Non-malleable Secret Sharing*

Vipul Goyal
CMU
Pittsburgh, USA
goyal@cs.cmu.edu

Ashutosh Kumar
UCLA
Los Angeles, USA
a@ashutoshk.com

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.

CCS CONCEPTS

• **Security and privacy** → **Information-theoretic techniques**; *Mathematical foundations of cryptography*; • **Theory of computation** → *Cryptographic primitives*;

KEYWORDS

Non-Malleable Codes, Secret Sharing, Threshold Cryptography, Non-Malleable Secret Sharing, Non-Malleable Message Transmission

ACM Reference Format:

Vipul Goyal and Ashutosh Kumar. 2018. Non-malleable Secret Sharing. In *Proceedings of 50th Annual ACM SIGACT Symposium on the Theory of Computing (STOC’18)*. ACM, New York, NY, USA, 14 pages. <https://doi.org/10.1145/3188745.3188872>

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 [29] and Blakley [6] 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 [10], and, verifiable secret sharing [27]. A

*The full version appears on Crypto ePrint : <https://eprint.iacr.org/2018/316/>
The first author is supported by a grant from Northrop Grumman.

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.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

STOC’18, June 25–29, 2018, Los Angeles, CA, USA

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-5559-9/18/06...\$15.00

<https://doi.org/10.1145/3188745.3188872>

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. [17] 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 [25] 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. [14] proposed a construction which could encode single bit messages. A subsequent brilliant lines of works resulted in constructions for multi-bit messages as well [1, 7, 24].

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 function 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 [17], designing multi-prover interactive proof systems [18] and obtaining efficient encryption schemes [9]. 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 [20] (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 [3, 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 [29], forms the basis of secure multi-party computation protocols [5]. 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 efficient¹ 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.*

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

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

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 subsets² 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 [1, 7, 14, 24] (even leakage-resilient ones [2, 19]) 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 $f(l, \text{leak}(r))$ and the second share as $g(r)$, where f and g are arbitrary tampering functions and $\text{leak}(r)$ 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 [13, 21, 22, 30, 31], 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” 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 [1, 7, 14, 24]. 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

²One 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.

³Very recently, Goyal et al. [19] 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.

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

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 a $(t-1)$ -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 $(t-1)$ 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 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 $l_1, \dots, l_n$. We share r using a 2-out-of- n secret sharing scheme to obtain n shares $r_1, \dots, r_n$. 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 $\widetilde{l}_1, \dots, \widetilde{l}_n$. 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 $\widetilde{l}_1, \dots, \widetilde{l}_n$.
- 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 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.

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 $\widetilde{r}$ 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 $\widetilde{r}$. Moreover, as no subset can have all the t shares, we can argue that the fixed value of tampered $\widetilde{r}$ (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

⁵The 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 .

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 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 [7], 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 [29]. 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. Let’s 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 ϵ , by executing $O(\log(1/\epsilon))$ such checks in parallel, we can reduce the chance of not detecting the $\perp$ to lower than ϵ . 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 to 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 [28] 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. [23]. 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. [7] 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 :

- **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-malleable 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 [28] (instead of inner-product), and rely on its “strongness” (and ideas from the beautiful work of Cohen [8]) 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 [11, 15]), 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 uniformly 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 [7]), 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 :

- **Handling 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 $\frac{t}{2}$ ⁶.
- **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 present the construction of leakage-resilient non-malleable code, non-malleable message transmission and 2-out-of- n non-malleable (resp. leakage-resilient) secret sharing schemes in the full version of this extended abstract.

⁶We 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

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 [26] noticed that, the polynomial based Shamir's secret sharing scheme [29] is very closely related to Reed-Solomon codes, and allows for correct reconstruction of the secret even if less than one third of the shares are altered.

In a seminal work [27], 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. [10], 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. [12].

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 $\{1, 2, \dots, m\}$, and 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 $S_{i_1} \times S_{i_2} \times \dots \times S_{i_{|B|}}$, where $i_1, i_2, \dots, i_{|B|}$ are ordered elements of B , such that $i_j < i_{j+1}$.

Definition 2.1. (Statistical Distance) Let D_1 and D_2 be two distributions on a set S . The statistical distance between D_1 and D_2 is defined to be :

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

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

2.1 Non-Malleable Codes

Definition 2.2. (Coding Schemes) ([17]). A coding scheme consists of two functions : an encoding function (possibly randomized) $\text{Enc} : \mathcal{M} \rightarrow \mathcal{C}$, and a deterministic decoding function $\text{Dec} : \mathcal{C} \rightarrow \mathcal{M} \cup \{\perp\}$ such that , for each $m \in \mathcal{M}$, $Pr(\text{Dec}(\text{Enc}(m)) = m) = 1$ (over the randomness of the encoding function).

Definition 2.3. (Non-Malleable Codes) ([17]). Let $\mathcal{F}$ be some family of tampering functions. For each $f \in \mathcal{F}$, and $m \in \mathcal{M}$, define

the tampering experiment

$$\text{Tamper}_m^f = \left\{ \begin{array}{l} c \leftarrow \text{Enc}(m) \\ \tilde{c} \leftarrow f(c) \\ \tilde{m} \leftarrow \text{Dec}(\tilde{c}) \\ \text{Output} : \tilde{m} \end{array} \right\}$$

which is random variable over the randomness of the encoding function Enc . We say a coding scheme (Enc, Dec) is ϵ -**non-malleable** w.r.t $\mathcal{F}$ if for each $f \in \mathcal{F}$, there exists a distribution D^f (corresponding to the simulator) over $\mathcal{M} \cup \{\text{same}^*, \perp\}$ such that, for all $m \in \mathcal{M}$, we have that the statistical distance between Tamper_m^f and

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

is at most ϵ . Additionally, D^f should be efficiently samplable given oracle access to $f(\cdot)$.

2.2 Secret Sharing Schemes

The following definitions are inspired from the survey [4].

Definition 2.4. (Sharing function) Let $[n] = \{1, 2, \dots, n\}$ be a set of identities of n parties. Let $\mathcal{M}$ be the domain of secrets. A **sharing function** Share is a randomized mapping from $\mathcal{M}$ to $S_1 \times S_2 \times \dots \times S_n$, where S_i is called the domain of shares of party with identity i . A dealer distributes a secret $m \in \mathcal{M}$ by computing the vector $\text{Share}(m) = (s_1, \dots, s_n)$, and privately communicating each share s_j to the party j . For a set $S \subseteq [n]$, we denote $\text{Share}(m)_S$ to be a restriction of $\text{Share}(m)$ to its S entries.

Definition 2.5. $((t, n, \epsilon)$ -Secret Sharing Scheme [4]). Let $\mathcal{M}$ be a finite set of secrets, where $|\mathcal{M}| \geq 2$. Let $[n] = \{1, 2, \dots, n\}$ be a set of identities (indices) of n parties. A sharing function Share with domain of secrets $\mathcal{M}$ is a (t, n, ϵ) -**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 $\text{Rec} : \otimes_{i \in T} S_i \rightarrow \mathcal{M}$ such that for every $m \in \mathcal{M}$,

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

(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 D with output in $\{0, 1\}$, the following holds :

$$\left| \begin{array}{l} Pr_{\text{shares} \leftarrow \text{Share}(a)}[D(\text{shares}_U) = 1] \\ - Pr_{\text{shares} \leftarrow \text{Share}(b)}[D(\text{shares}_U) = 1] \end{array} \right| \leq \epsilon$$

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

Definition 2.6. (Non-Malleable Secret Sharing Schemes) Let $(\text{Share}, \text{Rec})$ be a (t, n, ϵ) -secret sharing scheme for message space $\mathcal{M}$. Let $\mathcal{F}$ be some family of tampering functions. For each $f \in \mathcal{F}$, $m \in \mathcal{M}$ and authorized set $T \subseteq [n]$ containing t indices, define the

tampering experiment

$$\text{STamper}_m^{f,T} = \left\{ \begin{array}{l} \text{shares} \leftarrow \text{Share}(m) \\ \overline{\text{shares}} \leftarrow f(\text{shares}) \\ \tilde{m} \leftarrow \text{Rec}(\overline{\text{shares}}_T) \\ \text{Output} : \tilde{m} \end{array} \right\}$$

which is a random variable over the randomness of the sharing function **Share**. We say that the (t, n, ϵ) -secret sharing scheme, $(\text{Share}, \text{Rec})$ is ϵ' -**non-malleable** w.r.t $\mathcal{F}$ if for each $f \in \mathcal{F}$ and authorized set T containing t indices, there exists a distribution $\text{SD}^{f,T}$ (corresponding to the simulator) over $\mathcal{M} \cup \{\text{same}^*, \perp\}$ such that, for all $m \in \mathcal{M}$, we have that the statistical distance between $\text{STamper}_m^{f,T}$ and

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

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

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 $\mathcal{F}_n^{\text{split}}$. Let **Share** be a sharing function that takes as input a message $m \in \mathcal{M}$ and outputs shares $\text{shares} \in \otimes_{i \in [n]} \mathcal{S}_i$. Parse the output shares into n blocks, namely $\text{share}_1, \text{share}_2, \dots, \text{share}_n$ where each $\text{share}_i \in \mathcal{S}_i$. For each $i \in [n]$, let $f_i : \mathcal{S}_i \rightarrow \mathcal{S}_i$ be an arbitrary tampering function, that takes as input share_i , the i^{th} share. Let $\mathcal{F}_n^{\text{split}}$ be a family of such n functions $(f_1, f_2, \dots, f_n)$.

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 3.1. ([2, Lemma 6.1]) *Let $\text{Enc} : \mathcal{M} \rightarrow \mathcal{C}^2$ be the encoding function, and $\text{Dec} : \mathcal{C}^2 \rightarrow \mathcal{M} \cup \{\perp\}$ be a deterministic decoding function. If a coding scheme (Enc, Dec) is ϵ -non-malleable w.r.t. $\mathcal{F}_2^{\text{split}}$ then (Enc, Dec) is also a $(2, 2\epsilon)$ -secret sharing scheme that is ϵ -non-malleable w.r.t. $\mathcal{F}_2^{\text{split}}$, 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 3.2. (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, ϵ) -secret sharing scheme, $(\text{Share}, \text{Rec})$ is ϵ' -**leakage-resilient** w.r.t. $\mathcal{L}$ if for each $f \in \mathcal{L}$, and for any two messages $a, b \in \mathcal{M}$, any distinguisher D with output in $\{0, 1\}$, the following holds :

$$\left| \begin{array}{l} \Pr_{\text{shares} \leftarrow \text{Share}(a)}[D(f(\text{shares})) = 1] \\ - \Pr_{\text{shares} \leftarrow \text{Share}(b)}[D(f(\text{shares})) = 1] \end{array} \right| \leq \epsilon'$$

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

Main result for individual tampering.

THEOREM 3.3. *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 $(\text{NMEnc}, \text{NMDec})$ be any coding scheme that is ϵ_1 -non-malleable w.r.t. $\mathcal{F}_2^{\text{split}}$, which encodes an element of $\mathcal{M}$ into two elements of $\mathbb{F}_1$.*
- (2) *For any $\epsilon_2 \geq 0$, let $(\text{TShare}_n^t, \text{TRec}_n^t)$ be any (t, n, ϵ_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 $(\text{LRShare}_n^2, \text{LRRec}_n^2)$, be any $(2, n, \epsilon_3)$ -secret sharing scheme that is ϵ_3 -leakage-resilient w.r.t. $\mathcal{L}_\mu^{\text{split}}$, 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. $\mathcal{F}_n^{\text{split}}$. The resulting scheme, $(\text{NMShare}_n^t, \text{NMRec}_n^t)$, 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 a simple scheme constructed in the full version. Otherwise, we give our construction of $(\text{NMShare}_n^t, \text{NMRec}_n^t)$:

(1) Sharing Function(NMShare_n^t)

On input a secret $m \in \mathcal{M}$, encode m using the encoding procedure of the non-malleable code. Let $l, r \leftarrow \text{NMEnc}(m)$. Use the sharing procedure TShare_n^t to share l . Let $((l_1, \dots, l_n) \leftarrow \text{TShare}_n^t(l)$. Use the sharing function of the leakage-resilient secret sharing scheme to share r . Let $(r_1, \dots, r_n) \leftarrow \text{LRShare}_n^2(r)$. For each $i \in [n]$, construct the i^{th} share of our scheme as follows : $\text{share}_i = (l_i, r_i)$. Output $(\text{share}_1, \dots, \text{share}_n)$

(2) Reconstruction Function(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 $\{i_1, i_2, \dots, i_t\}$ be ordered indices of T such that $i_j < i_{j+1}$ for each $j \in [t-1]$. On input the shares $\otimes_{i \in T} \text{share}_i$, for each $i \in T$, parse share_i as (l_i, r_i) . Run the reconstruction procedure TRec_n^t on t shares of l , to obtain $l \leftarrow \text{TRec}_n^t(\otimes_{i \in T} l_i)$. Run the reconstruction procedure of the leakage-resilient secret sharing scheme on the first two shares of r , to obtain $r \leftarrow \text{LRRec}_n^2(r_{i_1}, r_{i_2})$. Decode

l and r using decoding process of underlying non-malleable code to obtain : $m \leftarrow \text{NMDec}(l, r)$. Output m .

Let us show that $(\text{NMShare}_n^t, \text{NMRec}_n^t)$ 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 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) **Hybrid₁** : 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 NMShare_n^t on input a . Output $\otimes_{i \in U} \text{share}_i$.
- (2) **Hybrid₂** : Sample the shares as in **Hybrid₁**, 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 $\otimes_{i \in U} \text{share}_i$.
- (3) **Hybrid₃** : Sample the shares as in **Hybrid₂**, 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 NMShare_n^t on input b . Output $\otimes_{i \in U} \text{share}_i$.

It follows from the statistical privacy of the scheme $(\text{TShare}_n^t, \text{TRec}_n^t)$ that for any pair of secrets $a, b \in \mathcal{M}$, any unauthorized set U , the statistical distance between **Hybrid₁** and **Hybrid₂** is at most ϵ_2 .

Claim: For any pair of secrets $a, b \in \mathcal{M}$, any unauthorized U , the statistical distance between **Hybrid₂** and **Hybrid₃** 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 3.1.

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 $r_1, r_2, \dots, r_n \leftarrow \text{LRShare}_n^2(r)$. Sample $\otimes_{i \in U} \text{share}_i$ according to **Hybrid₂**. For each $i \in U$, replace ar_i with r_i in share_i to obtain a share of form al_i, r_i, br_i . Invoke the distinguisher $\mathbf{D}$ with the modified $\otimes_{i \in U} \text{share}_i$ 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 **Hybrid₂**. Otherwise, it will be invoked with distributed according to **Hybrid₃**. 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 $(\text{NMEnc}, \text{NMDec})$. ■

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 **Hybrid₁** and **Hybrid₃** 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 $\mathcal{F}_n^{\text{split}}$, it specifies a set of n tampering functions $\{f_i : i \in [n]\}$. All these functions act on their respective shares independently of the other shares, i.e. every f_i takes share_i as input and outputs the tampered 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 $\{f_i : i \in T\}$ that tamper with the shares of our scheme $(\text{NMShare}_n^t, \text{NMRec}_n^t)$ to create explicit tampering functions $\mathbf{F}$ and $\mathbf{G}$ that tamper with the two parts of non-malleable code. Recall that as $\mathcal{F}_2^{\text{split}}$ allows arbitrary computation, the functions $\mathbf{F}$ and $\mathbf{G}$ are allowed to brute force over any finite subset.

- (1) **(Initial Setup)** : Fix l_s and r_s encoding any $m_s \leftarrow \text{NMDec}(l_s, r_s)$. Run the sharing function TShare_n^t with input l_s to obtain $\otimes_{i \in [n]} tl_i$. Run the sharing function $\text{LRShare}_n^2(r_s)$ to obtain $\otimes_{i \in [n]} tr_i$. For each $i \in [n]$, create $t\text{Share}_i$ as tl_i, tr_i . For each $i \in T \setminus \{i_t\}$, run f_i on $t\text{Share}_i$ to obtain $\widetilde{t\text{Share}_i} \leftarrow f_i(t\text{Share}_i)$. Parse $\widetilde{t\text{Share}_i}$ as $\widetilde{tl_i}, \widetilde{tr_i}$. Fix $l_i \leftarrow tl_i$ and $\widetilde{r_i} \leftarrow \widetilde{tr_i}$. 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 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 f_{i_t} on share_{i_t} to obtain tampered $\widetilde{\text{share}_{i_t}}$. Parse $\widetilde{\text{share}_{i_t}}$ as $\widetilde{l_{i_t}}, \widetilde{r_{i_t}}$. Run the reconstruction function 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 LRShare_n^2 on

input r . In other words, these shares should correspond to the t shares of the scheme $(\text{LRShare}_n^2, \text{LRRec}_n^2)$, in which any two of these shares can be used to reconstruct r .

- For each $i \in T \setminus \{i_t\}$, let $\widetilde{\text{share}}_i$ be $\widetilde{l}_i, r_i$, run $\mathbf{f}_1$ on $\widetilde{\text{share}}_i$ to obtain $\widetilde{\text{share}}_i$. Parse $\widetilde{\text{share}}_i$ 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 $\widetilde{r} \leftarrow \text{LRRec}_n^2(\widetilde{nl}_1, \widetilde{nr}_1, \widetilde{nl}_2, \widetilde{nr}_2)$. 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 NMShare_n^t conditioned on output of $\text{NMEnc}(m)$ 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_s and r_s . Similarly, let share_i be of the form bl_i, br_i when the secret m is encoded by the sharing procedure NMShare_n^t conditioned on output of $\text{NMEnc}(m)$ being l, r .

- (1) **Hybrid₁** : 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_s and r_s . Output $\otimes_{i \in T} \text{share}_i$.
- (2) **Hybrid₂** : Fix $l_s \leftarrow l$ in the initial setup phase. Proceed with the reduction to create the t shares. Output $\otimes_{i \in T} \text{share}_i$.
- (3) **Hybrid₃** : Fix $l_s \leftarrow l$ and fix $r_s \leftarrow r$ in the initial setup phase. Proceed with the reduction to create the t shares. Output $\otimes_{i \in T} \text{share}_i$.
- (4) **Hybrid₄** : 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 NMShare_n^t conditioned on output of $\text{NMEnc}(m)$ being l, r . Output $\otimes_{i \in T} \text{share}_i$.

Claim: For any l, l_s , any authorized set T containing t indices, the statistical distance between **Hybrid₁** and **Hybrid₂** is at most ϵ_2 .

Proof: The two hybrids differ in the initial setup phase. In **Hybrid₁**, $t - 1$ shares of l_s are fixed, while in **Hybrid₂** $t - 1$ shares of l are fixed. To violating the statistical secrecy of TShare_n^t , we 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_s and there exists a distinguisher $\mathbf{D}$ which differentiates the two distributions with probability greater than ϵ_2 . In this case, we create a distinguisher $\mathbf{D}_1$ which violates the statistical privacy property of TShare_n^t mentioned above. More formally, given a set of $t - 1$ shares of l or l_s , 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 TShare_n^t on l_s in the initial setup phase and proceeds in a manner similar to **Hybrid₁**. Then it invokes the distinguisher $\mathbf{D}$ with the sampled shares and outputs its output. Notice, in case $l_{i_1}, \dots, l_{i_{t-1}}$ encoded the message l_s , we are

in a distribution identical to **Hybrid₁**. Otherwise, if they encoded the secret l , we are in **Hybrid₂**. Therefore, the distinguishing advantages are of both the distinguishers is the same, and we have arrived at the contradiction of statistical privacy of (t, n, ϵ_2) -secret sharing scheme $(\text{TShare}_n^t, \text{TRec}_n^t)$. ■

The above also shows that the function $\mathbf{F}$ in the reduction aborts with probability less than ϵ_2 .

Claim: For any r, r_s , any authorized set T containing t indices, the statistical distance between **Hybrid₂** and **Hybrid₃** is at most ϵ_3 .

Proof: Assume towards contradiction that there exists $r, r_s \in \mathbb{F}_1$, and a distinguisher $\mathbf{D}$ that is successful in distinguishing **Hybrid₂** and **Hybrid₃** with probability greater than ϵ_3 . We use distinguisher $\mathbf{D}$ to construct another distinguisher $\mathbf{D}_1$ and a leak function $\mathbf{g} \in \mathcal{L}_\mu^{\text{split}}$ which violates the property of leakage-resilience satisfied by the scheme $(\text{LRShare}_n^2, \text{LRRec}_n^2)$ for the secrets r, r_s . The reduction is described below :

- (1) **(Initial Setup)** : Run the sharing function TShare_n^t 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 $\mathbf{g}$)** : 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 $t\text{Share}_i$ as l_i, r_i . Run $\mathbf{f}_1$ on $t\text{Share}_i$ to obtain $t\text{Share}_i \leftarrow \mathbf{f}_1(t\text{Share}_i)$. Parse $t\text{Share}_i$ as tl_i, tr_i . Output tl_i .

As 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 μ . This shows that the above leak function $\mathbf{g}$ belongs to the class $\mathcal{L}_\mu^{\text{split}}$ which allows one share to be fully leaked and μ 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 $\mathbf{g}(r_1, \dots, r_n)$, parse it as $tl_{i_1}, tl_{i_2}, \dots, tl_{i_{t-1}}, r_{i_t}$. For each $i \in T \setminus \{i_t\}$, fix $l_i \leftarrow tl_i$. 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 $\{\text{share}_i : i \in T\}$. 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_s , $\mathbf{D}$ will be invoked with input distributed according to **Hybrid₂**. In the other case, in which r was hidden, $\mathbf{D}$ will be invoked with distributed according to **Hybrid₃**. 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 ϵ_3 by assumption. Hence, we have arrived at a contradiction to statistical leakage-resilience property of the scheme $(\text{LRShare}_n^2, \text{LRRec}_n^2)$. ■

The above also shows that the function $\mathbf{G}$ in the reduction aborts with probability less than ϵ_3 .

Claim: For any l, r , **Hybrid**₃ is identical to **Hybrid**₄.

Proof: In **Hybrid**₃, the shares of r_s that are sampled in the initial setup already encode the value r (as $r_s = r$). Recall that these shares are used for fixing the tampered shares of l , but now as the unfixed shares of r_s are already from the correct distribution, the tampering function $G(r)$ can trivially use the same shares, and all the leakages (deterministic function computing tampered shares of $\tilde{l}$) will be trivially satisfied. Similarly, $F(l)$ can use the value of unfixed share tl_{i_t} , because the t shares of l already encode l (as $l_s = l$ in **Hybrid**₃). (Basically, (F, G) will be always successful in finding such satisfying shares by brute-force.) Therefore, all the t shares created in **Hybrid**₃ will be identically distributed to the ones produced while executing NMShare_n^t with the output of NMEnc being (l, r) . ■

By repeated application of triangle inequality, we get that for any $a, b \in \mathcal{M}$, the statistical distance between **Hybrid**₁ and **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 F and 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,

$$\text{NMRec}_n^t(\{\text{share}_i : i \in T\}) = \text{NMDec}(l, r)$$

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,

$$\text{NMRec}_n^t(\{\mathbf{f}_i(\text{share}_i) : i \in T\}) = \text{NMDec}(F(l), G(r))$$

Therefore, the tampering experiments of non-malleable codes (see definition 2.3) and non-malleable secret-sharing schemes (see definition 2.6) are statistically indistinguishable, specifically,

$$\text{STamper}_m^{\mathbf{f}, T} \approx_{\epsilon_2 + \epsilon_3} \text{Tamper}_m^{\mathbf{F}, G}$$

By the ϵ_1 -non malleability of the scheme $(\text{NMEnc}, \text{NMDec})$, we know that there exists a distribution $\mathbf{D}^{\mathbf{F}, G}$ such that

$$\text{Tamper}_m^{\mathbf{F}, G} \approx_{\epsilon_1} \text{Sim}_m^{\mathbf{F}, G}$$

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

$$\text{Sim}_m^{\mathbf{F}, G} \equiv \text{SSim}_m^{\mathbf{f}, T}$$

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

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

□

COROLLARY 3.4. *For any threshold $t \geq 2$, any number of parties $n \geq t$, any $\epsilon > 0$, there exists an efficient (t, n, ϵ) -secret sharing scheme that is ϵ -non-malleable w.r.t. $\mathcal{F}_n^{\text{split}}$. 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 3.3 using the following constructions :

- (1) We use split-state non-malleable code constructed by Li [24] (**LiNMEnc**, **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 [29] (**TShare**_n^t, **TRec**_n^t). The sharing function **TShare**_n^t shares a k bit secret into n shares each of size k bits.
- (3) We use the 2-out-of- n secret sharing scheme that is ϵ_3 -leakage-resilient w.r.t. to $\mathcal{L}_\mu^{\text{split}}$ and encodes a k bit secret into n shares, where each share has length $O(nk)$, as constructed in the full version. We set $\epsilon_3 = \frac{\epsilon}{2}$.

In case threshold t is equal to two, we instantiate the theorem 3.3 with the split-state non-malleable code constructed by Li [24] (**LiNMEnc**, **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 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 $\mathcal{F}_{t,n}^{\text{joint}}$. 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 cardinalities⁷, 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 the full version⁸.

Definition 4.1. (Leakage-Resilient Non-Malleable Codes) Let $\mathcal{M}$ be any message space. Let (Enc, Dec) be any coding scheme, such that $\text{Enc} : \mathcal{M} \rightarrow \mathcal{L} \times \mathcal{R}$ encodes any message into two parts. Let $\text{Leak} : \mathcal{R} \rightarrow \{0, 1\}^\mu$ be any leakage function that outputs μ bits of arbitrary information about its input. Similarly, let $\mathbf{F} : \mathcal{L} \times \{0, 1\}^\mu \rightarrow \mathcal{L}$ and $\mathbf{G} : \mathcal{R} \rightarrow \mathcal{R}$ 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

⁷We 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.

⁸More general definition of multi-round leakage-resilient non-malleable codes can be found in the beautiful work of Aggarwal et al. [2]. 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 [2, 15, 16]), 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.

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

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

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

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

We now give the main result of this section.

THEOREM 4.2. *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 $(\text{NMEnc}, \text{NMDec})$ 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 ϵ -non-malleable (as defined in def 4.1).*

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

PROOF. The case of threshold $t = 2$ follows from the previous theorem 3.3. Otherwise, we begin with defining our building blocks :

- Let $(\text{TShare}_n^t, \text{TRec}_n^t)$ denote t -out-of- n Shamir's secret sharing scheme [29] 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 $(\text{TShare}_n^k, \text{TRec}_n^k)$ 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**(JNMShare_n^t)

On input a secret $m \in \mathcal{M}$, encode m using the encoding procedure of the non-malleable code. Let $l, r \leftarrow \text{NMEnc}(m)$. Use the sharing function of the t -out-of- n threshold secret sharing scheme to share l . Let $(l_1, \dots, l_n) \leftarrow \text{TShare}_n^t(l)$. Use the sharing function of the k -out-of- n threshold secret sharing scheme to share r . Let $(r_1, \dots, r_n) \leftarrow \text{TShare}_n^k(r)$. For each $i \in [n]$, construct the i^{th} share of our scheme as follows : $\text{share}_i = (l_i, r_i)$. Output $(\text{share}_1, \dots, \text{share}_n)$

(2) **Reconstruction Function**(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 $(\text{share}_{i_1}, \dots, \text{share}_{i_t})$, for each $i \in T$, parse share_i as (l_i, r_i) . Verify that all the t shares of r , namely $(r_{i_1}, \dots, r_{i_t})$ 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, TRec_n^t on the t shares of l , namely, $(l_{i_1}, \dots, l_{i_t})$. Reconstruct r by running the reconstruction procedure of the k -out-of- n threshold secret sharing scheme, TRec_n^k on the t shares of r , namely, $(r_{i_1}, \dots, r_{i_t})$. Then decode l and r using decoding process of underlying non-malleable code to obtain : $m \leftarrow \text{NMDec}(l, r)$ and output m .

Correctness, efficiency and statistical privacy : Correctness and efficiency trivially follows. A proof similar to one given for theorem 3.3 shows statistical privacy.

Statistical Non Malleability : The adversary specifies an authorized set T of cardinality t . As the adversary is in $\mathcal{F}_{t,n}^{\text{joint}}$, 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 3.3, 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 $\mathbf{F}, \mathbf{G}, \text{Leak}$.

- (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 $\text{ECC} : \mathcal{R}^{|B|} \rightarrow \{0, 1\}^p$ 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 : 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 F_A on the set of shares $\{share_i : i \in A\}$ to obtain tampered shares $\{\widehat{share}_i : i \in A\}$. For each $i \in A$, parse $\widehat{share}_i$ as $\tilde{l}_i, \tilde{r}_i$. Using the tampered shares of l compute the following :

$$sum \leftarrow \sum_{i \in A} \alpha_i \times \tilde{l}_i$$

where each α_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 [29] where one obtains the secret as $\tilde{l} \leftarrow \sum_{i \in A \cup B} \alpha_i \times \tilde{l}_i$).

If the tampered shares of r , namely $\{\tilde{r}_i : i \in A\}$, 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 $\tilde{r}$, namely $\{\widehat{r}_i : i \in B\}$. Now concatenate all these (interpolated) values to obtain a string $\widehat{cr} \leftarrow \widehat{cr}_{i_1} \circ \widehat{cr}_{i_2} \circ \dots \circ \widehat{cr}_{i_{|B|}}$, where $(i_1, \dots, i_{|B|})$ are ordered indices present in the set B . Encode the string $\widehat{cr} \in \mathcal{R}^{|B|}$ using ECC to obtain $\overline{cr} \leftarrow \text{ECC}(\widehat{cr})$. Use randomness $\$C$ to sample $\log(\frac{1}{\epsilon})$ locations from the binary string $\overline{cr}$ to obtain

$$samples \leftarrow \overline{cr}|_{\$C}$$

Output ($sum \circ samples$).

- (4) (**Tampering function F(l , Leak(r))**) : On input l and 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 F_B on this set of shares to obtain $\{\widehat{share}_j : j \in B\}$. For each $j \in B$, parse $\widehat{share}_j$ as $\tilde{l}_j, \tilde{r}_j$. Concatenate all the tampered shares of r , namely $\{\tilde{r}_i : i \in B\}$, to obtain a string $\widehat{r}_B \leftarrow \tilde{r}_{i_1} \circ \tilde{r}_{i_2} \circ \dots \circ \tilde{r}_{i_{|B|}}$, where $(i_1, \dots, i_{|B|})$ are ordered indices present in the set B . Encode the string $\widehat{r}_B \in \mathcal{R}^{|B|}$ using ECC to obtain $\overline{r}_B \leftarrow \text{RS}(\widehat{r}_B)$. Use randomness $\$C$ to sample $\log(\frac{1}{\epsilon})$ locations from the binary string $\overline{r}_B$ to obtain

$$samples_B \leftarrow \overline{r}_B|_{\$C}$$

Parse 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 :

$$\tilde{l} \leftarrow sum + \sum_{i \in B} \alpha_i \times \tilde{l}_i$$

where α_i is the Lagrangian's coefficient (as in the Leak(r) function above).

- (5) (**Tampering function G(r)**) : On input r , proceed as in leakage function Leak(r) to compute tampered shares of r , namely $\{\tilde{r}_i : i \in A\}$. 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 $\tilde{r}$ by running TRec_n^k on $\{\tilde{r}_i \in A\}$. Output $\tilde{r}$.

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

Now we argue the trickier case, in which the t tampered shares of r are inconsistent and therefore JNMRec_n^t always outputs $\perp$. If the inconsistency is in the shares of subset A , then the function 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 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 $\tilde{r}$ and $\tilde{r}_B$ differ, then with probability at least $1 - \epsilon$, the strings $samples$ and $samples_B$ will not match and the tampering function F will output $\perp$. Otherwise, with probability at most ϵ , our reduction fails to detect a $\perp$ and outputs valid values of tampered $\tilde{l}$ and $\tilde{r}$. Therefore, overall the statistical distance in between message encoded between $\tilde{l}$ and $\tilde{r}$ constructed by reduction and the secret computed by the reconstruction $JNMRec_n^c$ can be at most ϵ .

We can now say that the tampering experiments for the non-malleable secret-sharing schemes (see definition 2.6) and the underlying leakage-resilient non-malleable codes (see definition 4.1) are statistically indistinguishable, specifically,

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

As we know by the properties of leakage-resilient non-malleable codes (4.1),

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

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

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

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

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

□

ACKNOWLEDGMENTS

We thank the anonymous reviewers, as their detailed and insightful reviews significantly helped in improving the presentation of this article.

REFERENCES

- [1] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. 2014. Non-malleable codes from additive combinatorics. In *Proceedings of the 46th Annual ACM Symposium on Theory of Computing*. ACM, 774–783.
- [2] Divesh Aggarwal, Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. 2015. Leakage-resilient non-malleable codes. In *Twelfth IACR Theory of Cryptography Conference (TCC 2015)*.
- [3] Amos Beimel. *Secure schemes for secret sharing and key distribution*, PhD Thesis.
- [4] Amos Beimel. 2011. Secret-sharing schemes: a survey. In *International Conference on Coding and Cryptology*. Springer Berlin Heidelberg, 11–46.
- [5] Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. 1988. Completeness theorems for non-cryptographic fault-tolerant distributed computation. In *Proceedings of the twentieth annual ACM symposium on Theory of computing*. ACM, 1–10.
- [6] G. R. Blakley. 1979. Safeguarding cryptographic keys. In *AFIPS National Computer Conference (NCC '79)*. IEEE Computer Society, Los Alamitos, CA, USA, 313–317. <https://doi.org/doi.ieeecomputersociety.org/10.1109/AFIPS.1979.98>
- [7] Eshan Chattopadhyay, Vipul Goyal, and Xin Li. 2016. 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*. 285–298. <https://doi.org/10.1145/2897518.2897547>
- [8] Gil Cohen. 2016. Non-Malleable Extractors—New Tools and Improved Constructions. In *31st Conference on Computational Complexity*.
- [9] Sandro Coretti, Yevgeniy Dodis, Björn Tackmann, and Daniele Venturi. 2016. Non-malleable encryption: simpler, shorter, stronger. In *Theory of Cryptography Conference*. Springer, 306–335.
- [10] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. 2008. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In *Eurocrypt*. Springer, 471–488.
- [11] Y. Dodis, R. Ostrovsky, L. Reyzin, and A. Smith. 2008. Fuzzy extractors: How to generate strong keys from biometrics and other noisy data. 38 (2008), 97–139.
- [12] Danny Dolev, Cynthia Dwork, and Moni Naor. 1991. 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*. 542–552.
- [13] Danny Dolev, Cynthia Dwork, Orli Waarts, and Moti Yung. 1993. Perfectly Secure Message Transmission. *J. ACM* 40, 1 (Jan. 1993), 17–47. <https://doi.org/10.1145/138027.138036>
- [14] Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. 2013. Non-malleable codes from two-source extractors. In *CRYPTO 2013*. Springer, 239–257.
- [15] Stefan Dziembowski and Krzysztof Pietrzak. 2007. Intrusion-resilient secret sharing. In *Foundations of Computer Science, 2007. FOCS'07. 48th Annual IEEE Symposium on*. IEEE, 227–237.
- [16] Stefan Dziembowski and Krzysztof Pietrzak. 2008. Leakage-resilient cryptography. In *Foundations of Computer Science, 2008. FOCS'08. IEEE 49th Annual IEEE Symposium on*. IEEE, 293–302.
- [17] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. 2010. Non-Malleable Codes. In *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5–7, 2010. Proceedings*. 434–452.
- [18] Vipul Goyal, Aayush Jain, and Dakshita Khurana. 2015. Non-Malleable Multi-Prover Interactive Proofs and Witness Signatures. *Cryptology ePrint Archive*, Report 2015/1095. (2015). <http://eprint.iacr.org/2015/1095>.
- [19] Vipul Goyal, Ashutosh Kumar, Sunoo Park, Silas Richelson, and Akshayaram Srinivasan. 2018. Non-Malleable Commitments from Non-Malleable Extractors. Manuscript. (2018).
- [20] Vipul Goyal, Omkant Pandey, and Silas Richelson. 2016. 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*. 1128–1141. <https://doi.org/10.1145/2897518.2897657>
- [21] Ravi Kishore, Ashutosh Kumar, Chiranjeevi Vanarasa, and Kannan Srinathan. 2018. On the Price of Proactivizing Round-Optimal Perfectly Secret Message Transmission. *IEEE Transactions on Information Theory* 64, 2 (Feb 2018), 1404–1422. <https://doi.org/10.1109/TIT.2017.2776099>
- [22] Kaoru Kurosawa and Kazuhiro Suzuki. 2009. Truly Efficient-Round Perfectly Secure Message Transmission Scheme. *IEEE Transactions on Information Theory* 55, 11 (2009), 5223–5232.
- [23] Chia-Jung Lee, Chi-Jen Lu, Shi-Chun Tsai, and Wen-Guey Tzeng. 2005. Extracting randomness from multiple independent sources. *IEEE Transactions on Information Theory* 51, 6 (2005), 2224–2227.
- [24] Xin Li. 2017. Improved non-malleable extractors, non-malleable codes and independent source extractors. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*. ACM, 1144–1156.
- [25] Feng-Hao Liu and Anna Lysyanskaya. 2012. Tamper and Leakage Resilience in the Split-State Model. In *CRYPTO 2012 - Volume 7417*. Springer-Verlag New York, Inc., New York, NY, USA, 517–532.
- [26] Robert J. McEliece and Dilip V. Sarwate. 1981. On sharing secrets and Reed-Solomon codes. *Commun. ACM* 24, 9 (1981), 583–584.
- [27] T. Rabin and M. Ben-Or. 1989. Verifiable Secret Sharing and Multiparty Protocols with Honest Majority. In *STOC*. ACM, New York, NY, USA, 73–85. <https://doi.org/10.1145/73007.73014>
- [28] Ran Raz. 2005. Extractors with weak random seeds. In *Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*. ACM, 11–20.
- [29] Adi Shamir. 1979. How to share a secret. *Commun. ACM* 22, 11 (1979), 612–613.
- [30] K. Srinathan, Arvind Narayanan, and C. Pandu Rangan. 2004. Optimal Perfectly Secure Message Transmission. In *In Proc. of Advances in Cryptology: CRYPTO 2004, LNCS 3152*. Springer-Verlag New York, Inc., 545–561.
- [31] Yongge Wang and Yvo Desmedt. 2008. Perfectly secure message transmission revisited. *IEEE Transactions on Information Theory* 54, 6 (2008), 2582–2595.