed by the reconstruction $\mathbf{JNMRec_n^t}$ can be at most $\epsilon$.

We can now say that the tampering experiments for the non-malleable secret-sharing schemes (see definition 6) and the underlying leakage-resilient non-malleable codes (see definition 8) are statistically indistinguishable, specifically,

$$\mathbf{STamper_m^{(F_A,F_B),T}} \approx_\epsilon \mathbf{LTamper_m^{F,G,Leak}}$$

As we know by the properties of leakage-resilient non-malleable codes (8),

$$\mathbf{LTamper_m^{F,G,Leak}} \approx_\epsilon \mathbf{LSim_m^{F,G,Leak}}$$

We use the simulator of the underlying non-malleable code as the simulator of our non-malleable secret sharing scheme. That is, we let

$$\mathbf{SSim_m^{(F_A,F_B),T}} \equiv \mathbf{LSim_m^{F,G,Leak}}$$

By applying triangle inequality to above relations we complete the proof of non-malleability.

$$\mathbf{STamper_m^{(F_A,F_B),T}} \approx_{2\epsilon} \mathbf{SSim_m^{(F_A,F_B),T}}$$

□

## 5 Leakage Resilient Non-Malleable Codes

In this section, we build 2 split-state non-malleable codes that satisfy a strong leakage-resilience property. In particular, we allow the leakage-from the second part of non-malleable code to be as large as the size of the first part, and allow the tampering of the first part to depend on this additional leakage. As the existing 2 split-state non-malleable codes have the two parts of equal length, they are insufficient for our needs.

There are two major approaches to construct 2 split-state non-malleable codes, one uses the properties of inner-product (as a two-source extractor [DKO13], further rely on additive combinatorics for analysis [ADL14]) and other proceeds by constructing 2-source non-malleable extractor [CGL16, Li17]. Unfortunately, we cannot directly use inner-product as a result of our leakage-model, and it is not clear how to extend the highly non-trivial analysis of [ADL14] to work with other two-source extractors (like Raz's extractor [Raz05]).

Therefore, we take the second approach and modify the construction of Chattopadhyay et al. [CGL16] to incorporate leakage-resilience. Their construction constructs 2-source non-malleable extractors with efficient pre-image sampler to obtain a split-state non-malleable code (using the reduction of Cheraghchi and Guruswami [CG14]).We follow their approach and first construct a leakage-resilient 2-source non-malleable extractor that works with different length sources and then giving a pre-image sampler for this extractor to arrive at the non-malleable codes completing the proof of Theorem 3. As the construction of [CGL16] is very complex, we describe the main ideas below. In the next subsections, we formally define all these objects, and give the full construction :

**Construction of the decoder**

As we will be modifying the 2-source non-malleable extractor construction of Chattopadhyay et al. [CGL16], we try to give some intuition behind their construction. At a very high level, given two sources, CGL first generates an advice using these two sources and guarantees that any (independent) tampering of these two sources will almost always result in a different advice being generated. Then they run a sequence of extractors on these two sources, and the executed sequence depends on the advice. That is, different values of advice necessarily lead to different sequence of extractor invocations. They utilize this difference in achieving the non-malleability property and guarantee that any tampering will force the extractor to have an "unrelated" output.

Unfortunately, the two sources in their construction (also of [GKP$^+$18]) are of equal lengths and the adversary can simply leak the entire second source in our leakage-model. Therefore to make their construction work in our model, we have to make the following changes :

- **Use Raz's extractor instead of inner-product.** In CGL, in the intial phases, there are two invocations of inner-product as a two-source extractor. As inner-product cannot work in our case, we use Raz's extractor [Raz05] that works with uneven length sources. Now the leakage from the larger source can be thought of as a loss of min-entropy from it, and Raz's extractor can handle slightly lower min-entropy as well, we can use it to obtain uniform-randomness even in our leakage-model.
- **Change the parameters of linear-seeded extractor.** We change the parameters of seeded extractors, to increase the min-entropy and length of the source to handle the loss of min-entropy caused by leakage. We need not worry about the seed being leaked, because the seed is never present in any of the two-sources, and is always uniformly random, as it is output of either a leakage-resilient two-source extractor or some other leakage-resilient seeded-extractor. [10]
- **Modify advice generator.** The advice generator in CGL works by slicing each of the two-sources, and executing an inner-product on the two-slices to obtain a uniform output (recall that the two-sources are independent). Then they use the generated randomness to randomly sample from some encoding of both the sources (with slices removed). Then they use the high distance of the encoding, to guarantee that any tampering of the source(with slice removed), will lead to a very different encoding, and will be caught by the random sampling. The final tag they generate has the two slices, and the random sampling from the encodings (of the sources). They get the required property by noticing that any change in the source, will either change the slice or the random sampling, and this will result in a different tag.

  As already noted, we will use Raz's extractor instead of inner-product for generating randomness from the two slices. While the output will be uniformly random, the length of our tag will be unacceptably large (the second slice needs to be at least as large as the first source because of our leakage-model). This breaks the construction, as the length of the advice dictates the number of following executions of the seeded-extractor, and it is trivial to see that number of extractions from the first source cannot be larger than the number of bits in the first source. Therefore, instead of having these slices explicitly in our generated advice, we only have the output of the Raz's extractor (which can be much smaller than the larger slice). We extend the argument of sampling from the source(with slice removed) to sampling from the full source (including the slice). This makes the analysis slightly more non-trivial,

[10]Note that the use of inner-product in the construction of CGL's linear-seeded extractor (lemma. 5) is fine because one of the sources of inner product is a slice of the seed (that is always unifrom), and the other source of inner product is randomly sampled from a very large-source that cannot be fully leaked.

as we are now using the value of the source to generate uniform randomness, and then we are crucially using this randomness to sample from the same source. Fortunately, these two sources are independent, and the ideas present in the analysis of [CGL16, Coh16] can be extended to work in our case as well.

**Construction of the encoder**

For constructing the encoder of the non-malleable code, [CGL16] provides a procedure to uniformly pre-image sample from the output of the 2-source non-malleable extractor. As their extractor construction is highly complex, uniformly inverting the same requires delicate care. To this end, they device an efficient inverter for each of the components of their final extractor. This includes, uniformly sampling the value of the source, given the value of output and seed for the linear seeded extractors. Similarly, they need to invert the inner-product (which can be done efficiently) and uniformly sample the two sources which encode the given output. They also have to ensure that the sources that they have sampled is consistent with advice (in particular the random-sampling present in the advice). Once they have an efficient pre-image sampling procedure for each of the sub-components, they invert the whole non-malleable extractor by first randomly sampling the advice and all the seeds, after which they can uniformly sample the value of sources which lead to these seeds and output.

Unfortunately, we run into two more problems, firstly, unlike inner-product, it is not clear how to uniformly and efficiently invert Raz's extractor [11]. Secondly, a little more care is required while encoding because we have changed the construction of the advice generator. Fortunately, the techniques of [CGL16] can be extended to solve the second issue. We solve the first issue, by interleaving the process of sampling seeds and source. In particular, we ensure that we do not need to invert Raz's extractor, and adjust the procedure of [CGL16] accordingly. While we make specific changes in various parts of the encoding and decoding procedures, for the sake of completeness we give the entire procedure. To ensure consistency, some of the unchanged parts are verbatim as in [CGL16].

## 5.1 Preliminaries

Before constructing leakage-resilient non-malleable extractors we recall some useful definitions and lemmas from the literature.

**Definition 9.** ***(min-entropy)***. *The min-entropy of a source $X$ is defined as*

$$H_\infty(X) = \min_{x \in Support(X)} \left\{ \frac{1}{log(Pr[X = x])} \right\}$$

*A $(n, k)$-source is a distribution on $\{0, 1\}^n$ with min-entropy $k$. A distribution $\mathbf{D}$ is flat if it is uniform over a set $S$.*

**Definition 10.** ***(Strong seeded extractor)*** *A function* $\mathbf{Ext} : \{0, 1\}^n \times \{0, 1\}^d \to \{0, 1\}^m$ *is called a strong seeded extractor for min-entropy $k$ and error $\epsilon$ if for any $(n, k)$-source $X$ and an independent uniformly random string $U_d$, we have*

$$|\mathbf{Ext}(X, U_d) \circ U_d - U_m \circ U_d| < \epsilon,$$

[11] Raz's extractor uses a "merger" as one of its sub-components, which performs an "indexing operation" that is a non-linear operation. [CGL16] crucially used the linearity property of the inner-product for efficient pre-image sampling.

*where $U_m$ is independent of $U_d$. Further if the function* $\mathbf{Ext}(\cdot, u)$ *is a linear function over* $\mathbb{F}_2$ *for every* $u \in \{0,1\}^d$, *then* **Ext** *is called a linear seeded extractor.*

**Definition 11.** *(**Two Source Extractor**) A function $IExt : (\{0,1\}^{n_1} \times \{\{0,1\}^{n_2}\}) \to \{0,1\}^m$ is an 2 source extractor for min-entropy $(k_1, k_t)$ and error $\epsilon$, if for any independent $(n_i, k_i)$-sources for each $i \in [2]$ we have*

$$|IExt(X_1, X_2) - U_m| \leq \epsilon$$

*Further, $IExt$ is called a strong two source extractor if its output is uniform even if one of the two sources is given. More formally,*

$$|IExt(X_1, X_2), X_1 - U_m, X_1| \leq \epsilon \text{ and } |IExt(X_1, X_2), X_2 - U_m, X_2| \leq \epsilon$$

As we need two source extractors with uneven length sources and exponentially small error, we recall a theorem proved by Raz in [Raz05].

**Theorem 3.** *( [Raz05]) For any $n_1, n_2, k_1, k_2, m$ and $0 < \delta < 1/2$, such that*

$$\begin{aligned}
&n_1 \geq 6 \log n_1 + 2 \log n_2, \\
&k_1 \geq (0.5 + \delta) \cdot n_1 + 3 \log n_1 + \log n_2, \\
&k_2 \geq 5 \log(n_1 - b_1), \\
&m \leq \delta \cdot \min[n_1/8, k_2/40] - 1
\end{aligned}$$

*then there exists an explicit strong 2 source extractor* $\mathrm{RExt} : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}^m$, *for min-entropy $(k_1, k_2)$ and error $\epsilon = 2^{-1.5m}$.*

As we will be handling leakages, it will convenient to define the following notion, which helps to argue how much "min-entropy" will be left in a variable even after knowing the value of another "dependent" variable.

**Definition 12.** *( [DORS08]) (**average conditional min-entropy**). The average conditional min-entropy is defined as*

$$\tilde{H}_\infty(X|W) = -log\Big(E_{w \leftarrow W}\Big[\max_x Pr[X = x|W = w]\Big]\Big) = -log\, E_{w \leftarrow W}\left[2^{-H_\infty(X|W=w)}\right]$$

It is sometimes convenient to work with average case seeded extractors, where if a source $X$ has average case conditional min-entropy $\tilde{H}_\infty(X|Z) \geq k$ then the output of the extractor is uniform even when $Z$ is given.

**Lemma 2.** *( [DORS08]) If a random variable $B$ can take at most $l$ values, then $\tilde{H}_\infty(A|B) \geq H_\infty(A) - \log l$.*

**Lemma 3.** *( [DORS08]) For and $\delta > 0$, if $Ext$ is a $(k, \epsilon)$-strong seeded extractor then it is also a $(k + log(1/\delta), \epsilon + \delta)$-strong seeded average case extractor.*

A corresponding analog for two-source extractors is the following.

**Lemma 4.** *Let* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ *be a 2-source extractor for min-entropy $k$ and error $\epsilon$. Then for any $\delta > 0$,* Ext *is $\mu$-leakage resilient for min entropy $k + t\sec + \log(\frac{1}{\delta})$ and error $\epsilon + \delta$.*

Now we recall the building blocks used in the non-malleable code construction of Chattopadhyay et al. in [CGL16].

**Averaging Sampler** Samp. Let $f : [n] \to [0, 1]$ be an arbitrary function. An averaging sampler Samp is an algorithm that takes a random string as input and outputs distinct samples $(a_1, \cdots, a_t)$ where each $a_i \in [n]$ and $t = o(n)$. The property of these samples is that $(1/t) \sum_{i=1}^{t} f(a_i)$ is "close" to the actual mean with some probability.

**Definition 13** (Averaging sampler [Vad04])**.** *A function* $\mathrm{Samp} : \{0, 1\}^r \to [n]^t$ *is a* $(\mu, \theta, \gamma)$ *averaging sampler if for every function* $f : [n] \to [0, 1]$ *with average value* $\frac{1}{n} \sum_i f(i) \geq \mu$*, it holds that*

$$\Pr_{i_1,\ldots,i_t \leftarrow \mathrm{Samp}(U_R)} \left[ \frac{1}{t} \sum_i f(i) \leq \mu - \theta \right] \leq \gamma.$$

Samp *has distinct samples if for every* $x \in \{0, 1\}^r$*, the samples produced by* $\mathrm{Samp}(x)$ *are all distinct.*

**Corollary 2.** *( [CGL16]) For any constants* $\delta_{\mathrm{Samp}}, \nu_{\mathrm{Samp}} > 0$*, there exist constants* $\alpha, \beta < \nu_{\mathrm{Samp}}$ *such that for all* $n > 0$ *and any* $r \geq n^{\alpha}$ *there exists a polynomial time computable function* $\mathrm{Samp} : \{0, 1\}^r \to [n]^{t_{\mathrm{Samp}}}$ $t_{\mathrm{Samp}} = O(n^{\beta})$ *satisfying the following property: for any set* $S \subset [n]$ *of size* $\delta_{\mathrm{Samp}} n$*,*

$$\Pr[|\mathrm{Samp}(U_r) \cap S| \geq 1] \geq 1 - 2^{-\Omega(n^{\alpha})}.$$

*Further* Samp *has distinct samples.*

**Definition 14.** *( [CGL16]) For any seeded extractor* $\mathrm{Ext} : \{0, 1\}^n \times \{0, 1\}^d \to \{0, 1\}^m$*, any* $s \in \{0, 1\}^d$ *and* $r \in \{0, 1\}^m$*, we define:*

- $\mathrm{Ext}(\cdot, s) : \{0, 1\}^n \to \{0, 1\}^m$ *to be the map* $\mathrm{Ext}(\cdot, s)(x) = \mathrm{Ext}(x, s)$.
- $\mathrm{Ext}^{-1}(r)$ *to be the set* $\{(x, y) \in \{0, 1\}^n \times \{0, 1\}^d : \mathrm{Ext}(x, y) = r\}$.
- $\mathrm{Ext}^{-1}(\cdot, s)$ *to be the set* $\{x : \mathrm{Ext}(x, s) = r\}$.

**Lemma 5.** *( [CGL16] Lemma 8.5) There exists an efficient deterministic function* $\mathrm{iExt} : \{0, 1\}^n \times \{0, 1\}^d \to \{0, 1\}^m$*, such that, if* $X$ *is a* $(n, 0.9n)$ *source and* $S$ *is an independent uniform seed on* $\{0, 1\}^d$*, then the following holds:*

$$|\mathrm{iExt}(X, S), S - U_m, S| < 2^{-n^{\Omega(1)}}.$$

*Further for any* $r \in \{0, 1\}^m$ *and any* $s \in \{0, 1\}^d$*,* $|\mathrm{iExt}(\cdot, s)^{-1}(r)| = 2^{n-m}$*. Moreover, there exists a polynomial time algorithm* $\mathrm{Samp}_1$ *that takes as input* $r \in \{0, 1\}^m$*, and samples from a distribution that is uniform on* $\mathrm{iExt}^{-1}(r)$.

## 5.2 Decoder (two-source non-malleable extractor)

Below we modify the parameters of [CGL16] and then modify their construction to make it leakage-resilient.

### Parameters

Along with the parameters in [CGL16], we introduce more parameters to take care of the leakage. Suppose the length of the left source is $n$, to not complicate the parameters further, we just multiply the length of each of the $n^{\omega(1)}$ blocks of the right source by $n$. Such an increase (instead of an additive increase of $n + \log(\frac{1}{\epsilon})$) allows for leakage of much more than $n$ bits. This idea can be generalized to any *poly*$(n)$ for any (a-priori) fixed poly, and in such a case, length of each block will be increased by *poly*$(n) + \log(\frac{1}{\epsilon})$.

1. Let $n$ be the bit-length of the first source $x$, and $p = n^2$ be the bit-length of the second source.

2. Let $\gamma$ be a small enough constant and $C$ a large one. Let $t = n^{\gamma/C}$.
3. Let $n_1 = n^{\beta_1}$, $\beta_1 = 10\gamma$. Let $p_1 = n \times n_1$. Let $n_2 = n - n_1$ and $p_2 = p - p_1$. Let $\mathrm{LExt}_1 : \{0, 1\}^{n_1} \times \{0,1\}^{p_1} \to \{0,1\}^{2n_3}$, $n_3 = \frac{n_1}{20}$ be the strong two-source extractor from Theorem 3.
4. Let $\mathbb{F}_1$ be the finite field $\mathbb{F}_{2^{\log(n+1)}}$. Let $n_4 = \frac{n}{\log(n+1)}$. Let $\mathbf{RS}_1 : \mathbb{F}_1^{n_4} \to \mathbb{F}_1^n$ be the Reed-Solomon code encoding $n_4$ symbols of $\mathbb{F}_1$ to $n$ symbols in $\mathbb{F}_1$ (we overload the use of $\mathbf{RS}$, using it to denote both the code and the encoder). Thus $\mathbf{RS}_1$ is a $[n, n_4, n - n_4 + 1]_n$ error correcting code.
5. Let $\mathbb{F}_2$ be the finite field $\mathbb{F}_{2^{\log(n^2+1)}}$. Let $p_4 = \frac{p}{\log(n^2+1)}$. Let $\mathbf{RS}_2 : \mathbb{F}_2^{p_4} \to \mathbb{F}_2^p$ be the Reed-Solomon code encoding $p_4$ symbols of $\mathbb{F}_2$ to $p$ symbols in $\mathbb{F}_2$ (we overload the use of $\mathbf{RS}$, using it to denote both the code and the encoder). Thus $\mathbf{RS}_2$ is a $[p, p_4, p - p_4 + 1]_{n^2}$ error correcting code.
6. Let $\mathrm{Samp}_1 : \{0,1\}^{n_3} \to [n]^{n_5}$ be a $(\mu, \frac{1}{10}, 2^{-n^{\Omega(1)}})$ averaging sampler with distinct samples. We set $n_5 = n^{\beta_2}$, $\beta_2 < \beta_1/2$.
7. Let $\mathrm{Samp}_2 : \{0,1\}^{n_3} \to [n^2]^{n_5}$ be a $(\mu, \frac{1}{10}, 2^{-n^{\Omega(1)}})$ averaging sampler with distinct samples.
8. Let $\ell = n_3 + 3n_5 \log n < 5n^{\beta_1}$. Thus $\ell \leq n^{11\gamma}$.
9. Let $n_6 = 50Ct\ell$. Let $p_6 = n \times n_6$. Let $\mathrm{LExt}_2 : \{0,1\}^{n_6} \times \{0,1\}^{p_6} \to \{0,1\}^{2n_q}$, $n_q = 3Ct\ell$, be the strong two-source extractor from Theorem 3.
10. Let $n_7 = n - n_1 - n_6$ and $p_7 = p - p_1 - p_6$. Let $n_x = \frac{n_7}{8\ell}$. Let $n_y = \frac{p_7}{16Ct\ell}$. Thus $n_x \geq n^{1-15\gamma}$.
11. Let $d_1 = 80\ell$.
12. Let $\mathrm{iExt}_1 : \{0,1\}^{n_x} \times \{0,1\}^{d_1} \to \{0,1\}^{d_2}$, $d_2 = 40\ell$, be the extractor from Lemma 5.
13. Let $\mathrm{iExt}_2 : \{0,1\}^{n_q} \times \{0,1\}^{d_2} \to \{0,1\}^{d_3}$, $d_3 = 20\ell$, be the extractor from Lemma 5.
14. Let $\mathrm{iExt}_3 : \{0,1\}^{n_x} \times \{0,1\}^{d_3} \to \{0,1\}^{d_4}$, $d_4 = 10\ell$ be the extractor from Lemma 5.
15. Let $\mathrm{iExt}_4 : \{0,1\}^{n_y} \times \{0,1\}^{d_4} \to \{0,1\}^{d_5}$, $d_5 = 5\ell$, be the extractor from Lemma 5.
16. Let $\mathrm{LExt} : \{0,1\}^{4Ctn_y} \times \{0,1\}^{d_4} \to \{0,1\}^{2n_q}$ be defined in the following way. Let $v_1, \ldots, v_{4t}$ be strings, each of length $n_y$. Define $\mathrm{Ext}(v_1 \circ \ldots \circ v_{4Ct}, s) = \mathrm{iExt}_4(v_1, s) \circ \ldots \circ \mathrm{iExt}_4(v_{4Ct}, s)$.

**Algorithm 1** NMExt(x,y)

**Input:** Bit strings $x, y$ of length $n$ and $n^2$ respectively.
**Output:** A bit string of length $m$.

```
1:  procedure NMExt(x,y)
2:      Let $x_1 = \mathrm{Slice}(x, n_1)$, $y_1 = \mathrm{Slice}(y, p_1)$. Compute $\nu = (\nu_1 \circ \nu_2) = \mathrm{LExt}_1(x_1, y_1)$.
3:      Let $x_2, y_2$ be the strings formed by removing $x_1, y_1$ from $x, y$ respectively.
4:      Let $T_1 = \mathrm{Samp}_1(\nu_1) \subset [n]$.
5:      Let $T_2 = \mathrm{Samp}_2(\nu_2) \subset [n^2]$.
6:      Interpret $x, y$ as elements in $\mathbb{F}_1^{n_4}$ and $\mathbb{F}_2^{p_4}$.
7:      Let $\overline{x} = \mathbf{RS}_1(x), \overline{y} = \mathbf{RS}_2(y)$.
8:      Let $\overline{x}_1 = (\overline{x})_{\{T_1\}}, \overline{y}_1 = (\overline{y})_{\{T_2\}}$, interpreting $\overline{x}_2 \in \mathbb{F}^n$ and $\overline{y}_2 \in \mathbb{F}^{n^2}$.
9:      Let $z = \nu \circ \overline{x}_1 \circ \overline{y}_1$, where $z$ is interpreted as a binary string.
10:     Interpret $x_2, y_2$ as binary strings.
11:     Output $\mathrm{nmExt}_1(x_2, y_2, z)$.
12: end procedure
```

**Analysis of new advice generator.** Before we analyse the new advice geneator, let us recall a simple lemma from [Coh16] (with variables renamed).

**Lemma 6.** *( [Coh16]) Let $V, X_1, X_1'$ be random variables such that for any $x_1 \in support(X_1)$, the random variables $(V|X_1 = x_1)$ and $(X_1'|X_1 = x_1)$ are independent. Assume that $V$ is supported on*

**Algorithm 2** nmExt$_1(x_2, y_2, z)$

```
1: procedure nmExt$_1(x_2, y_2, z)$
2:     Let $x_3 = \text{Slice}(x_2, n_6), y_3 = \text{Slice}(y_2, p_6)$. Let $w, v$ be the remaining parts of $x_2, y_2$ respectively.
3:     Let $\text{LExt}_2(x_3, y_3) = (q_{1,1}, q_{1,2})$, where each of $q_1$ is of length $n_q$.
4:     Let $w_1, \dots, w_{8\ell}$ be an equal sized partition of the string $w$ into $8\ell$ strings.
5:     Let $v_1, \dots, v_{16Ct\ell}$ be an equal sized partition of the string $v$ into $16Ct\ell$ strings.
6:     for $h = 1$ to $\ell$ do
7:         $(q_{h+1,1}, q_{h+1,2}) = \text{2ilaExt}(v_{[8C(h-1)t+1, 8Cht]}, w_{[4h-3,4h]}, q_{h,1}, q_{h,2}, z_{\{h\}})$
8:     end for
9:     Output $(q_{\ell+1,1}, q_{\ell+1,2})$.
10: end procedure
```

**Algorithm 3** 2ilaExt$(v_{[8C(h-1)t+1,8Cht]}, w_{[4h-3,4h]}, q_{h,1}, q_{h,2}, b)$

```
1: procedure 2ilaExt$(v_{[8C(h-1)t+1,8Cht]}, w_{[4h-3,4h]}, q_{h,1}, q_{h,2}, b)$
2:     Let $s_{h,1} = \mathit{Slice}(q_{h,1}, d_1)$, $r_{h,1} = \text{iExt}_1(w_{4h-3}, s_{h,1})$, $s_{h,2} = \text{iExt}_2(q_{h,2}, r_{h,1})$, $r_{h,2} = \text{iExt}_3(w_{4h-2}, s_{h,2})$.
3:     if $b = 0$ then
4:         Let $r_h = \text{Slice}(r_{h,1}, |r_{h,2}|)$.
5:     else
6:         Let $r_h = r_{h,2}$
7:     end if
8:     Let $(\overline{q}_{h,1}, \overline{q}_{h,2}) := \text{LExt}(v_{[8C(h-1)t+1, 8(h-1)t+4Ct]}, r_h)$.
9:     Let $\overline{s}_{h,1} = \mathit{Slice}(\overline{q}_{h,1}, d_1)$, $\overline{r}_{h,1} = \text{iExt}_1(w_{4h-3}, \overline{s}_{h,1})$, $\overline{s}_{h,2} = \text{iExt}_2(\overline{q}_{h,2}, \overline{r}_{h,1})$, $\overline{r}_{h,2} = \text{iExt}_3(w_{4h-2}, \overline{s}_{h,2})$.
10:    if $b = 0$ then
11:        Let $\overline{r}_h = \overline{r}_{h,2}$
12:    else
13:        Let $\overline{r}_h = \text{Slice}(\overline{r}_{h,1}, |r_{h,2}|)$.
14:    end if
15:    Let $(q_{h+1,1}, q_{h+1,2}) := \text{LExt}(v_{[8C(h-1)t+4Ct+1, 8(h-1)t]}, \overline{r}_h)$.
16:    Output $(q_{h+1,1}, q_{h+1,2})$.
17: end procedure
```

$\{0,1\}^n$. *Then,*

$$|(V, X_1, X_1') - (U_n, X_1, X_1')| = |(V, X_1) - (U_n, X_1)|$$

*where $|A - B|$ denotes the statistical distance between the distributions $A$ and $B$.*

Now we show that our advice generator detect tampering of any of the two sources with high probability.

**Lemma 7.** *In case an adversary tampers the left source $X$ to $X' \leftarrow f(X)$ and the right source $Y$ to $Y' \leftarrow g(Y)$, with probability at least $1 - 2^{-n^{\Omega(1)}}$, the tags $Z \neq Z'$.*

*Proof.* If $f$ has no fixed points, then $X \neq X'$, where $X = X_1 \circ X_2$. Let $V' \leftarrow \text{LExt}_1(X_1', Y_1')$. Recall that in our construction the tag is computed as $Z \leftarrow V \circ \overline{X}_1 \circ \overline{Y}_1$. Therefore, if $V \neq V'$, then $Z \neq Z'$. Now using the strongness of two-source extractor, and properties of averaging sampler we will handle the case of $V = V'$.As $\text{LExt}_1$ is a strong two-source extractor(Theorem 3), we get

$$V, X_1 \approx_{2^{-\Omega(n_1)}} U_{n_2}, X_1$$

Further by lemma 6,

$$V, X_1, X_1' \approx_{2^{-\Omega(n_1)}} U_{n_2}, X_1, X_1'$$

Indeed the hypothesis of lemma 6 is met as conditioned on fixing of $X_1$, $V$ becomes a deterministic function of $Y$ and is independent of $X_1'$ which is a deterministic function of $X$.

Thus, even after fixing $x_1, x_1'$, $V$ is $2^{-\Omega(n_1)}$ close to uniform (with probability at least $1-2^{-\Omega(n_1)}$). As $V$ is now a deterministic function of $Y$, it can be used to sample from the $X$ which is independent of $V$ conditioned on this fixing. Also note that $V = V'$.

Since $f$ has no fixed points, it follows that since $E$ is an encoder of a code with relative distance distance $\frac{1}{10}$, $\Delta(E(X), E(X')) \geq \frac{n}{10}$. Let $D = \{j \in [n] : E(X)_{\{j\}} \neq E(X')_{\{j\}}\}$. Thus $|D| \geq \frac{n}{10}$. Using Corollary 2, it follows that with probability at least $1 - 2^{-\Omega(n_1)}$, $|D \cap \text{Samp}(V)| \geq 1$, and thus $\overline{X_1} \neq \overline{X_1'}$ (since $\text{Samp}(V) = \text{Samp}(V^{(i)})$). This proves the lemma when $f$ has no fixed points. In a similar manner the case when $g$ has no fixed points can be handled. □

**Leakage-resilient 2-source non-malleable extractor.** We define leakage-resilient strong non-malleable extractors, the object we want to construct. We use the definition given by Goyal et al. [GKP+18]. We note that they only needed to leak a-priori bounded $\lambda$ bits of information about the second source, where $\lambda$ did not depend on $n$.

**Definition 15.** *A 2-source non-malleable extractor* $\text{nmExt} : \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}^m$ *is said to be $\mu$-leakage resilient for min-entropy $k_1, k_2$ and error $\epsilon$ if it satisfies the following: if $X$ is an $(n_1, k_1)$-source and $Y$ is an independent $(n_2, k_2)$-sources, $\mathcal{A} = (f, g)$ is an arbitrary 2-split-state tampering functions and $\{\mathsf{Leak} : \{0,1\}^{n_2} \to \{0,1\}^{\mu}\}$ is an arbitrary leakage function, then there exists a random variable $D_{\vec{f},\vec{g},\overrightarrow{\mathsf{Leak}}}$ on $(\{0,1\}^m \cup \{\mathsf{same}^\star\})$ which is independent of the sources $X$ and $Y$ such that*

$$|\text{nmExt}(X, Y), \text{nmExt}(f(X, \mathsf{Leak}(Y)), g(Y)) - U_m, Y, \text{copy}(D_{\vec{f},\vec{g},\overrightarrow{\mathsf{Leak}}}, U_m)| < \epsilon$$

*where both $U_m$'s refer to the same uniform $m$-bit string.*

Using the proof technique of [CGL16], we can arrive at the following result. As the full proof is quite long, we only give a very brief sketch below. For full proof, please refer [CGL16].

**Theorem 4.** *Let* NMExt *be the function computed by Algorithm 1. Then* NMExt *is a $(\mu)$-leakage resilient 2-source non-malleable extractor for min-entropy $(n - n^{\Omega(1)}, n^2 - n - n^{\Omega(1)})$ with error $2^{-n^{\Omega(1)}}$.*

**Proof Sketch.** Suppose there was $n$ bits of leakage from the second source. We fix this leakage. By fixing the leakage, by lemma 2, we know that the second source looses at most $n + log(1/\epsilon)$ bits of min-entropy except with probability $\epsilon$. After the advice has been generated, the non-malleable extractor only performs extractor operations using chunks of the two sources. For ensuring the sanity of the non-malleable extractor, the proof of [CGL16] only requires that the output of each of these invocations is uniform (even after having fixed previous outputs). Now notice, that we have not changed anything, apart from increasing the sizes of each of the chunks of the second source. The sizes of the chunk has been changed to ensure that even after fixing the leakage from the second source, we will have enough min-entropy in each of the chunks. Therefore, we can successfully invoke Raz's two-source extractor to generate uniform seed for alternating extraction. Subsequently, this uniform seed can be used to extract uniform bits from the chunks of the two-sources using a strong seeded extractor. Therefore, one can duplicate the analysis of [CGL16] and prove that such a construction is in fact a 2-source non-malleable extractor.

## 5.3 Encoder (Pre-Image sampler for non-malleable extractor)

As in [CGL16], we follow a similar sampling strategy and first argue about the invertibility of each of the sub-algorithms and then show a way to compose these sampling procedure to sample uniformly from the pre-image of nmExt. We refer to all the variables, sub-routines and notations introduced in these algorithms while developing the sampling procedures. Unless we state otherwise, by a subspace we mean a subspace over $\mathbb{F}_2$, the finite field with cardinality two.

We first show how to sample uniformly from the pre-image of 2ilaExt (Algorithm 3 ), since it is a crucial sub-part of nmExt. We have the following lemma.

**Lemma 8.** *For any fixing of the variables* $q_{1,1}, q_{1,2}$, $\{r_{1,i}, \overline{s}_{1,i}, \overline{r}_{1,i} : i \in \{1,2\}\}$, *and any* $b \in \{0,1\}$ *define the set:*

$$\begin{aligned}\text{2ilaExt}^{-1}(q_{2,1}, q_{2,2}) = \{(v_{[1,8Ct]}, w_{[1,4]}) \in \{0,1\}^{8Ctn_y+4n_x} :\\ \text{2ilaExt}(v_{[1,8Ct]}, w_{[1,4]}, q_{1,1}, q_{1,2}, b) = (q_{2,1}, q_{2,2})\}\end{aligned}$$

*There exists an efficient algorithm* $\text{Samp}_2$ *that takes as input* $q_{2,1}, q_{2,2}, q_{1,1}, q_{1,2}, b, \{r_{1,i}, \overline{s}_{1,i}, \overline{r}_{1,i} : i \in \{1,2\}\}$, *and samples uniformly from* $\text{2ilaExt}^{-1}(q_{2,1}, q_{2,2})$.

*Further, the set* $\text{2ilaExt}^{-1}(q_{2,1}, q_{2,2})$ *is a subspace over* $\mathbb{F}_2$, *and its size does not depend on the inputs to* $\text{Samp}_2$.

*Proof.* We being by showing that fixing the variables $q_{1,1}, q_{1,2}, r_{1,1}$ also fixes $s_{1,1}$ and $s_{1,2}$ uniquely. Recall that, $q_{1,1}$ is sliced to obtain $s_{1,1} \leftarrow \mathit{Slice}(q_{1,1}, d_1)$. Also, $q_{1,2}$ and $r_{1,1}$ are used to obtain the value for $s_{1,2} \leftarrow \text{iExt}_2(q_{1,2}, r_{1,1})$. As $\mathit{Slice}$, $\text{LExt}_2$ and $\text{iExt}_2$ are efficient deterministic functions, for a fixed $x_3, y_3, r_{1,1}$, the values of $s_{1,1}$ and $s_{1,2}$ are fixed and can be efficiently computed.

Note that now all the seeds for alternating extraction have been fixed, and we can follow the proof strategy of [CGL16].

Since $r_{1,1} = \text{iExt}_1(w_1, s_{1,1})$, it follows that $w_1$ is restricted to the set $\text{iExt}_1(\cdot, s_{1,1})^{-1}(r_{1,1})$. Further, it follows by Lemma 5 that this is a subspace of size $2^{n_x-d_2}$ irrespective of the value of $s_{1,1}$. As $r_{1,2} = \text{iExt}_3(w_2, s_{1,2})$, similar argument shows that $w_2$ is restricted to a subspace of size $2^{n_x-d_4}$ irrespective of the value of $s_{1,2}$. Further, we note that as $w_1$ and $w_2$ are different blocks, these variables have no correlation.

By repeating this argument for the next two rounds of alternating extraction, it follows that $\overline{q}_{1,1}$ is restricted to a subspace of size $2^{n_q-d_1}$, $w_3$ is restricted to a subspace of size $2^{n_x-d_2}$, $\overline{q}_{1,2}$ is restricted to a subspace of size $2^{n_q-d_3}$, and $w_4$ is restricted to a subspace of size $2^{n_x-d_4}$.

As $(\overline{q_{1,1}}, \overline{q_{1,2}}) = \text{Ext}(v_{[1,4Ct]}, r_1) = \text{iExt}_4(v_1, r_1) \circ \ldots \circ \text{iExt}_4(v_{4Ct}, r_1)$, it follows by an application of Lemma 5 that for any $\overline{q_{1,1}}$, we that $v_{[1,2Ct]}$ is restricted to a subspace of size $2^{2Ct(n_y-d_5)}$. A similar argument shows that for any $\overline{q_{1,2}}$, $v_{[2Ct+1,4Ct]}$ is restricted to a subspace of size $2^{2Ct(n_y-d_5)}$.

Further since $(q_{2,1}, q_{2,2}) = \text{Ext}(v_{[4Ct+1,8t]}, \overline{r_1}) = \text{iExt}_4(v_{4Ct+1}, \overline{r_1}) \circ \ldots \circ \text{iExt}_4(v_{8Ct}, \overline{r_1})$, using a similar argument, we have that for any fixed $q_{2,1}$, we get that $v_{[4Ct+1,6Ct]}$ is restricted to a subspace of size $2^{2Ct(n_y-d_5)}$. A similar argument shows that for any fixed $q_{2,2}$, $v_{[6Ct+1,8Ct]}$ is restricted to a subspace of size $2^{2Ct(n_y-d_5)}$.

It is clear from the arguments that we did not use any specific values of the inputs given to the algorithm $\text{Samp}_1$ (including the value of the bit $b$) to argue about the size of $2\text{ilaExt}^{-1}(q_{2,1}, q_{2,2})$. Also note that each of $v_{[1,4Ct]}, w_{[1,4]}$ is restricted to some subspace. Since $2\text{ilaExt}^{-1}(q_{2,1}, q_{2,2})$ is the cartesian product of these subspaces, it follows that it is a subspace over $\mathbb{F}_2$. Thus the lemma now follows since we can efficiently sample from a given subspace.

□

**Lemma 9.** *( [CGL16]) For any $h \in \{2, \ldots, \ell\}$, any fixing of the variables $\{s_{h,i}, r_{h,i}, \overline{s}_{h,i}, \overline{r}_{h,i} : i \in \{1, 2\}\}$, and any $b \in \{0,1\}$ define the set:*

$$2\text{ilaExt}^{-1}(q_{h+1,1}, q_{h+1,2}) = \{(q_{h,1}, q_{h,2}, v_{[8C(h-1)t+1,8C(h)t]}, w_{[4h-3,4h]}) \in \{0,1\}^{2n_q+8Ctn_y+4n_x} : \\ 2\text{ilaExt}(v_{[8C(h-1)t+1,8Cht]}, w_{[4h-3,4h]}, q_{h,1}, q_{h,2}, b) = (q_{h+1,1}, q_{h+1,2})\}.$$

*There exists an efficient algorithm $\text{Samp}_{h+1}$ that takes as input $q_{h+1,1}, q_{h+1,2}, b, \{s_{h,i}, r_{h,i}, \overline{s}_{h,i}, \overline{r}_{h,i} : i \in \{1,2\}\}$, and samples uniformly from $2\text{ilaExt}^{-1}(q_{h+1,1}, q_{h+1,2})$.*

*Further, $2\text{ilaExt}^{-1}(q_{h+1,1}, q_{h+1,2})$ is a subspace over $\mathbb{F}_2$, and its size does not depend on the inputs to $\text{Samp}_{h+1}$.*

We now adapt the procedure of [CGL16] to work with the changed advice generator.

**Lemma 10.** *For any fixing of the variables $\alpha, \{s_{h,i}, r_{h,i}, \overline{s}_{h,i}, \overline{r}_{h,i} : h \in [\ell], i \in \{1,2\}\} \setminus \{s_{1,1}, s_{1,2}\}$, define the set:*

$$\text{nmExt}_1^{-1}(q_{\ell+1,1}, q_{\ell+1,2}) = \{(x_2, y_2) \in \{0,1\}^{2n_2} : \text{nmExt}_1(x_2, y_2, \alpha) = (q_{\ell+1,1}, q_{\ell+1,2})\}.$$

*There exists an efficient algorithm $\text{Samp}_{nm_1}$ that takes as input $\{s_{h,i}, r_{h,i}, \overline{s}_{h,i}, \overline{r}_{h,i} : h \in [\ell], i \in \{1, 2\}\} \setminus \{s_{1,1}, s_{1,2}\}, \alpha, q_{\ell+1,1}, q_{\ell+1,2}$, and samples uniformly from $\text{nmExt}_1^{-1}(q_{\ell+1,1}, q_{\ell+1,2})$.*

*Further, $\text{nmExt}_1^{-1}(q_{\ell+1,1}, q_{\ell+1,2})$ is a subspace over $\mathbb{F}_2$, and its size does not depend on the inputs to $\text{Samp}_{nm_1}$.*

*Proof.* We have $q_{\ell+1,1}, q_{\ell+1,2}$ and $\{s_{h,i}, r_{h,i}, \overline{s}_{h,i}, \overline{r}_{h,i} : h \in [\ell] \setminus \{1\}, i \in \{1,2\}\} \setminus \{s_{1,1}, s_{1,2}\}$.

For $h = \ell, \ell-1, \ldots, 2$, in reverse order, use $2\text{ilaExt}^{-1}(q_{h+1,1}, q_{h+1,2})$ to sample $(q_{h,1}, q_{h,2}, v_{[8C(h-1)t+1,8C(h)t]}, w_{[4h-3,4h]})$ from lemma 9.

Sample $x_3, y_3$ uniformly and compute $q_{1,1}, q_{1,2} \leftarrow \text{LExt}_2(x_3, y_3)$. Using the value of $q_{1,1}, q_{1,2}$ and $\{r_{1,i}, \overline{s}_{1,i}, \overline{r}_{1,i} : i \in \{1,2\}\}$, sample the value of $w_{[1,4]}, v_{[1,8Ct]}$ using lemma 8.

Since $x_2, y_2$ are concatenations of uniformly chosen $x_3, y_3$ and the blocks sampled above, we can indeed sample efficiently from a distribution uniform on $\{(x_2, y_2) \in \{0,1\}^{2n_2} : \text{nmExt}(x, y, \alpha) = (q_{\ell+1,1}, q_{\ell+1,2})\}$. Further since by lemma 8 and lemma 9, the size of the pre-images of each of the blocks generated do not depend on the inputs (and is also a subspace), it follows that $2\text{nmExt}_1^{-1}(q_{\ell+1,1}, q_{\ell+1,2})$ is a subspace, and its size does not depend on the inputs to $\text{Samp}_{nm_1}$.

□

We now proceed to construct an algorithm to uniformly sample from the pre-image of any output of the function nmExt (Algorithm 6), which will yield the required efficient encoder for the resulting one-many non-malleable codes.

**Lemma 11.** *For any fixing of the variables, $\{s_{h,i}, r_{h,i}, \overline{s}_{h,i}, \overline{r}_{h,i} : h \in [\ell], i \in \{1,2\}\} \setminus \{s_{1,1}, s_{1,2}\}$ and $x_1, y_1, x_3, y_3, \overline{x_1}, \overline{y_1}$, define the set:*

$$\text{nmExt}^{-1}(q_{\ell+1,1}, q_{\ell+1,2}) = \{(x,y) \in \{0,1\}^{2n} : \text{nmExt}(x,y) = (q_{\ell+1,1}, q_{\ell+1,2})\}.$$

*There exists an efficient algorithm* $\text{Samp}_{nm}$ *that takes as input $\{s_{h,i}, r_{h,i}, \overline{s}_{h,i}, \overline{r}_{h,i} : h \in [\ell], i \in \{1,2\}\} \setminus \{s_{1,1}, s_{1,2}\}, x_1, y_1, x_3, y_3, \overline{x_1}, \overline{y_1}, q_{\ell+1,1}, q_{\ell+1,2}$, and samples uniformly from* $\text{nmExt}^{-1}(q_{\ell+1,1}, q_{\ell+1,2})$.

*Further,* $\text{nmExt}^{-1}(q_{\ell+1,1}, q_{\ell+1,2})$ *is a subspace over $\mathbb{F}_2$, and its size does not depend on the inputs to* $\text{Samp}_{nm}$.

*Proof.* Let $T$ be any set of $n_5$ distinct elements of $\mathbb{F}$. We think of $x$ as an element in $\mathbb{F}^{n_4}$, $\mathbb{F} = \mathbb{F}_{2^{\log(n+1)}}$. Let $x = (x^1, \ldots, x^{n_4})$, where each $x^i$ is in $\mathbb{F}$. Recall that the $n_4 \times n$ generator matrix $G$ of the code $\mathbf{RS}$ is the following:

$$G = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_1 & \alpha_2 & \cdots & \alpha_n \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_1^{n_4-1} & \alpha_2^{n_4-1} & \cdots & \alpha_n^{n_4-1} \end{pmatrix}$$

where $\alpha_1, \ldots, \alpha_n$ are distinct non-zero field elements of $\mathbb{F}$.

Let

$$G_T = \begin{pmatrix} 1 & 1 & \cdots & 1 \\ \alpha_{t_1} & \alpha_{t_2} & \cdots & \alpha_{t_{n_5}} \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{t_1}^{n_4-1} & \alpha_{t_2}^{n_4-1} & \cdots & \alpha_{t_{n_5}}^{n_4-1} \end{pmatrix}$$

Since $\overline{x}_1 = \mathbf{RS}(x)_{\{T\}}$, we have the following identity:

$$\begin{pmatrix} x^1 & \cdots & x^{n_4} \end{pmatrix} G_T = \overline{x}_1 \tag{1}$$

Thus, for any fixing of $\overline{x}_1$, the variable $x$ is restricted to a subspace of dimension $(n_4 - n_5)$ over the field $\mathbb{F}$.

Now, let $j \in [n_4]$ be such that $(x^1, \ldots, x^j)$ is the string $(x_1, x_3, w_{[1,4\ell]})$, and $(x^{j+1}, \ldots, x^{n_4})$ is the string $w_{[4\ell+1,8\ell]}$. Clearly, $(n_4 - j)\log n = 4\ell n_x$, and thus by our choice of parameters it follows that $j = n_4 - \frac{4\ell n_x}{\log n} = \frac{n_4}{2} + \frac{n_1+n_6}{log(n+1)} < \frac{n_4}{2} + \frac{n}{6log(n+1)} < \frac{2n_4}{3} < n_4 - n_5$.

We further note since any $n_5 \times n_5$ sub-matrix of $G_T$ has full rank (since it is the Vandermonde's matrix), it follows by the rank-nullity thorem that any $j \times n_5$ sub-matrix of $G_T$ has null space of dimension exactly $j - n_5$. Thus for any $\lambda \in \mathbb{F}^{n_5}$, the equation:

$$\begin{pmatrix} x^{j+1} & \cdots & x^{n_4} \end{pmatrix} \begin{pmatrix} \alpha_{t_1}^j & \alpha_{t_2}^j & \cdots & \alpha_{t_{n_5}}^j \\ \vdots & \vdots & \ddots & \vdots \\ \alpha_{t_1}^{n_4-1} & \alpha_{t_2}^{n_4-1} & \cdots & \alpha_{t_{n_5}}^{n_4-1} \end{pmatrix} = \overline{x}_1 + \lambda \tag{2}$$

has exactly $|\mathbb{F}|^{(j-n_5)}$ solution.

Thus, for any fixing of the variables, $x^1, \ldots, x^j$, equation (1) has exactly $|\mathbb{F}|^{j-n_5}$ solutions. In other words, for any fixing of $x_1, x_3, w_{[1,4\ell]}, \overline{x}_1$, the variable $w_{[4\ell+1,8\ell]}$ is restricted to a subspace, and the size of the subspace does not depend on the fixing of $x_1, x_3, w_{[1,4\ell]}, \overline{x}_1$. We note that the above argument works for any set $T$ which has $n_5$ distinct elements of $\mathbb{F}$. Using, a similar argument, we can show that for any fixing of $y_1, y_3, v_{[1,8Ct\ell]}, \overline{y}_1$, the variable $v_{[8Ct\ell+1,16Ct\ell]}$ is restricted to a subspace, and the size of the subspace does not depend on the fixing of $y_1, y_3, v_{[1,8Ct\ell]}, \overline{y}_1$.

Now consider any fixing of the variables $\{s_{h,i}, r_{h,i}, \overline{s}_{h,i}, \overline{r}_{h,i} : h \in [\ell], i \in \{1,2\}\} \setminus \{s_{1,1}, s_{1,2}\}, x_1, y_1, x_3, y_3, \overline{x_1}, \overline{y_1}$. This fixes $\nu (= \nu_1, \nu_2) \leftarrow \text{LExt}_1(x_1, y_1)$ and $z \leftarrow v \circ \overline{x_1} \circ \overline{y_1}$ as well. As proved in the lemma 10, we can efficiently sample the variables $w_{[1,4\ell]}, v_{[1,8Ct\ell]}$. Let $T_1 = \text{Samp}(\nu_1) = \{t_1, \ldots, t_{n_5}\}$ be the set of $n_5$ distinct elements of $\mathbb{F}$. By the above argument, which works for any set $T$ of size $n_5$, we get that for the constructed $T_1$, the variables $v_{[4\ell+1,8\ell]}$ and $w_{[8Ct\ell+1,16Ct\ell]}$ now lie in a subspace, and hence we can efficiently sample these variables as well. Thus we have an efficient procedure $\text{Samp}_{nm}$ for uniformly sampling $(x, y)$ from the set $\text{nmExt}^{-1}(q_{\ell+1,1}, q_{\ell+1,2})$ .

It also follows by lemma 10, that the total size of the pre-image of the variables $x_3, w_{[1,4\ell]}, y_3, v_{[1,8Ct\ell]}$ does not depend on $z$ or the variables $\{s_{h,i}, r_{h,i}, \overline{s}_{h,i}, \overline{r}_{h,i} : h \in [\ell], i \in \{1,2\}\} \setminus \{s_{1,1}, s_{1,2}\}$. Further, for any fixing of $x_3, w_{[1,4\ell]}, v_{[1,8Ct\ell]}, y_3, z$, as argued above, the variables $v_{[4\ell+1,8\ell]}$ and $w_{[8Ct\ell+1,16Ct\ell]}$ now lie in a subspace, whose size does not depend on the fixed variables. Thus, overall the size of the total pre-image of $x, y$ does not depend on the inputs to $\text{Samp}_{nm}$.

□

We now state the main result of this section.

**Theorem 5.** *There exists an efficient procedure that given an input* $(q_{\ell+1,1}, q_{\ell+1,2}) \in \{0,1\}^{n_q} \times \{0,1\}^{n_q}$*, samples uniformly from the set* $\{(x, y) : \text{NMExt}(x, y) = (q_{\ell+1,1}, q_{\ell+1,2})\}$.

*Proof.* We use the following simple strategy.

1. Uniformly sample the variables $\{s_{h,i}, r_{h,i}, \overline{s}_{h,i}, \overline{r}_{h,i} : h \in [\ell], i \in \{1,2\}\} \setminus \{s_{1,1}, s_{1,2}\}$ and $x_1, y_1, x_3, y_3, \overline{x_1}, \overline{y_1}$.
2. Use the variables sampled in step (1) as input to the algorithm $\text{Samp}_{nm}$ to sample $(x, y)$.

The correctness of this procedure follows directly from lemma 11, since it was proved that for any fixing of the variables of Step 1, the size of pre-image of NMExt is the same.

□

## 6 Non-Malleable Message Transmission

As an application of non-malleable secret sharing schemes, we initiate the study of non-malleable message transmission. While the existing works on message transmission have been primarily concerned with ensuring reliability and secrecy of the message [DDWY93, SNR04, WD08, KS09, KKVS18], the new model introduced here tries to obtain non-malleability against much more threatening adversaries which can corrupt all the nodes of the network (apart from the sender and the receiver). We formally define our setting below.

### General Network Model and Definitions

We model our communication network as an undirected graph $G(V, E)$, where each edge is private, authentic and a (bi-directional) reliable channel. In the network, adversary can only corrupt the nodes, and not the edges. This is without loss of generality, as an edge can be further split into a node with one incoming edge and outgoing edge. There is a designated sender $S$ and receiver $R$. We assume that there is no edge from sender to receiver, as otherwise the problem becomes trivial.

Every player on the network, knows the topology of the network. We also assume that every player including the adversary knows the full protocol specifications. The faults in the network is characterized by a central (fictitious) adversary who can byzantinely corrupt all the nodes of the network, other than the sender $S$ and receiver $R$. We also assume that the adversary cannot communicate using a hidden network, as otherwise, the adversary controlling all nodes can simply pool all the ongoing transmission in the network at a single node (using hidden links), and break all possible security. We slightly relax this requirement by allowing the adversary to add a bounded number of hidden links of its choice to the network.

**Definition 16.** *(**Byzantine fault**) A node $P$ is said to be Byzantine corrupted if the adversary fully controls the actions of $P$. The adversary will have full access to the computation and communication of $P$ and can force $P$ to deviate from the protocol in any arbitrary manner.*

**Definition 17.** *(**Protocol** $\pi(\mathbf{G})$) Given a network, represented by an graph $G = \langle V, E\rangle$, consider each node $v \in V$ as an interactive Turing Machine. A protocol $\pi(G)$ is the specification of the code running on each of these nodes. Specifically, consider $\pi(G)$ as a set of turning machines $\{\pi_v(G) : v \in G\}$, where $\pi_v(G)$ specifies the behavior of the node $v$ during the execution of the protocol.*

**Definition 18.** *(**Secure Non-Malleable Message Transmission**) Given a network, represented by an graph $G = \langle V, E\rangle$, with a designated Sender $S$ and receiver $R$. Let $\mathcal{C}$ be some family of corruptions. Let $\mathcal{M}$ be the message space from which the sender wishes to transmit a message $m$ to the receiver. The protocol $\pi(G, S, R)$ is said to be a $\epsilon$-secure non-malleable message transmission protocol wrt Adv, if the following properties hold.*

- ***Correctness** : Suppose the sender wishes to transmit message $m \in \mathcal{M}$, and all the nodes follow that protocol $\pi(G, S, R)$ honestly ( node $v$ executes $\pi_v(G, S, R)$), then at the end of the protocol the receiver $R$ outputs the transmitted message $m$ with probability 1.*

- ***Statistical Privacy** : During the execution of the protocol, any node $v \in V$, other than the receiver $R$, should not learn any information about the message $m$. More formally, for all $m_1, m_2 \in \mathcal{M}$, and any node $v(\neq R) \in V$, any distinguisher $D$ which outputs in $\{0, 1\}$, the following holds :*

$$|Pr_{view_v \leftarrow \pi_v^{view}(G,S,R)(m_1)}[D(view_v) = 1] - Pr_{view_v \leftarrow \pi_v^{view}(G,S,R)(m_2)}[D(view_v) = 1]| \leq \epsilon$$

  *where $\pi_v^{view}$ denotes the view of the node $v$ during the execution of the protocol, and $\pi_v^{view}(G, S, R)(m_1)$ denotes the view of $v$ in the protocol with senders private input as $m_1$.*
- ***Statistical Non Malleability** : For each $c \in \mathcal{C}$, for each $m \in \mathcal{M}$, define $\mathbf{Tamper_m^f}$ as the output of the receiver $R$ when all the nodes behave according to $c$ instead of following protocol $\pi(G, S, R)$, that is each node $v \in V$ executes $c_v$ rather than $\pi_v(G, S, R)$. For each $f \in \mathcal{C}$, there exists a distribution $\mathbf{D^f}$ (corresponding to simulator) over $\mathcal{M} \cup \{same^*, \perp\}$ such that, for all $m \in \mathcal{M}$, we have that the statistical distance between $\mathbf{Tamper_m^f}$ and*

$$\mathbf{Sim_m^f} = \left\{ \begin{array}{c} \widetilde{m} \leftarrow \mathbf{D^f} \\ Output : m \text{ } if \widetilde{m} = same^*, or \text{ } \widetilde{m}, otherwise \end{array} \right\}$$

  *is at most $\epsilon$. Additionally, $\mathbf{D^f}$ should be efficiently samplable given oracle access to $\mathbf{f}$(.).*

**Definition 19.** *(**Path and Induced Subgraph**). For an graph $G = \langle V, E\rangle$, for any $u, w \in V$, a sequence $u \to v_1 \to v_2 \to \ldots \to v_i \to w$ is called a **path** from $u$ to $v$ of length $i + 1$ if $(u, v_1) \in E \wedge (v_i, w) \in E$ and for all $j \in [i - 1]$, $(v_i, v_{i+1}) \in E$. We say that a node $v$ is reachable by $u$ if there exists a path $u \to v$ in graph $G$.*

*We define* $\mathbf{G}[\mathbf{V}']$ *as the induced subgraph of* $G$ *which has vertex set* $V'$ *and edge set* $E' \leftarrow \{(u, v) \in E : u \in V' \wedge v \in V'\}$.

**Definition 20.** ***(Collection of*** $k$ ***Non-Malleable Paths)*** *Consider a network represented by an graph* $G = \langle V, E\rangle$, *with a designated sender* $S$ *and receiver* $R$. *Let there be* $k$ *paths from sender* $S$ *to receiver* $R$. *The set of these paths is called a* ***collection of*** $k$ ***non-malleable paths*** *from* $S$ *to* $R$ *if any node in the induced graph* $G[V \setminus \{S, R\}]$*(graph left after removing sender* $S$ *and receiver* $R$ *from* $G$*) is reachable by nodes present on at most one of these* $k$ *paths.*

We note that it is easy to find such a collection of paths for graphs by just prunning away all the vertices which do not have a path to receiver or are unreachable by the sender from the graph. We further remove sender and receiver, and run connected components algorithm [Cor09], using which we obtain the required paths.

**Corruption family $\mathcal{C}_t$**

$\mathcal{C}_t$ represents the family of corruptions in which all the nodes of the network (except $S$ and $R$) can be byzantine corrupted by the adversary. The adversary has the power to add $t$ hidden (bidirectional) links in the network. All the corrupted nodes can deviate arbitrarily from the protocol and communicate with each other using less than $t$ hidden links in addition to the original edges $E$ of the network. Formally, for a network represented by graph $G = \langle V, E\rangle$, for any protocol $\pi(G, S, R)$, the adversary specifies a set $E_t$ containing less than $t$ new hidden links, and specifies a protocol $c = \{c_v : v \in V\} \in \mathcal{C}_t$, where $c_S = \pi_S(G, S, R)$ and $c_R = \pi_R(G, S, R)$. During the corrupted execution, the protocol $c$ is executed.

**Lemma 12.** *In any network represented by an graph* $G = \langle V, E\rangle$, *with a designated sender* $S$ *and receiver* $R$, *if there exists a collection of* $t$ *non-malleable paths from* $S$ *to* $R$, *then non-malleable secure message transmission protocol wrt* $\mathcal{C}_k$ *is possible where* $k \leftarrow 2\lceil t/2\rceil - 3$.[12]

*Proof.* We begin with the description of the protocol. Let $\mathbf{JNMShare}_\mathbf{t}^\mathbf{t}$ be a $(t, t, \epsilon)$-non-malleable secret sharing sharing scheme wrt. $\mathcal{F}_{t,n}^{joint}$ as constructed in the proof of theorem 2. Let $p_1, p_2, \ldots, p_t$ be the $t$ non-malleable paths from $S$ to $R$. Sender computes $\{Share_i : i \in [t]\} \leftarrow \mathbf{JNMShare}_\mathbf{t}^\mathbf{t}(m)$ using the non-malleable secret sharing scheme. Sender sends $Share_i$ using $p_i$ for each $i \in [t]$. Receiver receives $\widetilde{Share_i}$ from each of the paths $\{p_i : i \in [t]\}$, and computes $\widetilde{m} \leftarrow \mathbf{JNMRec}_\mathbf{t}^\mathbf{t}(\{\widetilde{share_i} : i \in [t]\})$ and outputs $\widetilde{m}$.

**Correctness** : Trivially follows from the correctness of secret-sharing scheme.

**Statistical Privacy** : Notice that by adding $k$ hidden links in the network, the adversary can merge at most $k + 1$ non-malleable paths, in other words, it can pool together at most $k + 1$ shares. Secrecy of our protocol follows from the statistical secrecy of the underlying t-out-of-n non-malleable secret-sharing scheme (as $k + 1 < t$).

**Statistical Non Malleabilty** (Proof Sketch): As there are $t$ non-malleable paths from $S$ to $R$, an addition of a hidden link can only decrease the number of non-malleable paths by at most one. Another way to see this would be to see in terms of connected components in the induced graph (after removing $S$ and $R$), and new edge can possible merge only two disjoint components. Therefore, by adding at most $k$ edges, the adversary can merge at most $k+1$ connected components

[12]Optimal value is $t - 2$, so for the odd values of $t$, our construction is optimal, for even values, it is one less than the optimal.

into a single one (or more components). This merging gives adversary the power to jointly tamper subsets of shares in the protocol. Notice that if the adversary add $2\lceil t/2 \rceil - 3$ links, and we will always be able to create two non-empty subsets containing different number of non-malleable paths. This allows us to use the non-malleability of the underlying scheme to show non-malleability of this protocol.

□

## References

[ADKO15] Divesh Aggarwal, Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Leakage-resilient non-malleable codes. In *Twelfth IACR Theory of Cryptography Conference (TCC 2015)*, 2015.

[ADL14] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*, pages 774–783. ACM, 2014.

[Bei] Amos Beimel. *Secure schemes for secret sharing and key distribution, PhD Thesis.*

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

[Bla79] G. R. Blakley. Safeguarding cryptographic keys. In *AFIPS National Computer Conference (NCC '79)*, pages 313–317, Los Alamitos, CA, USA, 1979. IEEE Computer Society.

[BOGW88] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*, pages 1–10. ACM, 1988.

[CDF$^+$08] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 471–488. Springer, 2008.

[CDTV16] Sandro Coretti, Yevgeniy Dodis, Björn Tackmann, and Daniele Venturi. Non-malleable encryption: simpler, shorter, stronger. In *Theory of Cryptography Conference*, pages 306–335. Springer, 2016.

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

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

[CGL16] Eshan Chattopadhyay, Vipul Goyal, and Xin Li. Non-malleable extractors and codes, with their many tampered extensions. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 285–298, 2016.

[Coh16] Gil Cohen. Non-malleable extractors–new tools and improved constructions. In *31st Conference on Computational Complexity*, 2016.

[Cor09] Thomas H Cormen. *Introduction to algorithms*. MIT press, 2009.

[DDN91] Danny Dolev, Cynthia Dwork, and Moni Naor. Non-malleable cryptography (extended abstract). In *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing, May 5-8, 1991, New Orleans, Louisiana, USA*, pages 542–552, 1991.

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

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

[DLWW11] Yevgeniy Dodis, Allison Lewko, Brent Waters, and Daniel Wichs. Storing secrets on continually leaky devices. In *Proceedings of the 2011 IEEE 52Nd Annual Symposium on Foundations of Computer Science*, FOCS '11, pages 688–697, Washington, DC, USA, 2011. IEEE Computer Society.

[DORS08] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. *SIAM Journal on Computing*, 38:97–139, 2008.

[DP07] Stefan Dziembowski and Krzysztof Pietrzak. Intrusion-resilient secret sharing. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*, pages 227–237. IEEE, 2007.

[DP08] Stefan Dziembowski and Krzysztof Pietrzak. Leakage-resilient cryptography. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*, pages 293–302. IEEE, 2008.

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, pages 434–452, 2010.

[GJK15] Vipul Goyal, Aayush Jain, and Dakshita Khurana. Non-malleable multi-prover interactive proofs and witness signatures. Cryptology ePrint Archive, Report 2015/1095, 2015. `http://eprint.iacr.org/2015/1095`.

[GKP$^+$18] Vipul Goyal, Ashutosh Kumar, Sunoo Park, Silas Richelson, and Akshayaram Srinivasan. New constructions of non-malleable commitments. Manuscript, 2018. in submission.

[GPR16] Vipul Goyal, Omkant Pandey, and Silas Richelson. Textbook non-malleable commitments. In *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, pages 1128–1141, 2016.

[GR15] Shafi Goldwasser and Guy N Rothblum. How to compute in the presence of leakage. *SIAM Journal on Computing*, 44(5):1480–1549, 2015.

[KKVS18] Ravi Kishore, Ashutosh Kumar, Chiranjeevi Vanarasa, and Kannan Srinathan. On the price of proactivizing round-optimal perfectly secret message transmission. *IEEE Transactions on Information Theory*, 64(2):1404–1422, Feb 2018.

[KS09] Kaoru Kurosawa and Kazuhiro Suzuki. Truly efficient-round perfectly secure message transmission scheme. *IEEE Transactions on Information Theory*, 55(11):5223–5232, 2009.

[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.

[LL12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In *Proceedings of the 32Nd Annual Cryptology Conference on Advances in Cryptology — CRYPTO 2012 - Volume 7417*, pages 517–532, New York, NY, USA, 2012. Springer-Verlag New York, Inc.

[LLTT05] Chia-Jung Lee, Chi-Jen Lu, Shi-Chun Tsai, and Wen-Guey Tzeng. Extracting randomness from multiple independent sources. *IEEE Transactions on Information Theory*, 51(6):2224–2227, 2005.

[MS81] Robert J. McEliece and Dilip V. Sarwate. On sharing secrets and reed-solomon codes. *Communications of the ACM*, 24(9):583–584, 1981.

[Raz05] Ran Raz. Extractors with weak random seeds. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 11–20. ACM, 2005.

[RBO89] T. Rabin and M. Ben-Or. Verifiable secret sharing and multiparty protocols with honest majority. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC '89, pages 73–85, New York, NY, USA, 1989. ACM.

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

[SNR04] K. Srinathan, Arvind Narayanan, and C. Pandu Rangan. Optimal perfectly secure message transmission. In *In Proc. of Advances in Cryptology: CRYPTO 2004, LNCS 3152*, pages 545–561. Springer-Verlag New York, Inc., 2004.

[Vad04] Salil P. Vadhan. Constructing locally computable extractors and cryptosystems in the bounded-storage model. *J. Cryptology*, 17(1):43–77, 2004.

[WD08] Yongge Wang and Yvo Desmedt. Perfectly secure message transmission revisited. *IEEE Transactions on Information Theory*, 54(6):2582–2595, 2008.

## A 2-out-of-n Secret Sharing Schemes

### A.1 2-out-of-n Leakage-Resilient Secret Sharing Schemes

In this subsection, we use a $(2, 2, \epsilon)$-secret sharing scheme that is leakage-resilient w.r.t. $\mathcal{L}_{\mu}^{split}$to construct a $(2, n, \epsilon')$-secret sharing scheme that is leakage-resilient w.r.t. $\mathcal{L}_{\mu}^{split}$(see 3 for definition). In the next subsection, we give an efficient construction of a $(2, 2, \epsilon)$-secret sharing scheme that is leakage-resilient w.r.t. to $\mathcal{L}_{\mu}^{split}$.

**Theorem 6.** *Assume that for any $\epsilon > 0$, let $(\mathbf{Share}, \mathbf{Rec})$, be any efficient $(2, 2, \epsilon)$-secret sharing scheme that is $\epsilon$-leakage-resilient w.r.t. $\mathcal{L}_{\mu}^{split}$and encodes an element of $\mathcal{M}$ into two elements of $\mathbb{F}$.*

*Then for any number of parties $n$, there exists an efficient $(2, n, n^2\epsilon)$-secret sharing scheme that is $(n^2\epsilon)$-leakage-resilient w.r.t. $\mathcal{L}_{\mu}^{split}$. The resulting scheme, $(\mathbf{LRShare_n^2}, \mathbf{LRRec_n^2})$, shares an element of $\mathcal{M}$ into $n$ shares, where each share is an element of the $(\mathbb{F})^n$.*

*Proof.* The efficient construction of $(\mathbf{LRShare_n^2}, \mathbf{LRRec_n^2})$ is given below:

- **Sharing function** $(\mathbf{LRShare_n^2})$.
  On input a secret message $m \in \mathcal{M}$, for each $\{i, j\} \in \mathcal{A}^{min}$ such that $i < j$, share $m$ using the sharing procedure of underlying leakage-resilient scheme, as $u_i^j, u_j^i \leftarrow \mathbf{Share}(m)$. For each $i \in [n]$, let $u_i^i \leftarrow 0$. Finally, for each $i \in [n]$, construct $share_i$ as $u_i^1, u_i^2, \ldots, u_i^n$.

- **Reconstruction function** $(\mathbf{LRRec_n^2})$.
  Let $i$ and $j$ be the first two indices of $T$ such that $i < j$. Parse $share_i$ as $u_i^1, u_i^2, \ldots, u_i^n$ and parse $share_j$ as $u_j^1, u_j^2, \ldots, u_j^n$. Using the reconstruction procedure of underlying leakage resilient scheme compute, $m \leftarrow \mathbf{Rec}(u_i^j, u_j^i)$. Output $m$.

**Correctness, Efficiency and Statistical Privacy**: Correctness and efficiency trivially follows from construction. Statistical privacy follows as a corollary to leakage-resilience proved below.

**Leakage-Resilience**: We prove leakage-resilience by contradiction. Assume, that there exists a leakage function $\mathbf{f} = (\mathbf{f_1}, \mathbf{f_2}, \ldots, \mathbf{f_n}) \in \mathcal{L}_{\mu}^{split}$, a pair of message $a, b \in \mathcal{M}$, and a distinguisher $\mathbf{D}$ such that

$$|Pr_{shares \leftarrow \mathbf{LRShare_n^2}(a)}[f(shares) = 1] - Pr_{shares \leftarrow \mathbf{LRShare_n^2}(b)}[f(shares) = 1]| > n^2\epsilon$$

then we create a leakage function $\mathbf{g} = (gp1, gp2) \in \mathcal{L}_{\mu}^{split}$, and a distinguisher $\mathbf{D_1}$ which violates the leakage-resilient of $\epsilon$-secret sharing scheme $(\mathbf{Share}, \mathbf{Rec})$ w.r.t. to secrets $a$ and $b$. We achieve this using a hybrid argument. Let $share_i$ be of the form $au_i^1, au_i^2, \ldots, au_i^n$ when secret $a$ is encoded by the sharing procedure $\mathbf{LRShare_n^2}$. Similarly, let $share_i$ be of the form $bu_i^1, bu_i^2, \ldots, bu_i^n$ when secret $b$ is encoded.

0,0. $\mathbf{Hybrid_{0,0}}$ : for each $i \in [n]$, $share_i$ is of the form $au_i^1, au_i^2, \ldots, au_i^n$. This is similar to the set of $n$ shares obtained on running the $\mathbf{LRShare_n^2}$ on input $a$. Output $(\mathbf{f_1}(share_1), \ldots, \mathbf{f_n}(share_n))$.

1,1. $\mathbf{Hybrid_{1,1}}$ : Similar to previous hybrid $\mathbf{Hybrid_{0,0}}$, with $au_1^1$ replaced by $bu_1^1$. $share_1$ is of the form $bu_1^1, au_1^2, \ldots, au_1^n$. Output $(\mathbf{f_1}(share_1), \ldots, \mathbf{f_n}(share_n))$.

$\vdots$

1,n. $\mathbf{Hybrid_{1,n}}$ : Similar to previous hybrid $\mathbf{Hybrid_{1,n-1}}$, with $au_1^n$ and $au_n^1$ replaced by $bu_1^n$ and $bu_n^1$ respectively. $share_1$ is of the form $bu_1^1, bu_1^2, \ldots, bu_1^n$. $share_n$ is of the form $bu_n^1, au_n^2, \ldots, au_n^n$. Note that the distribution of $share_1$ is similar to the $share_1$ obtained on running the $\mathbf{LRShare_n^2}$ on input $b$. Output $(\mathbf{f_1}(share_1), \ldots, \mathbf{f_n}(share_n))$.

2,2. $\mathbf{Hybrid_{2,2}}$ : Similar to previous hybrid $\mathbf{Hybrid_{1,n}}$, with $au_2^2$ replaced by $bu_2^2$. $share_2$ is of the form $bu_2^1, bu_2^2, au_2^3 \ldots, au_2^n$. Output $(\mathbf{f_1}(share_1), \ldots, \mathbf{f_n}(share_n))$.

$\vdots$

n,n. $\mathbf{Hybrid_{n,n}}$ : Similar to previous hybrid $\mathbf{Hybrid_{n,n-1}}$,with $au_n^n$ replaced by $bu_n^n$. $share_n$ is of the form $bu_n^1, bu_n^2, \ldots, bu_n^n$. In this hybrid, the distribution of $n$ shares is identical to the one obtained on running the $\mathbf{LRShare_n^2}$ on input $b$. Output $(\mathbf{f_1}(share_1), \ldots, \mathbf{f_n}(share_n))$.

The number of hybrids is $n(n-1)/2 + 1$. As the distinguisher $\mathbf{D}$ can distinguish between $\mathbf{Hybrid_{0,0}}$ and $\mathbf{Hybrid_{n,n}}$ with probability greater than $n^2\epsilon$, by averaging argument we know that there exists a $j$ and $k$ such $0 \leq j \leq n$ and $j < k \leq n$, such that $\mathbf{D}$ can distinguish $\mathbf{Hybrid_{j,k}}$ and the next hybrid $\mathbf{Hybrid_{j,k+1}}$ with probability greater than $n^2\epsilon/(n(n-1)) \geq \epsilon$ (as $n \geq 2$). Note that we can assume that such a $j$ cannot be equal to $k$, as both $au_j^k$ and $bu_j^k$ will be equal to 0 in such a case. We give a reduction by creating explicit leak function $\mathbf{g} = (\mathbf{g_1}, \mathbf{g_2}) \in \mathcal{L}_\mu^{split}$ and dinstinguisher $\mathbf{D_1}$.

1. **Initial setup** : Sample $share_i$ for all $i \in [n]$ according to $\mathbf{Hybrid_{j,k}}$. For each $i \in [n] \setminus \{j, k\}$ compute $\mathbf{f_i}(share_i)$.
2. **Leakage function $\mathbf{g_1}(l)$** : On input $l$ replace $au_j^k$ with $l$ in $share_j$, compute $\mathbf{f_j}$ on the modified $share_j$ and output $\mathbf{f_j}(share_j)$.
3. **Leakage function $\mathbf{g_2}(r)$**: On input $r$ replace $au_k^j$ with $r$ in $share_k$, compute $\mathbf{f_k}$ on modified $share_k$ and output $\mathbf{f_k}(share_k)$.
4. **Distinguisher $\mathbf{D_1}(g_1(l), g_2(r))$** : On input $g_1(l), g_2(r)$ replace $\mathbf{f_j}(share_j)$ and $\mathbf{f_k}(share_k)$ with $g_1(l)$ and $g_2(r)$ to obtain $(\mathbf{f_1}(share_1), \ldots, \mathbf{f_n}(share_n))$. Invoke the distinguisher $\mathbf{D}$ with the modified $(\mathbf{f_1}(share_1), \ldots, \mathbf{f_n}(share_n))$ and output its output.

Notice, in case $l, r$ was encoding $a$, then $(\mathbf{f_1}(share_1), \ldots, \mathbf{f_n}(share_n))$ will be distributed according to $\mathbf{Hybrid_{j,k}}$. Otherwise, $(\mathbf{f_1}(share_1), \ldots, \mathbf{f_n}(share_n))$ will be distributed according to $\mathbf{Hybrid_{j,k+1}}$. Therefore the success probablity of $\mathbf{D_1}$ will be equal to the advantage of $\mathbf{D}$ in distinguishing $\mathbf{Hybrid_{j,k}}$ and $\mathbf{Hybrid_{j,k+1}}$. As the latter probablity is greater than $\epsilon$, we have arrived at a contradiction to $\epsilon$-leakage-resilience of the secret sharing scheme $(\mathbf{Share}, \mathbf{Rec})$.

□

**2-out-of-2 Leakage-Resilient Secret-Sharing**

In this section we construct efficient construction of $(\mathbf{Share}, \mathbf{Rec})$, the secret sharing scheme realizing access structure $\mathcal{A}_2^2$ that is leakage-resilient w.r.t. to $\mathcal{L}_\mu^{split}$. We use one of the primitives of Goldwasser and Rothblum ( [GR15]) which they used to construct a 2-out-of-2 secret-sharing scheme that allowed $\lambda$ bits of leakage from each of shares. Using the strongness property of two-source extractors, we give an analysis which allows one of the sides to be fully leaked. There are lot of interesting works on constructing leakage-resilient secret-sharing schemes, which handle much more advanced form of leakages. ( [DP07, DP08, DLWW11]).

We recall Hadamard matrix based folklore construction of two source extractors before giving our construction.

**Lemma 13.** *( [CG88]) Let* IP *be the inner product function. Let $L$ and $R$ be independent random variables over $\mathbb{F}_p^n$. If*

$$H_\infty(L) + H_\infty(R) \geq (n+1)\, log\, p + 2\, log(\frac{1}{\epsilon})$$

*then*

$$|(L, \mathrm{IP}(L, R)) - (L, U_{\mathbb{F}_p})| \leq \epsilon \; and \; |(R, \mathrm{IP}(L, R)) - (R, U_{\mathbb{F}_p})| \leq \epsilon$$

**Lemma 14.** *For $\epsilon > 0$, any leak function $(\mathbf{f}, \mathbf{g}) \in \mathcal{L}_\mu^{split}$, there exists an efficient $(2, 2, \epsilon)$-secret sharing scheme that is $(\epsilon)$-leakage-resilient w.r.t. $\mathcal{L}_\mu^{split}$. The resulting scheme, $(\mathbf{Share}, \mathbf{Rec})$ , shares a $m$ bit secret into two $n$ bit shares, where $n = m + \mu + 3\log\frac{4}{\epsilon}$.*

*Proof.* Let $n = m + \mu + 3\log\frac{4}{\epsilon}$. Let $\mathbf{IP} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ represent the inner product function. The leakage-resilient scheme is defined as :

1. The **sharing function Share**, on input a secret $s$, samples a random $l, r$ such that $\mathbf{IP}(l, r) = s$, and outputs $(l, r)$.
2. The **reconstruction function Rec**, parses the input as two $n$ bit strings $l$ and $r$, computes $s \leftarrow \mathbf{IP}(l, r)$ and outputs $s$.

The correctness follows trivially. The statistical privacy can be obtained on observing that inner product is strong 2 source extractor, and the output remains statistically hidden even if one of the two sources is given (lemma 13), or from the leakage-resilience proved below.

Consider any leakage functions $\mathbf{f}, \mathbf{g} \in \mathcal{L}_{\mu}^{split}$. Without loss of generality assume these functions are deterministic, and as the sharing scheme is symmetric, assume $\mathbf{f}$ outputs $\mu$ bits of information about $l$, while $\mathbf{g}$ outputs the whole share $r$. Using lemma 13, we get that $\mathbf{IP}$ is a $(m + 2log\frac{4}{\epsilon}, \frac{\epsilon}{4})$ strong two-source extractor. We note that if $R$ is chosen uniformly at random (which is true in our construction), we can treat $\mathbf{IP}$ as a strong seeded extractor. By lemma 3, we know that a $\left(m + 2log\frac{4}{\epsilon}, \frac{\epsilon}{4}\right)$ seeded extractor is also a $\left(m + 3log\frac{4}{\epsilon}, \frac{\epsilon}{2}\right)$ average-case seeded extractor.

We know $H_\infty(L) = n$, as $l$ is a random $n$ bit string. We know by applying lemma 2, that $\tilde{H}_\infty(L|f(L)) \geq H_\infty(L) - \mu = n - \mu$, as the output of $f$ is at most $\mu$ bits. To be able to use the average case seeded extractor property of $\mathbf{IP}$, we need $n - \mu \geq m + 3log\frac{4}{\epsilon}$ (this is true for our setting of $n$). Therefore, we get that output of $\mathbf{IP}(L, R)$ is $\frac{\epsilon}{2}$-close to uniform, even when $R$ and $f(L)$ is given. Therefore, we can say that for any two messages $a$ and $b$, no distinguisher can distinguish with probability more than $\epsilon$.

□

## A.2 2-out-of-n Non-Malleable Secret Sharing Schemes

**Theorem 7.** *Assume that for any number of parties $n$, any $\epsilon \geq 0$, there exists an efficient coding scheme, $(\mathbf{NMEnc}, \mathbf{NMDec})$, that is $\epsilon$-non-malleable wrt $\mathcal{F}_2^{split}$ that encodes an element of $\mathcal{M}$ into two elements of $\mathbb{F}$.*

*Then there exists an efficient $(n, 4n\epsilon)$-secret sharing scheme realizing paired access structure $\mathcal{A}$ that is $(4n\epsilon)$-non-malleable w.r.t $\mathcal{F}_n^{split}$. The resulting scheme, $(\mathbf{NMShare_n^2}, \mathbf{NMRec_n^2})$, shares an element of $\mathcal{M}$ into $n$ shares where each share is an element of $(\mathbb{F})^n$.*

*Proof.* The construction of the scheme $(\mathbf{NMShare_n^2}, \mathbf{NMRec_n^2})$ is given below :

1. **Sharing Function**$(\mathbf{NMShare_n^2})$
   For each $\{i, j\} \in [n]$ such that $i < j$, encode $m$ using the encoding procedure of non-malleable code to obtain $v_i^j, v_j^i \leftarrow \mathbf{NMEnc(m)}$. For each $\{i, j\} \notin [n]$ such that $i \leq j$, let $v_i^j, v_j^i \leftarrow 0$. For each $i \in [n]$, construct the $i^{th}$ share of the scheme as follows : $share_i = (v_i^1, \ldots, v_i^n)$. Output $(share_1, \ldots, share_n)$

2. **Reconstruction Function**$(\mathbf{NMRec_n^2})$
   Let $i$ and $j$ be the first two indices of $T$ such that $i < j$. On input the shares $\otimes_{i \in T} share_i$, for each $i \in T$, parse $share_i$ as $v_i^1, \ldots, v_i^n$. Use the decoding procedure $NMDec$ to obtain the hidden secret $m \leftarrow \mathbf{NMDec}(v_i^j, v_j^i)$. Output $m$.

Correctness trivially follows from the construction. Statistical privacy follows from a simple hybrid argument that uses the fact that every 2 split-state non-malleable code is also a 2-out-of-2 secret-sharing scheme (see lemma 1).

**Statistical Non Malleability** : Using the tampering functions $\{\mathbf{f_i} : i \in [n]\}$ belonging to $\mathcal{F}_n^{split}$, we give a reduction, by creating explicit function $(\mathbf{F}, \mathbf{G}) \in \mathcal{F}_2^{split}$ that tampers with the two shares of the split-state non-malleable code. Let $T$ be an authorized set containing two elements $i$

and $j$ such that $i < j$. The reduction giving explicit $(\mathbf{F}, \mathbf{G}) \in \mathcal{F}_2^{split}$ is described below.

1. **(Initial Setup)** : Randomly choose a message $m_\$ \in \mathcal{M}$, and run the sharing function $\mathbf{NMShare_n^2}$ to obtain temporary shares. That is, $(tShare_1, \ldots, tShare_n) \leftarrow \mathbf{NMShare_n^2}(m_\$)$. For each $i \in [n]$, parse $tShare_i$ as $tv_i^1, \ldots, tv_i^n$.
2. The **tampering function** $\mathbf{F}(l)$ is defined as follows : On input $l \in \mathbb{F}$, replace $tv_i^j$ by $l$ in $tShare_i$ to obtain $share_i$. Run $f_i$ on $share_i$ to obtain $\widetilde{share_i}$. Parse $\widetilde{share_i}$ as $\widetilde{v_i^1}, \ldots, \widetilde{v_i^n}$. Let $\widetilde{l} \leftarrow \widetilde{v_i^j}$. Output $\widetilde{l}$.
3. The **tampering function** $\mathbf{G}(r)$ is defined as follows : On input $r$, replace $tv_j^i$ by $r$ in $tShare_j$ to obtain $share_j$. Run $f_j$ on $share_j$ to obtain $\widetilde{share_j}$. Parse $\widetilde{share_j}$ as $\widetilde{l_j}, \widetilde{r_j}, \widetilde{v_j^1}, \ldots, \widetilde{v_j^n}$. Let $\widetilde{r} \leftarrow \widetilde{v_j^i}$. Output $\widetilde{r}$.

The functions $F$ and $G$ have been defined in this way to ensure that the message encoded by $l$ and $r$ of the coding cheme $(\mathbf{NMEnc}, \mathbf{NMDec})$ is the same as the secret hidden by $share_i$ and $share_j$ of the secret sharing scheme $(\mathbf{NMShare_n^2}, \mathbf{NMRec_n^2})$. We can employ a hybrid argument similar to the one in proof of statistical privacy to argue that the statistical distance in between the distribution of $share_i, share_j$ generated while executing $\mathbf{NMShare_n^2}(m)$ and the two shares generated by the reduction is at most $4(n-1)\epsilon$. Therefore, the tampering experiments of non-malleable codes (see definition 3) and non-malleable secret-sharing schemes (see definition 6) are statistically indistinguishable, specifically,

$$\mathbf{STamper_m^{f,T}} \approx_{4(n-1)\epsilon} \mathbf{Tamper_m^{F,G}}$$

By the $\epsilon$-non malleability of the scheme $(\mathbf{NMEnc}, \mathbf{NMDec})$, we know that there exists a distribution $\mathbf{D^{F,G}}$ such that

$$\mathbf{Tamper_m^{F,G}} \approx_\epsilon \mathbf{Sim_m^{F,G}}$$

Using the underlying distribution $\mathbf{D^{F,G}}$ as our distribution $\mathbf{SD^{f,T}}$ we get our simulator.

$$\mathbf{Sim_m^{F,G}} \equiv \mathbf{SSim_m^{f,T}}$$

Applying triangle inequality to the above relations we prove the statistical non malleability of our scheme.

$$\mathbf{STamper_m^{f,T}} \approx_{4n\epsilon} \mathbf{SSim_m^{f,T}}$$

□

## B Why simple approaches do not work ?

We would like to mention why some of the simple approaches do not work :

- **First attempt** : Can we build computational non-malleable secret-sharing schemes from signatures schemes? It is easy to see that any construction of non-malleable secret-sharing implies a construction of (some form of) split-state non-malleable code. So even before constructing such secret-sharing schemes, it may be natural goal to build computational non-malleable codes. Unfortunately, as far as we know, the only computational split-state non-malleable code built without using additive combinatorics (or alternating extraction) is the one of Liu and Lysyanskaya [LL12], which ensures non-malleability *only in the CRS model.* So in fact, as far as we know, it is an open problem to obtain a "simple" construction of computational split state non-malleable codes from standard computational assumptions in the plain model. We try to highlight why it might be hard to build non-malleable codes in a black-box way from signatures. While it is true that signatures guarantee some non-malleability, but this guarantee only holds when the "verification key" remains unaltered (we might have to additionally deal with tampering of "signing key" as well). So if we wish to construct non-malleable codes in plain-model, we would have to embed this "verification key" somehow into the two states of non-malleable code. Unfortunately, this allows the adversary to change the "verification key" and we can no longer rely on the non-malleability of signatures (in a black-box way). The construction of Liu and Lysyanskaya instead relies on non-malleable non-interactive zero-knowledge which necessarily relies on a trusted CRS: something we wish to avoid in the current work.

- **Second attempt** : What about a tree-based construction? Consider, for example, a complete binary tree with $2^k$ leaves corresponding to $2^k$ parties. To share a secret, we put the secret at the root of this tree, and encode it using a non-malleable code to obtain the value of nodes at level 1 ( children of root). We can recursively apply this process to obtain the value of all the $2^k$ leaves, and these values correspond to the shares of $2^k$ parties (this gives us a $2^k$-out-of-$2^k$ secret sharing scheme). While this seems like a promising approach, there a couple of basic issues. As constant rate statistical non-malleable codes are not yet constructed, the sizes of shares will blow up expondentially with the depth of the tree $k$. The best known statistical non-malleable code of Li [Li17] has rate $\frac{1}{\log(n)}$, and therefore can only support O(log n / loglog n) depth, while we need $\Omega(\log(n))$. However even more fundamentally, we were unable to reduce the security of such a construction to that of the underlying 2-out-of-2 non-malleable code. As a concrete example, consider a simple depth 2 tree having 4 leaves and 3 AND gates. Suppose the first subset chosen by the adversary includes the first and the last leaf of the tree while the second one includes the second and the third leaf. It seems hard to come up with a reduction to the underlying split state non-malleable code, and it seems that we need stronger variants of non-malleable code (while still maintaining constant rate). Note that the property required from the underlying non-malleable code may also vary with different choice of partioning, for example, consider this partition $\{1\}$ and $\{2, 3, 4\}$.