# Revisiting Non-Malleable Secret Sharing

Saikrishna Badrinarayanan
UCLA
saikrishna@cs.ucla.edu

Akshayaram Srinivasan
University of California, Berkeley
akshayaram@berkeley.edu

August 8, 2019

### Abstract

A threshold secret sharing scheme (with threshold $t$) allows a dealer to share a secret among a set of parties such that any group of $t$ or more parties can recover the secret and no group of at most $t-1$ parties learn any information about the secret. A non-malleable threshold secret sharing scheme, introduced in the recent work of Goyal and Kumar (STOC'18), additionally protects a threshold secret sharing scheme when its shares are subject to tampering attacks. Specifically, it guarantees that the reconstructed secret from the tampered shares is either the original secret or something that is unrelated to the original secret.

In this work, we continue the study of threshold non-malleable secret sharing against the class of tampering functions that tamper each share independently. We focus on achieving greater *efficiency* and guaranteeing a *stronger* security property. We obtain the following results:

- **Rate Improvement.** We give the first construction of a threshold non-malleable secret sharing scheme that has rate $> 0$. Specifically, for every $n, t \geq 4$, we give a construction of a $t$-out-of-$n$ non-malleable secret sharing scheme with rate $\Theta(\frac{1}{t \log^2 n})$. In the prior constructions, the rate was $\Theta(\frac{1}{n \log m})$ where $m$ is the length of the secret and thus, the rate tends to 0 as $m \to \infty$. Furthermore, we also optimize the parameters of our construction and give a concretely efficient scheme.
- **Multiple Tampering.** We give the first construction of a threshold non-malleable secret sharing scheme secure in the stronger setting of bounded tampering wherein the shares are tampered by multiple (but bounded in number) possibly different tampering functions. The rate of such a scheme is $\Theta(\frac{1}{k^3 t \log^2 n})$ where $k$ is an apriori bound on the number of tamperings. We complement this positive result by proving that it is impossible to have a threshold non-malleable secret sharing scheme that is secure in the presence of an apriori unbounded number of tamperings.
- **General Access Structures.** We extend our results beyond threshold secret sharing and give constructions of rate-efficient, non-malleable secret sharing schemes for more general monotone access structures that are secure against multiple (bounded) tampering attacks.

# Contents

# 1 Introduction

A $t$-out-of-$n$ threshold secret sharing scheme [Sha79, Bla79] allows a dealer to share a secret among $n$ parties such that any subset of $t$ or more parties can recover the secret but any subset of $t-1$ parties learn no information about the secret. Threshold secret sharing schemes are central tools in cryptography and have several applications such as constructing secure multiparty computation protocols [GMW87, BGW88, CCD88], threshold cryptographic systems [DF90, Fra90, DDFY94] and leakage resilient circuit compilers [ISW03, FRR$^+$10, Rot12] to name a few.

Most of the threshold secret sharing schemes in literature are *linear*. This means that if we multiply each share by a constant $c$, we get a set of shares that correspond to a new secret that is $c$ times the original secret. This property has in fact, been crucially leveraged in most of the applications including designing secure multiparty computation protocols and constructing threshold cryptosystems. However, this highly desirable feature becomes undesirable if our primary goal is to protect the shares against tampering attacks. More specifically, this linearity property allows an adversary to tamper (or maul) each share independently and output a new set of shares that reconstruct to a related secret (for example, two times the original secret). Indeed, if the shares of the secret are stored on a device such as a smart card, an adversary could potentially tamper with the smart card and change the value of the share that is being stored by overwriting it with a new value or maybe flipping a few bits. Notice that in the above tampering attack, the adversary need not learn the actual secret. However, the adversary is guaranteed to produce a set of shares that reconstruct to a related secret. Such an attack could be devastating when the shares, for example, correspond to a cryptographic secret key (such as a signing key) as it allows an adversary to mount related-key attacks (see [BDL01]). In fact, most of the known constructions of threshold signatures use Shamir's secret sharing to distribute the signing key among the parties and hence they are all susceptible to such attacks.

**Non-Malleable Secret Sharing.** To protect a secret sharing scheme against such share tampering attacks, Goyal and Kumar [GK18a, GK18b] introduced the notion of Non-Malleable Secret Sharing. Roughly, a secret sharing scheme $(\mathsf{Share}, \mathsf{Rec})$ is non-malleable against a tampering function class $\mathcal{F}$ if for every $f \in \mathcal{F}$ and every secret $s$, $\mathsf{Rec}(f(\mathsf{shares}))$ where $\mathsf{shares} \leftarrow \mathsf{Share}(s)$ is either $s$ or something that is unrelated to $s$.[1] Of course, we cannot hope to protect against all possible tampering functions as a function can first reconstruct the secret from the shares, multiply it by 2 and then share this value to obtain a valid sharing of a related secret. Thus, the prior works placed restrictions on the set of functions that can tamper the shares. A natural restricted family of tampering functions that we will consider in this work is $\mathcal{F}_{ind}$ which consists of the set of all functions that tamper each share independently.

**Connection to Non-Malleable Codes.** Non-malleable secret sharing is related to another cryptographic primitive called as Non-Malleable Codes which was introduced in an influential work by Dziembowski, Pietrzak and Wichs [DPW10].[2] A non-malleable code relaxes the usual notion of error correction by requiring that the decoding procedure outputs either the original message or something that is independent of the message when given a tampered codeword as

[1] See Section 3 for a precise definition.

[2] We refer the reader to [GK18a, GK18b] for a thorough discussion on the connection between non-malleable secret sharing and related notions such as verifiable secret sharing [CGMA85] and AMD codes [CDF$^+$08].

input. A beautiful line of work, starting from [DPW10], has given several constructions of non-malleable codes with security against various tampering function classes [LL12, DKO13, FMNV14, FMVW14, ADL14, AGM+15, FMNV15, JW15, CKR16, CGM+16, AAG+16, CGL16, BDKM16, Li17, KOS17, CL17, KOS18, BDKM18, GMW17, OPVV18, KLT18, BDG+18].

We now elaborate on the connection between non-malleable codes and non-malleable secret sharing. A tampering function family in the literature of non-malleable codes that is somewhat similar to $\mathcal{F}_{ind}$ is the $k$-split-state function family. A $k$-split-state function compartmentalizes a codeword into $k$-parts and applies a tampering function to each part, independent of the other parts. Seeing the similarity between $\mathcal{F}_{ind}$ and $k$-split-state functions, it might be tempting to conclude that a non-malleable code against a $k$-split-state function family is in fact a $k$-out-of-$k$ non-malleable secret sharing. However, as demonstrated in [GK18a], this might not be true in general. In particular, [GK18a] showed that even a 3-split-state non-malleable code need not be a 3-out-of-3 non-malleable secret sharing as non-malleable codes may not always protect the secrecy of the message. In particular, the first few bits of the codeword could reveal some bits of the message and still, this coding scheme could be non-malleable. Nevertheless, for the special case of 2, Aggarwal et al. [ADKO15] showed that any 2-split-state non-malleable code is indeed a 2-out-of-2 non-malleable secret sharing scheme. In the other direction, we note that any $k$-out-of-$k$ non-malleable secret sharing scheme against $\mathcal{F}_{ind}$ is in fact a $k$-split-state non-malleable code.

**Rate of Non-Malleable Secret Sharing.** One of the main efficiency parameters in any secret sharing scheme is its *rate* which is defined as the ratio between the length of the secret and the maximum size of a share. In the prior work, Goyal and Kumar [GK18a] gave an elegant construction of $t$-out-of-$n$ non-malleable secret sharing from any 2-split-state non-malleable code. However, the rate of this scheme is equal to $O(\frac{1}{n \log m})$ where $m$ is the length of the secret. The rate tends to 0 as the length of the secret $m$ tends to $\infty$ and hence, a natural question to ask is:

*Can we obtain a construction of threshold non-malleable secret sharing with rate $> 0$?*

The problem of improving the rate was mentioned as an explicit open question in [GK18a].

**Multiple Tamperings.** In the real world, a tampering adversary could potentially mount more than one tampering attack. In particular, if each share of a cryptographic secret key is stored on a small device (such as smart cards), the adversary could potentially clone these devices to obtain multiple copies of the shares. The adversary could then apply a different tampering function on each copy and obtain information about related secrets. Thus, a more realistic security definition would be to consider multiple tampering functions $f_1, \ldots, f_k \in \mathcal{F}$, and require that for every secret $s$, the joint distribution $(\mathsf{Rec}(f_1(\mathsf{shares})), \ldots, \mathsf{Rec}(f_k(\mathsf{shares})))$ where $\mathsf{shares} \leftarrow \mathsf{Share}(s)$ is independent of $s$.[3] For the case of non-malleable codes, security against multiple tamperings has already been considered in [FMNV14, JW15, CGL16, OPVV18]. However, for the case of non-malleable secret sharing, the prior work [GK18a] only considered a single tampering function and a natural question would be:

*Can we obtain a construction of threshold non-malleable secret sharing against multiple tamperings?*

[3] As in the case of single tampering, a tampering function could just output the same shares and in which the reconstructed secret will be $s$. Our definition also captures this property and we refer to Section 3 for a precise definition.

## 1.1 Our Results

In this work, we obtain the following results.

### 1.1.1 Rate Improvement

We give the first construction of a threshold non-malleable secret sharing scheme that has rate $> 0$. Specifically, the rate of our construction is $\Theta(\frac{1}{t \log^2 n})$ where $t$ is the threshold and $n$ is the number of parties. More formally,

**Theorem 1.1** *For any $n, t \geq 4$ and any $\rho > 0$, there exists a construction of $t$-out-of-$n$ non-malleable secret sharing scheme against $\mathcal{F}_{ind}$ for sharing $m$-bit secrets for any $m > \log n$ with rate $\Theta(\frac{1}{t \log^2 n})$ and simulation error $2^{-\Omega(\frac{m}{\log^{1+\rho} m})}$. The running times of the sharing and reconstruction algorithms are polynomial in $n$ and $m$.*

**Local Leakage Resilient Secret Sharing.** One of the main tools used in proving Theorem 1.1 (which may be of independent interest) is an efficient construction of *local leakage-resilient* threshold secret sharing scheme [GK18a, BDIR18]. A $t$-out-of-$n$ secret sharing scheme is said to be local leakage-resilient (parameterized by a leakage bound $\mu$ and set size $s$), if the secrecy holds against any adversary who might obtain at most $t-1$ shares in the clear and additionally, for any set $S \subseteq [n]$ of size at most $s$, the adversary obtains $\mu$ bits from each share belonging to a party in the set $S$. Goyal and Kumar [GK18a] gave a construction of a 2-out-of-$n$ local leakage resilient secret sharing scheme. In this work, we give an efficient construction of $t$-out-of-$n$ local leakage resilient secret sharing scheme when $t$ is a constant. This result must be contrasted with a recent result by Benhamouda et al. [BDIR18] who showed that the Shamir's secret sharing scheme is local leakage resilient when the field size is sufficiently large and the threshold $t = n - o(\log n)$. A more precise statement of our construction of local leakage resilient secret sharing scheme appears below.

**Theorem 1.2** *For any $\varepsilon > 0$, $t, n \in \mathbb{N}$, and parameters $\mu \in \mathbb{N}, s \leq n$, there exists an efficient construction of $t$-out-of-$n$ secret sharing scheme for sharing $m$-bit secrets that is $(\mu, s)$-local leakage resilient with privacy error $\varepsilon$. The size of each share when $t$ is a constant is $O\left((m + s\mu + \log(\log n/\varepsilon)) \log n\right)$.*

**Concrete Efficiency.** A major advantage of our result is its *concrete efficiency*. In the prior work, the constant hidden inside the big-O notation was large and was not explicitly estimated. We have optimized the parameters of our construction and we illustrate the size of shares for various values of $(n, t)$ in Table 1.[4]

**Comparison with [GK18a].** When compared to the result of [GK18a] which could support thresholds $t \geq 2$, our construction can only support threshold $t \geq 4$. However, getting a rate $> 0$ non-malleable secret sharing scheme for threshold $t = 2$ would imply a 2-split-state non-malleable code with rate $> 0$ which is a major open problem. For the case of $t = 3$, though we know constructions of 3-split-state non-malleable codes with rate $> 0$ [KOS18, GMW17], they do not satisfy the privacy property of a 3-out-of-3 secret sharing scheme. In particular, given two states

[4] 812 bits is the minimal message length that gives 80 bits of security.

| (# of Parties, Threshold) | Secret Length (in bits) | Share Size (in KB) |
|---|---|---|
| $(7,4)$ | 812 | 273.73 |
| $(9,5)$ | 812 | 399.85 |
| $(25,13)$ | 812 | 1757.53 |
| $(100,51)$ | 812 | $12.34 \times 10^3$ |
| $(7,4)$ | 1024 | 345.19 |
| $(9,5)$ | 1024 | 504.24 |
| $(25,13)$ | 1024 | 2216.40 |
| $(100,51)$ | 1024 | $15.56 \times 10^3$ |

**Table 1**: Share sizes for simulation error of at most $2^{-80}$.

of the codeword, some information about the message is leaked. Thus, getting a 3-out-of-$n$ non-malleable secret sharing scheme with rate $> 0$ seems out of reach of the current techniques and we leave this as an open problem.

### 1.1.2 Multiple Tampering

We initiate the study of non-malleable secret sharing under multiple tampering. Here, the shares can be subject to multiple (possibly different) tampering functions and we require that the joint distribution of the reconstructed secrets to be independent of $s$. For this stronger security notion, we first prove a negative result that states that a non-malleable secret sharing cannot exist when the number of tamperings (also called as the tampering degree) is apriori unbounded. This result generalizes a similar result for the case of a split-state non-malleable codes. Formally,

**Theorem 1.3** *For any $n, t \in \mathbb{N}$, there does not exist a $t$-out-of-$n$ non-malleable secret sharing scheme against $\mathcal{F}_{ind}$ that can support an apriori unbounded tampering degree.*

When the tampering degree is apriori bounded, we get constructions of threshold non-malleable secret sharing scheme. Formally,

**Theorem 1.4** *For any $n, t \geq 4$, and $\mathsf{K} \in \mathbb{N}$, there exists a $t$-out-of-$n$ non-malleable secret sharing scheme with tampering degree $\mathsf{K}$ for sharing $m$-bit secrets for a large enough[5] $m$ against $\mathcal{F}_{ind}$ with rate $= \Theta(\frac{1}{\mathsf{K}^3 t \log^2 n})$ and simulation error $2^{-m^{\Omega(1)}}$. The running time of the sharing and reconstruction algorithms are polynomial in $n$ and $m$.*

### 1.1.3 General Access Structures

We extend our techniques used in the proof of Theorems 1.1,1.4 to give constructions of non-malleable secret sharing scheme for more general monotone access structures rather than just threshold structures. Before we state our result, we give some definitions.

**Definition 1.5** *An access structure $\mathcal{A}$ is said to be monotone if for any set $S \in A$, any superset of $S$ is also in $\mathcal{A}$. A monotone access structure $\mathcal{A}$ is said to be 4-monotone if for any set $S \in \mathcal{A}$, $|S| \geq 4$.*

[5] See the main body for the precise statement.

We also give the definition of a minimal authorized set.

**Definition 1.6** *For a monotone access structure $\mathcal{A}$, a set $S \in \mathcal{A}$ is a minimal authorized set if any strict subset of $S$ is not in $\mathcal{A}$. We denote $t_{max}$ to be $\max |S|$ where $S$ is a minimal authorized set of $\mathcal{A}$.*

We now state our extension to general access structures.

**Theorem 1.7** *For any $n, \mathsf{K} \in \mathbb{N}$ and 4-monotone access structure $\mathcal{A}$, if there exists a statistically private (with privacy error $\varepsilon$) secret sharing scheme for $\mathcal{A}$ that can share $m$-bit secrets for a large enough $m$ with rate $R$, there exists a non-malleable secret sharing scheme for sharing $m$-bit secrets for the same access structure $\mathcal{A}$ with tampering degree $\mathsf{K}$ against $\mathcal{F}_{ind}$ with rate $\Theta(\frac{R}{\mathsf{K}^3 t_{\max} \log^2 n})$ and simulation error $\varepsilon + 2^{-m^{\Omega(1)}}$.*

Thus, starting with a secret sharing scheme for monotone span programs [KW93] or for more general access structures [LV18], we get non-malleable secret sharing schemes for the same access structures with comparable rate.

**Comparison with [GK18b].** In the prior work [GK18b], the rate of the non-malleable secret sharing for general access structures also depended on the length of the message and thus, even when $R$ is constant, their construction could only achieve a rate of 0. However, unlike our construction, they could support all monotone access structures (and not just 4-monotone) and they could even start with a computational secret sharing scheme for an access structure $\mathcal{A}$ and convert it to a non-malleable secret sharing scheme for $\mathcal{A}$.

**Concurrent Work.** In a concurrent and independent work, Aggarwal et al. [ADN$^+$18] consider the multiple tampering model and give constructions of non-malleable secret sharing for general access structures in this model. There are three main differences between our work and their work. Firstly, the rate of their construction asymptotically tends to 0 even for the threshold case. However, the rate of our construction is greater than 0 when we instantiate the compiler with a rate $> 0$ secret sharing scheme. Secondly, their work considers a stronger model wherein each tampering function can choose a different reconstruction set. We prove the security of our construction in a weaker model wherein the reconstruction set is the same for each tampering function. We note that the impossibility result for unbounded tampering holds even if the reconstruction set is the same. Thirdly, their construction can give non-malleable secret sharing scheme for any 3-monotone access structure whereas our construction can only work for 4-monotone access structure. In another concurrent and independent work, Kumar et al. [KMS18] gave a construction of non-malleable secret sharing in a stronger model where the tampering functions might obtain bounded leakage from the other shares.

# 2 Our Techniques

In this section, we give a high level overview of the techniques used to obtain our results.

### 2.1 Rate Improvement

**Goyal and Kumar [GK18a] approach.** We first give a brief overview of the construction of threshold non-malleable secret sharing of Goyal and Kumar [GK18a] and then explain why it could achieve only a rate of 0. At a high level, Goyal and Kumar start with any 2-split-state non-malleable code and convert it into a $t$-out-of-$n$ non-malleable secret sharing scheme. We only explain their construction for the case when $t \geq 3$, and for the case of $t = 2$, they gave a slightly different construction. For the case when $t \geq 3$, the sharing procedure does the following. The secret is first encoded using a 2-split-state non-malleable code to obtain the two states $\mathsf{L}$ and $\mathsf{R}$. $\mathsf{L}$ is now shared using any $t$-out-of-$n$ secret sharing scheme, say Shamir's secret sharing to get the shares $\mathsf{SL}_1, \ldots, \mathsf{SL}_n$ and $\mathsf{R}$ is shared using a 2-out-of-$n$ local leakage resilient secret sharing scheme to get the shares $\mathsf{SR}_1, \ldots, \mathsf{SR}_n$. The share corresponding to party $i$ includes $(\mathsf{SL}_i, \mathsf{SR}_i)$. To recover the secret given at least $t$ shares, the parties first use the recovery procedures of the threshold secret sharing scheme and local leakage resilient secret sharing scheme to recover $\mathsf{L}$ and $\mathsf{R}$ respectively. Later, the secret is obtained by decoding $\mathsf{L}$ and $\mathsf{R}$ using the decoding procedure of the non-malleable code. The correctness of the construction is straightforward and to argue secrecy, it can been seen that given any set of $t-1$ shares, $\mathsf{L}$ is perfectly hidden and this follows from the security of Shamir's secret sharing. Now, using the fact that any 2-split-state non-malleable code is a 2-out-of-2 secret sharing scheme, it can be shown that the right state $\mathsf{R}$ statistically hides the secret.

To argue the non-malleability of this construction, Goyal and Kumar showed that any tampering attack on the secret sharing scheme can be reduced to a tampering attack on the underlying 2-split-state non-malleable code. The main challenge in designing such a reduction is that the tampering functions against the underlying non-malleable code must be split-state, meaning that the tampering function against $\mathsf{L}$ (denoted by $f$) must be independent of $\mathsf{R}$ and the tampering function against $\mathsf{R}$ (denoted by $g$) must be independent of $\mathsf{L}$. To make the tampering function $g$ to be independent of $\mathsf{L}$, [GK18a] made use of the fact that there is an inherent difference in the parameters used for secret sharing $\mathsf{L}$ and $\mathsf{R}$. Specifically, since $\mathsf{R}$ is shared using a 2-out-of-$n$ secret sharing scheme, the tampered right state $\widetilde{\mathsf{R}}$ can be recovered from any two tampered shares, say $\widetilde{\mathsf{SR}}_1, \widetilde{\mathsf{SR}}_2$. Now, since $\mathsf{L}$ is shared using a $t$-out-of-$n$ secret sharing scheme and $t \geq 3$, the shares $\mathsf{SL}_1$ and $\mathsf{SL}_2$ information theoretically provides no information about $\mathsf{L}$. This, in particular means that we can fix the shares $\mathsf{SL}_1$ and $\mathsf{SL}_2$ independent of $\mathsf{L}$ and the tampering function $g$ could use these fixed shares to output the tampered right state $\widetilde{\mathsf{R}}$. Now, when $f$ is given the actual $\mathsf{L}$, it can sample $\mathsf{SL}_3, \ldots, \mathsf{SL}_n$ as a valid secret sharing of $\mathsf{L}$ that is consistent with the fixed $\mathsf{SL}_1, \mathsf{SL}_2$. This allowed them to argue one-sided independence i.e., $g$ is independent of $\mathsf{L}$. On the other hand, making the tampering function $f$ to be independent of $\mathsf{R}$ is a lot trickier. This is because any two shares information theoretically fixes $\mathsf{R}$ and in order to recover $\widetilde{\mathsf{L}}$, we need at least $t$ $(\geq 3)$ shares. Hence, we may not be able to argue that $f$ is independent of $\mathsf{R}$. To argue this independence, Goyal and Kumar used the fact that $\mathsf{R}$ is shared using a *local leakage resilient* secret sharing scheme. In particular, they made the size of $\mathsf{SR}_i$ to be much larger than the size of $\mathsf{SL}_i$ and showed that even when we leak $|\mathsf{SL}_i|$ bits from each share $\mathsf{SR}_i$, $\mathsf{R}$ is still statistically hidden. This allowed them to define leakage functions $\mathsf{leak}_1, \ldots, \mathsf{leak}_n$ where $\mathsf{leak}_i$ had $\mathsf{SL}_i$ hardwired in its description, it applies the tampering function on $(\mathsf{SL}_i, \mathsf{SR}_i)$ and outputs the tampered $\widetilde{\mathsf{SL}}_i$. Now, from the secrecy of the local leakage resilient secret sharing scheme, the distribution $\widetilde{\mathsf{SL}}_1, \ldots, \widetilde{\mathsf{SL}}_n$ (which completely determines $\widetilde{\mathsf{L}}$) is independent of $\mathsf{R}$ and thus $\widetilde{\mathsf{L}}$ is independent of $\mathsf{R}$. This allowed them to obtain two-sided independence.

A drawback of this approach is that the rate of this scheme is at least as bad as that of the underlying 2-split-state non-malleable code. As mentioned before, obtaining a 2-split-state non-malleable code with rate $> 0$ is a major open problem. Thus, this construction could only achieve a rate of 0.

**Our Approach.** While constructing 2-split-state non-malleable code with rate $> 0$ has been notoriously hard, significant progress has been made for the case of 3-split-state non-malleable codes. Very recently, independent works of Gupta et al. [GMW17] and Kanukurthi et al. [KOS18] gave constructions of 3-split-state non-malleable codes with an explicit constant rate. The main idea behind our rate-improved construction is to use a constant rate, 3-split-state non-malleable code instead of a rate 0, 2-split-state non-malleable code. To be more precise, we first encode the secret using a 3-split-state non-malleable code to get the three states $(\mathsf{L}, \mathsf{C}, \mathsf{R})$. We then share the first state $\mathsf{L}$ using a $t$-out-of-$n$ secret sharing scheme to get $(\mathsf{SL}_1, \ldots, \mathsf{SL}_n)$ as before. Then, we share $\mathsf{C}$ using a $t_1$-out-of-$n$ secret sharing scheme to get $(\mathsf{SC}_1, \ldots, \mathsf{SC}_n)$ and $\mathsf{R}$ using a $t_2$-out-of-$n$ secret sharing scheme to get $(\mathsf{SR}_1, \ldots, \mathsf{SR}_n)$. Here, $t_1, t_2$ are some parameters that we will fix later. The share corresponding to party $i$ includes $(\mathsf{SL}_i, \mathsf{SC}_i, \mathsf{SR}_i)$. While the underlying intuition behind this idea is natural, proving that this construction is a non-malleable secret sharing scheme faces several barriers which we elaborate below.

**First Challenge.** The first barrier that we encounter is, unlike a 2-split-state non-malleable code which is always a 2-out-of-2 secret sharing scheme, a 3-split-state non-malleable code may not be a 3-out-of-3 secret sharing scheme. In particular, we will not be able use the [GK18a] trick of sharing the 3-states using secret sharing schemes with different thresholds to gain one-sided independence. This is because given $t - 1$ shares, complete information about two states will be revealed, and we could use these two states to gain some information about the underlying message. Thus, the privacy of the scheme breaks down. Indeed, as mentioned in the introduction, the constructions of Kanukurthi et al. [KOS18] and Gupta et al. [GMW17] are not 3-out-of-3 secret sharing schemes.

The main trick that we use to solve this challenge is that, while these constructions [KOS18, GMW17] are not 3-out-of-3 secret sharing schemes, we observe that there exist two states (let us call them $\mathsf{C}$ and $\mathsf{R}$) such that these two states statistically hide the message. This means that we can potentially share these two states using secret sharing schemes with smaller thresholds and may use it to argue one-sided independence.

**Second Challenge.** The second main challenge is in ensuring that the tampering functions we design for the underlying 3-split-state non-malleable code are indeed split-state. Let us call the tampering functions that tamper $\mathsf{L}, \mathsf{C}$, and $\mathsf{R}$ as $f, g$, and $h$ respectively. To argue that $f, g$ and $h$ are split-state, we must ensure $f$ is independent of $\mathsf{C}$ and $\mathsf{R}$ and similarly, $g$ is independent of $\mathsf{L}$ and $\mathsf{R}$ and $h$ is independent of $\mathsf{L}$ and $\mathsf{C}$. For the case of 2-split-state used in the prior work, this independence was achieved by using secret sharing with different thresholds and relying on the leakage resilience property. For the case of 3-split-state, we need a more sophisticated approach of *stratifying* the three secret sharing schemes so that we avoid circular dependence in the parameters. We now elaborate more on this solution.

To make $g$ and $h$ to be independent of $\mathsf{L}$, we choose the thresholds $t_1$ and $t_2$ to be less than $t$. This allows us to fix a certain number of shares independent of $\mathsf{L}$ and use these shares to extract $\widetilde{\mathsf{C}}$ and $\widetilde{\mathsf{R}}$. Similarly, to make $h$ to be independent of $\mathsf{C}$, we choose the threshold $t_2 < t_1$. This again allows us to fix certain shares $\mathsf{C}$ and use them to extract $\widetilde{\mathsf{R}}$. Thus, by choosing $t > t_1 > t_2$, we could achieve something analogous to one-sided independence. Specifically, we achieved independence of $g$ from $\mathsf{L}$ and independence of $h$ from $(\mathsf{L}, \mathsf{C})$. For complete split-state property, we still need to make sure that $f$ is independent of $(\mathsf{C}, \mathsf{R})$ and $g$ is independent of $\mathsf{R}$. To make the tampering function $f$ to be independent of $\mathsf{C}$, we rely on the local leakage resilience property of the $t_1$-out-of-$n$ secret sharing scheme. That is, we make the size of the shares $\mathsf{SC}_i$ to be much larger than $\mathsf{SL}_i$ such that, in spite of leaking $|\mathsf{SL}_i|$ bits from each share $\mathsf{SC}_i$, the secrecy of $\mathsf{C}$ is maintained. We can use this to show that the joint distribution $(\widetilde{\mathsf{SL}}_1, \ldots, \widetilde{\mathsf{SL}}_n)$ (which completely determines $\widetilde{\mathsf{L}}$) is independent of $\mathsf{C}$. Now, to argue that both $f$ and $g$ are independent of $\mathsf{R}$, we rely on the local leakage resilience property of the $t_2$-out-of-$n$ secret sharing scheme. That is, we make the shares of $\mathsf{SR}_i$ to be much larger than $(\mathsf{SL}_i, \mathsf{SC}_i)$ so that, in spite of leaking $|\mathsf{SL}_i| + |\mathsf{SC}_i|$ bits from each share $\mathsf{SR}_i$, the secrecy of $\mathsf{R}$ is maintained. We then use this property to argue that the joint distribution $(\widetilde{\mathsf{SL}}_1, \widetilde{\mathsf{SC}}_1), \ldots, (\widetilde{\mathsf{SL}}_n, \widetilde{\mathsf{SC}}_n)$ is independent of $\mathsf{R}$. Thus, the idea of stratifying the three threshold secret sharing schemes with different parameters as described above allows to argue that $f$, $g$ and $h$ are split-state. As we will later see, this technique of stratification is very powerful and it allows us to easily extend this construction to more general monotone access structures.

**Third Challenge.** The third and the more subtle challenge is the following. To reduce the tampering attack on the secret sharing scheme to a tampering attack on the underlying non-malleable code, we must additionally ensure *consistency* i.e., the tampered message output by the split-state functions must be statistically close to the message output by the tampering experiment of the underlying secret sharing scheme. To illustrate this issue in some more detail, let us consider the tampering functions $f$ and $g$ in the construction of Goyal and Kumar [GK18a] for the simple case when $n = t = 3$. Recall that the tampering function $g$ samples $\mathsf{SR}_1, \mathsf{SR}_2$ such that it is a valid 2-out-of-$n$ secret sharing of $\mathsf{R}$ and uses the fixed $\mathsf{SL}_1, \mathsf{SL}_2$ (independent of $\mathsf{L}$) to extract the tampered $\widetilde{\mathsf{R}}$ from $(\widetilde{\mathsf{SR}}_1, \widetilde{\mathsf{SR}}_2)$. However, note that $g$ cannot use any valid secret sharing of $\mathsf{SR}_1, \mathsf{SR}_2$ of $\mathsf{R}$. In particular, it must also satisfy the property that the tampering function applied on $\mathsf{SL}_1$, $\mathsf{SR}_1$ gives the exact same $\widetilde{\mathsf{SL}}_1$ that $f$ uses in the reconstruction (a similar condition for position 2 must be satisfied). This is crucial, as otherwise there might be a difference in the distributions of the tampered message output by the split-state functions and the message output in the tampering experiment of the secret sharing scheme. In case there is a difference, we cannot hope to use the adversary against the non-malleable secret sharing to break the underlying non-malleable code. This example illustrates this issue for a simple case when $t = n = 3$. To ensure consistency for larger values of $n$ and $t$, Goyal and Kumar fixed $(\mathsf{SL}_1, \ldots, \mathsf{SL}_{t-1})$ (instead of just fixing $\mathsf{SL}_1, \mathsf{SL}_2$) and the function $g$ ensures consistency of each of the tampered shares $\widetilde{\mathsf{SL}}_1, \ldots, \widetilde{\mathsf{SL}}_{t-1}$. However, this approach completely fails when we move to 3 states. For the case of 3-states, the tampering function, say $h$, must sample $\mathsf{SR}_1, \ldots, \mathsf{SR}_n$ such that it is consistent with $\widetilde{\mathsf{SL}}_1, \ldots, \widetilde{\mathsf{SL}}_{t-1}$ used by $f$. However, even to check this consistency, $h$ would need the shares $\mathsf{SC}_1, \ldots, \mathsf{SC}_{t-1}$ which completely determines $\mathsf{C}$. In this case, we cannot argue that $h$ is independent of $\mathsf{C}$.

To tackle this challenge, we deviate from the approach of Goyal and Kumar [GK18a] and have a new proof strategy that ensures consistency and at the same time maintains the split-state property. In this strategy, we only fix the values $(\mathsf{SL}_1, \mathsf{SL}_2, \mathsf{SL}_3)$ for the first secret sharing scheme, $(\mathsf{SC}_1, \mathsf{SC}_2)$ for the second secret sharing scheme and fix $\mathsf{SR}_3$ for the third secret sharing scheme. Note that we consider $t \geq 4$, $t_1 \geq 3$ and $t_2 \geq 2$ and thus, the fixed shares are independent of $\mathsf{L}$, $\mathsf{C}$, and $\mathsf{R}$ respectively.[6] We design our split-state functions in such a way that the tampering function $f$ need

[6] This is the reason why we could only achieve thresholds $t \geq 4$.

not do any consistency checks, the tampering function $g$ has to do the consistency check only on $\widetilde{\mathsf{SL}}_3$ (which it can do since $\mathsf{SL}_3$ and $\mathsf{SR}_3$ are fixed) and the function $h$ needs to do a consistency check only on $\{\widetilde{\mathsf{SL}}_i, \widetilde{\mathsf{SC}}_i\}_{i\in[1,2]}$ (which it can do since $\mathsf{SL}_1, \mathsf{SC}_1, \mathsf{SL}_2, \mathsf{SC}_2$ are fixed). This approach of reducing the number of checks to maintain consistency helps us in arguing independence between the tampering functions. However, this approach creates additional problems in extracting $\widetilde{\mathsf{L}}$ as the tampering function $f$ needs to use the shares $(\mathsf{SR}_4, \ldots, \mathsf{SR}_n)$ and $(\mathsf{SC}_4, \ldots, \mathsf{SC}_n)$ (which completely determines $\mathsf{C}$ and $\mathsf{R}$ respectively). We solve this by letting $f$ extract $\widetilde{\mathsf{L}}$ using shares of some arbitrary values of $\mathsf{C}$ and $\mathsf{R}$ and we then use the leakage resilience property to ensure that the outputs in the split-state tampering experiment and the secret sharing tampering experiment are statistically close.

**Completing the Proof.** This proof strategy helps us in getting a rate $> 0$ construction of a $t$-out-of-$n$ non-malleable secret sharing scheme for $t \geq 4$. However, there is one crucial block that is still missing. Goyal and Kumar [GK18a] only gave a construction of 2-out-of-$n$ local leakage resilient secret sharing scheme. And, for this strategy to work we also need a construction of $t_1$-out-of-$n$ local leakage resilient secret sharing scheme for some $t_1 > 2$. As mentioned in the introduction, the recent work by Benhamouda et al. [BDIR18] only gives a construction of local leakage resilient secret sharing when the threshold value is large (in particular, $n - o(\log n)$). To solve this, we give an efficient construction of a $t$-out-of-$n$ local leakage resilient secret sharing scheme when $t$ is a constant. This is in fact sufficient to get a rate $> 0$ construction of non-malleable secret sharing scheme. We now give details on the techniques used in this construction.

**Local Leakage Resilient Secret Sharing Scheme.** The starting point of our construction is the 2-out-of-2 local leakage resilient secret sharing from the work of Goyal and Kumar [GK18a] based on the inner product two-source extractor [CG88]. We first extend it to a $k$-out-of-$k$ local leakage resilient secret sharing scheme for any arbitrary $k$. Let us now illustrate this for the case when $k$ is even i.e., $k = 2p$. To share a secret $s$, we first additively secret share $s$ into $s_1, \ldots, s_p$ and we encode each $s_i$ using the 2-out-of-2 leakage resilient secret sharing scheme to obtain the shares $(\mathsf{share}_{2i-1}, \mathsf{share}_{2i})$. We then give $\mathsf{share}_i$ to party $i$ for each $i \in [k]$. Note that given $t-1$ shares, at most $p-1$ additive secret shares can be revealed. We now rely on the local leakage resilience property of the 2-out-of-2 secret sharing to argue that the final additive share is hidden even when given bounded leakage from the last share. This helps us in arguing the $k$-out-$k$ local leakage resilience property. The next goal is to extend this to a $k$-out-of-$n$ secret sharing scheme. Since we are interested in getting good rate, we should not increase the size of the shares substantially. A naïve way of doing this would be to share the secret $\binom{n}{k}$ times (one for each possible set of $k$-parties) using the $k$-out-of-$k$ secret sharing scheme and give the respective shares to the parties. The size of each share in this construction would blow up by a factor $\binom{n}{k-1}$ when compared to the $k$-out-of-$k$ secret sharing scheme. Though, this is polynomial in $n$ when $k$ is a constant, this is clearly sub-optimal when $n$ is large and would result in bad concrete parameters. We note that Goyal and Kumar [GK18a] used a similar approach to obtain a 2-out-of-$n$ local leakage resilient secret sharing.

In this work, we use a very different approach to construct a $k$-out-of-$n$ local leakage resilient secret sharing from a $k$-out-of-$k$ local leakage resilient secret sharing. The main advantage of this transformation is that it is substantially more rate efficient than the naïve solution. Our transformation makes use of combinatorial objects called as perfect hash functions [FK84].[7] A family of functions mapping $\{1, \ldots, n\}$ to $\{1, \ldots, k\}$ is said to be a perfect hash function family if for every set $S \subseteq [n]$ of size at most $k$, there exists at least one function in the family that is injective on $S$. Let us now illustrate how this primitive is helpful in extending a $k$-out-of-$k$ secret sharing scheme to a $k$-out-of-$n$ secret sharing scheme. Given a perfect hash function family $\{h_i\}_{i \in [\ell]}$ of size $\ell$, we share the secret $s$ independently $\ell$ times using the $k$-out-of-$k$ secret sharing scheme to obtain $(\mathsf{share}^i_1, \ldots, \mathsf{share}^i_k)$ for each $i \in [\ell]$. We now set the shares corresponding to party $i$ as $(\mathsf{share}^1_{h_1(i)}, \ldots, \mathsf{share}^\ell_{h_\ell(i)})$. To recover the secret from some set of $k$ shares given by $S = \{s_1, \ldots, s_k\}$, we use the following strategy. Given any subset $S$ of size $k$, perfect hash function family guarantees that there is at least one index $i \in [\ell]$ such that $h_i$ is injective on $S$. We can now use $\{\mathsf{share}^i_{h_i(s_1)}, \ldots, \mathsf{share}^i_{h_i(s_k)}\} = \{\mathsf{share}^i_1, \ldots, \mathsf{share}^i_k\}$ to recover the secret using the reconstruction procedure of the $k$-out-of-$k$ secret sharing.

We show that this transformation additionally preserves local leakage resilience. In particular, if we start with a $k$-out-of-$k$ local leakage resilient secret sharing scheme then we obtain a $k$-out-of-$n$ local leakage resilient secret sharing. The size of each share in our $k$-out-of-$n$ leakage resilient secret sharing scheme is $\ell$ times the share size of $k$-out-of-$k$ secret sharing scheme. Thus, to minimize rate we must minimize the size of the perfect hash function family. Constructing perfect hash function family of minimal size for all $k \in \mathbb{N}$ is an interesting and a well-known open problem in combinatorics. In this work, we give an efficient randomized construction (with good concrete parameters) of a perfect hash function family for a constant $k$ with size $O(\log n + \log(1/\varepsilon))$ where $\varepsilon$ is the error probability. Alternatively, we can also use the explicit construction (which is slightly less efficient when compared to the randomized construction) of size $O(\log n)$ (when $k$ is a constant) given by Alon et al. [AYZ95]. Combining either the randomized/explicit construction of perfect hash function family with a construction of $k$-out-of-$k$ local leakage resilient secret sharing scheme, we get an efficient construction of $k$-out-of-$n$ local leakage resilient secret sharing scheme when $k$ is a constant.

## 2.2 Multiple Tampering

We also initiate the study of non-malleable secret sharing under multiple tamperings. As discussed in the introduction, this is a much stronger model when compared to that of a single tampering.

**Negative Result.** We first show that when the number of tampering functions that can maul the secret sharing scheme is apriori unbounded, there does not exist any threshold non-malleable secret sharing scheme. This generalizes a similar result for the case of split-state non-malleable code (see [GLM$^+$04,FMNV14] for details) and the main idea is inspired by these works. The underlying intuition behind the negative result is simple: we come up with a set of tampering functions such that each tampering experiment leaks one bit of a share. Now, given the outcomes of $t \cdot s$ such tampering experiments where $s$ is the size of the share, the distinguisher can clearly learn every bit of $t$ shares and thus, learn full information about the underlying secret and break non-malleability.

For the tampering experiment to leak one bit of the share of party $i$, we use the following simple strategy. Let us fix an authorized set of size $t$ say, $\{1, \ldots, t\}$. We choose two sets of shares:

[7] We note that using perfect hash function families for constructing threshold secret sharing scheme is not new (see [Bla99,SNW01] for a comprehensive discussion). However, to the best of our knowledge, this is the first application of this technique to construct local leakage resilient secret sharing scheme.

$\{\mathsf{share}_1, \ldots, \mathsf{share}_i, \ldots, \mathsf{share}_t\}$ and $\{\mathsf{share}_1, \ldots, \mathsf{share}'_i, \ldots, \mathsf{share}_t\}$ such that they reconstruct to two different secrets. Note that the privacy of a secret sharing scheme guarantees that such shares must exist. Whenever the particular bit of the share of party $i$ is 1, the tampering function $f_i$ outputs $\mathsf{share}'_i$ whereas the other tampering functions, say $f_j$ will output $\mathsf{share}_j$. On the other hand, if the particular bit is 0 then the tampering function $f_i$ outputs $\mathsf{share}_i$ and the other tampering functions still output $\mathsf{share}_j$. Observe that the reconstructed secret in the two cases reveals the particular bit of the share of party $i$. We can use a similar strategy to leak every bit of all the $t$ shares which completely determine the secret.

**Positive Result.** We complement the negative result by showing that when the number of tamperings is apriori bounded, we can obtain an efficient construction of a threshold non-malleable secret sharing scheme. A natural approach would be to start with a split-state non-malleable code that is secure against bounded tamperings and convert it into a non-malleable secret sharing scheme. To the best of our knowledge, the only known construction of split-state non-malleable code that is secure in the presence of bounded tampering is that of Chattopadhyay et al. [CGL16]. However, the rate of this code is 0 even when we restrict ourselves to just two tamperings. In order to achieve a better rate, we modify the constructions of Kanukurthi et al. [KOS18] and Gupta et al. [GMW17] such that we obtain a 3-split-state non-malleable code that secure in the setting of bounded tampering. The rate of this construction is $O(\frac{1}{k})$ where $k$ is the apriori bound on the number of tamperings. Fortunately, even in this construction, we still maintain the property that there exists two states that statistically hide the message. We then prove that the same construction described earlier is a secure non-malleable secret sharing under bounded tampering when we instantiate the underlying code with a bounded tampering secure 3-split-state non-malleable codes.

### 2.3 General Access Structures

To obtain a secret sharing scheme for more general access structures, we start with any statistically secure secret sharing scheme for that access structure, and use it to share $\mathsf{L}$ instead of using a threshold secret sharing scheme. We require that the underlying access structure to be 4-monotone so that we can argue the privacy of our scheme. Recall that a 4-monotone access structure is one in which the size of every set in the access structure is at least 4. Even in this more general case, the technique of stratifying the secret sharing schemes allows us to prove non-malleability in almost an identical fashion to the case of threshold secret sharing. We remark that the work of [GK18b] which gave constructions of non-malleable secret sharing scheme for general monotone access structures additionally required their local leakage resilient secret sharing scheme to satisfy a security property called as strong local leakage resilience. Our construction does not require this property and we show that "plain" local leakage resilience is sufficient for extending to more general monotone access structures.

**Organization.** We give the definitions of non-malleable secret sharing and non-malleable codes in Section 3. In Section 4, we present the construction of the $k$-out-of-$n$ leakage resilient secret sharing scheme. In Section 5, we describe our rate-efficient threshold non-malleable secret sharing scheme for the single tampering. We give the impossibility result for unbounded many tamperings in Appendix 6. Finally, in Section 7, we describe our result on non-malleable secret sharing for general access structures against multiple bounded tampering. Note that the result in Section 7 implicitly captures the result for threshold non-malleable secret sharing against bounded tampering. We present this more general result for ease of exposition.

# 3 Preliminaries

**Notation.** We use capital letters to denote distributions and their support, and corresponding lowercase letters to denote a sample from the same. Let $[n]$ denote the set $\{1, 2, \ldots, n\}$, and $U_r$ denote the uniform distribution over $\{0,1\}^r$. For any $i \in [n]$, let $x_i$ denote the symbol at the $i$-th co-ordinate of $x$, and for any $T \subseteq [n]$, let $x_T \in \{0,1\}^{|T|}$ denote the projection of $x$ to the co-ordinates indexed by $T$. We write $\circ$ to denote concatenation.

Standard definitions of min-entropy and statistical distance are given below.

**Definition 3.1 (Min-entropy)** *The min-entropy of a source $X$ is defined to be*

$$H_\infty(X) = \min_{s \in \text{support}(X)} \{\log(1/\Pr[X = s])\}$$

*A $(n,k)$-source is a distribution on $\{0,1\}^n$ with min-entropy $k$.*

**Definition 3.2 (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:*

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

*$D_1$ is $\varepsilon$-close to $D_2$ if $|D_1 - D_2| \leq \varepsilon$.*

We will use the notation $D_1 \approx_\varepsilon D_2$ to denote that the statistical distance between $D_1$ and $D_2$ is at most $\varepsilon$.

**Lemma 3.3 (Triangle Inequality)** *If $D_1 \approx_{\varepsilon_1} D_2$ and $D_2 \approx_{\varepsilon_2} D_3$ then $D_1 \approx_{\varepsilon_1+\varepsilon_2} D_3$.*

We now recall the definition of (average) conditional min-entropy [DORS08].

**Definition 3.4 ( [DORS08])** *The average conditional min-entropy is defined as*

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

We recall some results on conditional min-entropy from [DORS08].

**Lemma 3.5 ( [DORS08])** *If a random variable $B$ can take at most $\ell$ values, then $\widetilde{H}_\infty(A|B) \geq H_\infty(A) - \log \ell$.*

**Seeded Extractors.** We now recall the definition of a strong seeded extractor.

**Definition 3.6 (Strong seeded extractor)** *A function* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *is called a strong seeded extractor for min-entropy* $k$ *and error* $\varepsilon$ *if for any* $(n,k)$*-source* $X$ *and an independent uniformly random string* $U_d$*, we have*

$$|\mathrm{Ext}(X, U_d) \circ U_d - U_m \circ U_d| < \varepsilon,$$

*where* $U_m$ *is independent of* $U_d$*.*

An average case seeded extractor requires that if a source $X$ has average case conditional min-entropy $\tilde{H}_\infty(X|Z) \geq k$ then the output of the extractor is uniform even when $Z$ is given. We recall the following lemma from [DORS08] which states that every strong seeded extractor is also an average-case strong extractor.

**Lemma 3.7 ( [DORS08])** *For any* $\delta > 0$*, if* $\mathrm{Ext}$ *is a* $(k, \varepsilon)$*-strong seeded extractor then it is also a* $\left(k + \log\left(\frac{1}{\delta}\right), \varepsilon + \delta\right)$ *average case strong extractor.*

Guruswami et al. [GUV09] gave a construction of (strong) seeded extractor with near optimal parameters and we recall the result below.

**Theorem 3.8 ( [GUV09])** *For any constant* $\alpha > 0$*, and all integers* $n, k > 0$ *there exists a polynomial time computable strong seeded extractor* $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ *with* $d = O(\log n + \log(\frac{1}{\varepsilon}))$ *and* $m = (1 - \alpha)k$*.*

## 3.1 Threshold Non-Malleable Secret Sharing Scheme

We first give the definition of a sharing function, then define a threshold secret sharing scheme and finally give the definition of a threshold non-malleable secret sharing. These three definitions are taken verbatim from [GK18a]. In Section 7, we define non-malleable secret sharing for more general monotone access structures.

**Definition 3.9 (Sharing Function)** *Let* $[n] = \{1, 2, \ldots, n\}$ *be a set of identities of* $n$ *parties. Let* $\mathcal{M}$ *be the domain of secrets. A sharing function* $\mathsf{Share}$ *is a randomized mapping from* $\mathcal{M}$ *to* $\mathcal{S}_1 \times \mathcal{S}_2 \times \ldots \times \mathcal{S}_n$*, where* $\mathcal{S}_i$ *is called the domain of shares of party with identity* $i$*. A dealer distributes a secret* $m \in \mathcal{M}$ *by computing the vector* $\mathsf{Share}(m) = (\mathsf{S}_1, \ldots, \mathsf{S}_n)$*, and privately communicating each share* $\mathsf{S}_i$ *to the party* $i$*. For a set* $T \subseteq [n]$*, we denote* $\mathsf{Share}(m)_T$ *to be a restriction of* $\mathsf{Share}(m)$ *to its* $T$ *entries.*

**Definition 3.10 ($(t, n, \varepsilon_c, \varepsilon_s)$-Secret Sharing Scheme)** *Let* $\mathcal{M}$ *be a finite set of secrets, where* $|\mathcal{M}| \geq 2$*. Let* $[n] = \{1, 2, \ldots, n\}$ *be a set of identities (indices) of* $n$ *parties. A sharing function* $\mathsf{Share}$ *with domain of secrets* $\mathcal{M}$ *is a* $(t, n, \varepsilon_c, \varepsilon_s)$*-secret sharing scheme if the following two properties hold :*

- ***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* $\mathsf{Rec} : \otimes_{i \in T} \mathcal{S}_i \to \mathcal{M}$ *such that for every* $m \in \mathcal{M}$*,*

  $$\Pr[\mathsf{Rec}(\mathsf{Share}(m)_T) = m] = 1 - \varepsilon_c$$

  *where the probability is over the randomness of the* $\mathsf{Share}$ *function. We will slightly abuse the notation and denote* $\mathsf{Rec}$ *as the reconstruction procedure that takes in* $T$ *and* $\mathsf{Share}(m)_T$ *where* $T$ *is of size at least* $t$ *and outputs the secret.*

- ***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 $m_0, m_1 \in M$, for any distinguisher $D$ with output in $\{0,1\}$, the following holds :*

$$|\Pr[D(\mathsf{Share}(m_0)_U) = 1] - \Pr[D(\mathsf{Share}(m_1)_U) = 1]| \leq \varepsilon_s$$

*We define the rate of the secret sharing scheme as*

$$\lim_{|m| \to \infty} \frac{|m|}{\max_{i \in [n]} |\mathsf{Share}(m)_i|}$$

**Definition 3.11 (Threshold Non-Malleable Secret Sharing [GK18a])** *Let* $(\mathsf{Share}, \mathsf{Rec})$ *be a* $(t, n, \varepsilon_c, \varepsilon_s)$*-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 tampered distribution* $\mathsf{Tamper}_m^{f,T}$ *as* $\mathsf{Rec}(f(\mathsf{Share}(m))_T)$ *where the randomness is over the sharing function* $\mathsf{Share}$*. We say that the* $(t, n, \varepsilon_c, \varepsilon_s)$*-secret sharing scheme,* $(\mathsf{Share}, \mathsf{Rec})$ *is* $\varepsilon'$*-non-malleable w.r.t.* $\mathcal{F}$ *if for each* $f \in \mathcal{F}$ *and any authorized set* $T$ *consisting of* $t$ *indices, there exists a distribution* $D^{f,T}$ *over* $\mathcal{M} \cup \{\mathsf{same}^\star\}$ *such that:*

$$|\mathsf{Tamper}_m^{f,T} - \mathrm{copy}(D^{f,T}, m)| \leq \varepsilon'$$

*where* copy *is defined by* $\mathrm{copy}(x, y) = \begin{cases} x & \text{if } x \neq \mathsf{same}^\star \\ y & \text{if } x = \mathsf{same}^\star \end{cases}$.

**Many Tampering Extension.** We now extend the above definition to capture multiple tampering attacks. Informally, we say that a secret sharing scheme is non-malleable w.r.t. family $\mathcal{F}$ with tampering degree $\mathsf{K}$ if for any set of $\mathsf{K}$ functions $f_1, \ldots, f_{\mathsf{K}} \in \mathcal{F}$, the output of the following tampering experiment is independent of the shared message $m$: (i) we first share a secret $m$ to obtain the corresponding shares, (ii) we tamper the shares using $f_1, \ldots, f_{\mathsf{K}}$, (iii) we finally, output the $\mathsf{K}$-reconstructed tampered secrets. Note that in the above experiment the message $m$ is secret shared only once but is subjected to $\mathsf{K}$ (possibly different) tamperings.

**Definition 3.12 (Non-Malleable Secret Sharing against Multiple Tampering)** *Let* $(\mathsf{Share}, \mathsf{Rec})$ *be a* $(t, n, \varepsilon_c, \varepsilon_s)$*-secret sharing scheme for message space* $\mathcal{M}$*. Let* $\mathcal{F}$ *be some family of tampering functions. For* $\overrightarrow{f} = (f_1, \ldots, f_{\mathsf{K}}) \in \mathcal{F}^{\mathsf{K}}$*,* $m \in \mathcal{M}$ *and authorized set* $T$ *where* $T$ *contains* $t$ *indices, we define the tampered distribution* $\mathsf{Tamper}_m^{\overrightarrow{f},T}$ *as* $(\mathsf{Rec}(f_1(\mathsf{shares})_T), \ldots, \mathsf{Rec}(f_t(\mathsf{shares})_T) : \mathsf{shares} \leftarrow \mathsf{Share}(m))$ *where the randomness is over the sharing function* $\mathsf{Share}$*. We say that the* $(t, n, \varepsilon_c, \varepsilon_s)$*-secret sharing scheme,* $(\mathsf{Share}, \mathsf{Rec})$ *is* $\varepsilon'$*-non-malleable with tampering degree* $\mathsf{K}$ *w.r.t.* $\mathcal{F}$ *if for each* $\overrightarrow{f} \in \mathcal{F}^{\mathsf{K}}$ *and any authorized set* $T$ *where each elements consists of* $t$ *indices, there exists a distribution* $D^{\overrightarrow{f},T}$ *over* $(\mathcal{M} \cup \{\mathsf{same}^\star\})^{\mathsf{K}}$ *such that:*

$$|\mathsf{Tamper}_m^{\overrightarrow{f},T} - \widetilde{\mathrm{copy}}(D^{\overrightarrow{f},T}, m)| \leq \varepsilon'$$

*where* $\widetilde{\mathrm{copy}}$ *is defined by* $\widetilde{\mathrm{copy}}(\overrightarrow{x}, y) = (z_1, ..., z_n)$ *where* $z_i = \begin{cases} x_i & \text{if } x_i \neq \mathsf{same}^\star \\ y & \text{if } x_i = \mathsf{same}^\star \end{cases}$..

**Remark 3.13** *It is possible to further strengthen the above definition by requiring the output of every tampering function $f_i$ to use a different authorized set $T_i$ for reconstruction. Our construction does not satisfy this stronger definition. However, we note that the impossibility of apriori unbounded number of tamperings holds even with respect to the weakened definition of using the same authorized set for reconstruction in every tampering.*

### 3.2 Non-Malleable Codes

We start with the definition of a coding scheme.

**Definition 3.14 (Coding scheme)** *Let* $\mathsf{Enc} : \{0,1\}^m \to \{0,1\}^n$ *be a randomized algorithm and* $\mathsf{Dec} : \{0,1\}^n \to \{0,1\}^m \cup \{\perp\}$ *be a deterministic function. We say that* $(\mathsf{Enc}, \mathsf{Dec})$ *is a* coding scheme with code length $n$ and message length $m$ *if for all* $s \in \{0,1\}^m$*,* $\Pr[\mathsf{Dec}(\mathsf{Enc}(s)) = s] = 1$*, where the probability is taken over the randomness of* $\mathsf{Enc}$*. The* rate *of the coding scheme is* $\frac{m}{n}$*.*

Dziembowski, Pietrzak and Wichs [DPW10] introduced the notion of non-malleable codes which generalizes the usual notion of error correction. In particular, it guarantees that when a codeword is subject to tampering attack, the reconstructed message is either the original one or something that is independent of the original message.

**Definition 3.15 (Non-Malleable Codes [DPW10])** *Let* $\mathsf{Enc} : \{0,1\}^m \to \{0,1\}^n$ *and* $\mathsf{Dec} : \{0,1\}^n \to \{0,1\}^m \cup \{\perp\}$ *be (possibly randomized) functions, such that* $\mathsf{Dec}\big(\mathsf{Enc}(s)\big) = s$ *with probability 1 for all* $s \in \{0,1\}^m$*. Let* $\mathcal{F}$ *be a family of tampering functions and fix* $\varepsilon > 0$*. We say that* $(\mathsf{Enc}, \mathsf{Dec})$ *is* $\varepsilon-$non-malleable *w.r.t.* $\mathcal{F}$ *if for every* $f \in \mathcal{F}$*, there exists a random variable* $D_f$ *on* $\{0,1\}^m \cup \{\mathsf{same}^\star\}$*, such that for all* $s \in \{0,1\}^m$*,*

$$|\mathsf{Dec}(f(X_s)) - \mathrm{copy}(D_f, s)| \leq \varepsilon$$

*where* $X_s \leftarrow \mathsf{Enc}(s)$ *and* copy *is defined by* $\mathrm{copy}(x,y) = \begin{cases} x & \textit{if } x \neq \mathsf{same}^\star \\ y & \textit{if } x = \mathsf{same}^\star \end{cases}$*. We call* $n$ *the* length *of the code and* $m/n$ *the* rate*.*

Chattopadhyay, Goyal and Li [CGL16] defined a stronger notion of non-malleability against multiple tampering and we now recall this definition.

**Definition 3.16 (Non-Malleable Codes against Multiple Tampering [CGL16])** *A coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *with code length* $n$ *and message length* $m$ *is a non-malleable code with tampering degree* $t$ *w.r.t. a family of tampering functions* $\mathcal{F} \subset (\mathcal{F}_n)^t$ *and error* $\varepsilon$ *if for every* $(f_1, \ldots, f_t) \in \mathcal{F}$*, there exists a random variable* $D_{\overrightarrow{f}}$ *on* $(\{0,1\}^m \cup \{\mathsf{same}^\star\})^t$ *such that for all messages* $s \in \{0,1\}^m$*, it holds that*

$$|(\mathsf{Dec}(f_1(X)), \ldots, \mathsf{Dec}(f_t(X))) - \widetilde{\mathrm{copy}}(D_{\overrightarrow{f}}, s)| \leq \varepsilon$$

*where* $X = \mathsf{Enc}(s)$*. We refer to* $t$ *as the* tampering degree *of the code.*

**Split-state Tampering Functions.** We focus on the *split-state* tampering model where the encoding scheme splits $s$ into $c$ states: $\mathsf{Enc}(s) = (\mathsf{S}_1, \ldots, \mathsf{S}_c) \in \mathcal{S}_1 \times \mathcal{S}_2 \ldots \times \mathcal{S}_c$ and the tampering family is $\mathcal{F}_{\mathit{split}} = \big\{(f_1, \ldots, f_c)\big| f_i : \mathcal{S}_i \to \mathcal{S}_i\big\}$. We will call such a code as $c$-split-state non-malleable code.

**Augmented Non-Malleable Codes.** We recall the definition of augmented, 2-split-state non-malleable codes [AAG+16].

**Definition 3.17 (Augmented Non-Malleable Codes [AAG+16])** *A coding scheme* $(\mathsf{Enc}, \mathsf{Dec})$ *with code length* $2n$ *and message length* $m$ *is an augmented 2-split-state non-malleable code with error* $\varepsilon$ *if for every function* $f, g : \{0,1\}^n \to \{0,1\}^n$*, there exists a random variable* $D_{(f,g)}$ *on* $\{0,1\}^n \times (\{0,1\}^m \cup \{\mathsf{same}^\star\})$ *such that for all messages* $s \in \{0,1\}^m$*, it holds that*

$$|(\mathsf{L}, \mathsf{Dec}(f(\mathsf{L}), g(\mathsf{R}))) - \mathcal{S}(D_{(f,g)}, s)| \le \varepsilon$$

*where* $(\mathsf{L}, \mathsf{R}) = \mathsf{Enc}(s)$*,* $(\mathsf{L}, \widetilde{m}) \leftarrow D_{f,g}$ *and* $\mathcal{S}((\mathsf{L}, \widetilde{m}), s)$ *outputs* $(\mathsf{L}, s)$ *if* $\widetilde{m} = \mathsf{same}^\star$ *and otherwise outputs* $(\mathsf{L}, \widetilde{m})$*.*

**Explicit Constructions.** We now recall the constructions of split-state non-malleable codes.

**Theorem 3.18 ( [Li17])** *For any* $n \in \mathbb{N}$*, there exists an explicit construction of* $2$*-split-state non-malleable code with efficient encoder/decoder, code length* $2n$*, rate* $O(\frac{1}{\log n})$ *and error* $2^{-\Omega(\frac{n}{\log n})}$*.*

**Theorem 3.19 ( [KOS18, GMW17])** *For every* $n \in \mathbb{N}$ *and* $\rho > 0$*, there exists an explicit construction of* $3$*-split-state non-malleable code with efficient encoder/decoder, code length* $(3 + o(1))n$*, rate* $\frac{1}{3+o(1)}$ *and error* $2^{-\Omega(n/\log^{1+\rho}(n))}$*.*

**Theorem 3.20 ( [CGL16])** *There exists a constant* $\gamma > 0$ *such that for every* $n \in \mathbb{N}$ *and* $t \le n^\gamma$*, there exists an explicit construction of 2-split-state non-malleable code with an efficient encoder/decoder, tampering degree* $t$*, code length* $2n$*, rate* $\frac{1}{n^{\Omega(1)}}$ *and error* $2^{-n^{\Omega(1)}}$*.*

**Theorem 3.21 ( [GKP+18])** *There exists a constant* $\gamma > 0$ *such that for every* $n \in \mathbb{N}$ *and* $t \le n^\gamma$*, there exists an explicit construction of an augmented, split-state non-malleable code with an efficient encoder/decoder, tampering degree* $t$*, code length* $2n$*, rate* $\frac{1}{n^{\Omega(1)}}$ *and error* $2^{-n^{\Omega(1)}}$*.*

**Theorem 3.22** *There exists a constant* $\gamma > 0$ *such that for every* $n \in \mathbb{N}$ *and* $t \le n^\gamma$*, there exists an explicit construction of 3-split-state non-malleable code with an efficient encoder/decoder, tampering degree* $t$*, code length* $3n$*, rate* $\Theta(\frac{1}{t})$ *and error* $2^{-n^{\Omega(1)}}$*.*

We give the proof of this theorem in Appendix B.

**Additional Property.** We show in Appendix A that the construction given in [KOS18,GMW17] satisfies the property that given two particular states of the codeword, the message remains statistically hidden.

# 4 $k$-out-of-n Leakage Resilient Secret Sharing Scheme

In this section, we give a new, rate-efficient construction of $k$-out-of-$n$ leakage resilient secret sharing scheme for a constant $k$. Later, in Section 5, we will use this primitive along with a 3-split-state non-malleable code with explicit constant rate (see Theorem 3.19) from the works of Kanukurthi et al. [KOS18] and Gupta et al. [GMW17] to construct a $t$-out-of-$n$ non-malleable secret sharing scheme with the above mentioned rate.

We first recall the definition of a leakage resilient secret sharing scheme from [GK18a].

**Definition 4.1 (Leakage Resilient Secret Sharing [GK18a])** *A $(t, n, \varepsilon_c, \varepsilon_s)$ (for $t \geq 2$) secret sharing scheme* $(\mathsf{Share}, \mathsf{Rec})$ *for message space $\mathcal{M}$ is said to be $\varepsilon$-leakage resilient against a leakage family $\mathcal{F}$ if for all functions $f \in \mathcal{F}$ and for any two messages $m_0, m_1 \in \mathcal{M}$:*

$$|f(\mathsf{Share}(m_0)) - f(\mathsf{Share}(m_1))| \leq \varepsilon$$

**Leakage Function Family.** We are interested in constructing leakage resilient secret sharing schemes against the specific function family $\mathcal{F}_{k,\overline{k},\overrightarrow{\mu}} = \{f_{K,\overline{K},\overrightarrow{\mu}} : K \subseteq [n], |K| = k, \overline{K} \subseteq K, |\overline{K}| \leq \overline{k}\}$ where $f_{K,\overline{K},\overrightarrow{\mu}}$ on input $(\mathsf{share}_1, \ldots, \mathsf{share}_n)$ outputs $\mathsf{share}_i$ for each $i \in \overline{K}$ in the clear and outputs $f_i(\mathsf{share}_i)$ for every $i \in K \setminus \overline{K}$ such that $f_i$ is an arbitrary function outputting $\mu_i$ bits. When we just write $\mu$ (without the vector sign), we mean that every function $f_i$ outputs at most $\mu$ bits.

**Organization.** The rest of this section is organized as follows: we first construct a $k$-out-of-$k$ leakage resilient secret sharing scheme against $\mathcal{F}_{k,k-1,\mu}$ (in other words, $k-1$ shares are output in the clear and $\mu$ bits are leaked from the $k$-th share) in Section 4.1. In Section 4.2, we recall the definition of a combinatorial object called as *perfect hash function family* and give a randomized construction of such a family. Next, in section 4.3, we combine the construction of $k$-out-of-$k$ leakage resilient secret sharing scheme and a perfect hash function family to give a construction of $k$-out-of-$n$ leakage resilient secret sharing scheme (for a constant $k$).

## 4.1 $k$-out-of-$k$ Leakage Resilient Secret Sharing

In this subsection, we will construct a $k$-out-$k$ leakage resilient secret sharing scheme against $\mathcal{F}_{k,k-1,\mu}$ for an arbitrary $k \geq 2$ (and not just for a constant $k$). As a building block, we will use a 2-out-of-2 leakage resilient secret sharing which was constructed in [GK18a]. We first recall the lemma regarding this construction.

**Lemma 4.2 ( [GK18a])** *For any $\varepsilon > 0$ and $\mu, m \in \mathbb{N}$, there exists a construction of $(2, 2, 0, 0)$ secret sharing scheme for sharing $m$-bit secrets that is $\varepsilon$-leakage resilient against $\mathcal{F}_{2,1,\mu}$ such that the size of each share is $O(m + \mu + \log \frac{1}{\varepsilon})$. The running time of the sharing and reconstruction procedures are* $\mathrm{poly}(m, \mu, \log(1/\varepsilon))$.

Let us denote the secret sharing scheme guaranteed by Lemma 4.2 as $(\mathsf{LRShare}_{(2,2)}, \mathsf{LRRec}_{(2,2)})$. We will use this to construct a $k$-out-of-$k$ leakage resilient secret sharing scheme for $k > 2$.

**Lemma 4.3** *For any $\varepsilon > 0$, $k \geq 2$ and $\mu, m \in \mathbb{N}$, there exists a construction of $(k, k, 0, 0)$ secret sharing scheme for sharing $m$-bit secrets that is $\varepsilon$-leakage resilient against $\mathcal{F}_{k,k-1,\mu}$ such that the size of each share is $O(m + \mu + \log \frac{1}{\varepsilon})$. The running time of the sharing and the reconstruction procedures are* $\mathrm{poly}(m, \mu, k, \log(1/\varepsilon))$.

**Proof** We will use a $\varepsilon/2$-leakage resilient $(\mathsf{LRShare}_{(2,2)}, \mathsf{LRRec}_{(2,2)})$ as the main building block. We consider two cases depending on whether $k$ is odd or $k$ is even.

- **Case-1: $k$ is odd.** Let $k = 2k' + 1$. To share a secret $s \in \{0,1\}^m$, we first choose $k' + 1$ strings $s_1, \ldots, s_{k'+1}$ randomly from $\{0,1\}^m$ such that $s_1 \oplus s_2 \oplus \ldots \oplus s_{k'+1} = s$. For each $i \in [k' + 1]$, we share $s_i$ using $\mathsf{LRShare}_{(2,2)}$ to obtain $(\mathsf{share}_{2i-1}, \mathsf{share}_{2i})$. The $k$-shares are given by $(\mathsf{share}_1, \mathsf{share}_2, \ldots, \mathsf{share}_{2k'} \| \mathsf{share}_{2k'+1}, \mathsf{share}_{2k'+2})$. To reconstruct the secret from the shares, we first reconstruct $s_i$ from $(\mathsf{share}_{2i-1}, \mathsf{share}_{2i})$ for each $i \in [k'+1]$ using $\mathsf{LRRec}_{(2,2)}$ and then reconstruct $s$ as $s_1 \oplus s_2 \ldots \oplus s_{k'+1}$. The fact that this is a $(k, k, 0, 0)$ secret sharing scheme follows directly from the fact that $(\mathsf{LRShare}_{(2,n)}, \mathsf{LRRec}_{(2,n)})$ is a $(2, 2, 0, 0)$ secret sharing scheme. We now argue that this sharing scheme is $\varepsilon$-leakage resilient against $\mathcal{F}_{k,k-1,\mu}$.

Let us fix an arbitrary function $f_{K,\overline{K},\mu} \in \mathcal{F}_{k,k-1,\mu}$. Without loss of generality, we assume that $|\overline{K}| = k-1$ as the distinguisher sees strictly more information in this case. By a simple pigeon hole argument, we infer that there exists an $i \in [k'+1]$ such that either $\mathsf{share}_{2i-1}$ or $\mathsf{share}_{2i}$ is not in $\mathsf{shares}_{\overline{K}}$. Let us assume that $\mathsf{share}_{2i-1} \notin \mathsf{shares}_{\overline{K}}$ and the case where $\mathsf{share}_{2i} \notin \mathsf{shares}_{\overline{K}}$ is identical. We now consider a hybrid distribution where $s_i$ is replaced with a random and independent string instead of fixing it as $s \oplus s_1 \oplus \ldots s_{i-1} \oplus s_{i+1} \ldots \oplus s_{k'+1}$. It now follows from the leakage resilience of $\mathsf{LRShare}_{(2,2)}$ that this hybrid is $\varepsilon/2$-close to the real hybrid distribution where a secret $s$ is shared. By the same argument, we can show that this hybrid is $\varepsilon/2$-close to a hybrid where the secret $s'$ is shared. This completes the proof.

- **Case-2:** $k$ **is even.** Let $k = 2k'$. To share a secret $s \in \{0,1\}^m$, we choose $k'$ strings $s_1, \ldots, s_{k'}$ uniformly at random from $\{0,1\}^n$ subject to $s_1 \oplus \ldots \oplus s_{k'} = s$. For each $i \in [k']$, we share $s_i$ using $\mathsf{LRShare}_{(2,2)}$ to obtain $\mathsf{share}_{2i-1}, \mathsf{share}_{2i}$. The $k$-shares are given by $\mathsf{share}_1, \ldots, \mathsf{share}_{2k'}$. To reconstruct the secret from the shares, we first reconstruct $s_i$ from $\mathsf{share}_{2i-1}, \mathsf{share}_{2i}$ for each $i \in [k']$ using $\mathsf{LRRec}_{(2,2)}$ and then reconstruct $s$ as $s_1 \oplus s_2 \ldots \oplus s_{k'}$. The fact that this is a $(k, k, 0, 0)$ secret sharing scheme again follows from the fact that $(\mathsf{LRShare}_{(2,n)}, \mathsf{LRRec}_{(2,n)})$ is a $(2, 2, 0, 0)$ secret sharing scheme. We now apply a similar argument as in Case-1 to show that this is $\varepsilon$-leakage resilient against $\mathcal{F}_{k,k-1,\mu}$.

  Let us fix an arbitrary function $f_{K,\overline{K},\mu} \in \mathcal{F}_{k,k-1,\mu}$. Without loss of generality, we assume that $|\overline{K}| = k-1$ as the distinguisher sees strictly more information in this case. As in Case-1, we infer that there exists an $i \in [k']$ such that either $\mathsf{share}_{2i-1}$ or $\mathsf{share}_{2i}$ is not in $\mathsf{shares}_{\overline{K}}$. Let us assume that $\mathsf{share}_{2i-1} \notin \mathsf{shares}_{\overline{K}}$ and the case where $\mathsf{share}_{2i} \notin \mathsf{shares}_{\overline{K}}$ is identical. We now consider a hybrid distribution where $s_i$ is replaced with a random and independent string instead of fixing it as $s \oplus s_1 \oplus \ldots s_{i-1} \oplus s_{i+1} \ldots \oplus s_{k'}$. It now follows from the leakage resilience of $\mathsf{LRShare}_{(2,2)}$ that this hybrid is $\varepsilon/2$-close to the real hybrid distribution where a secret $s$ is shared. By the same argument, we can show that the same hybrid is $\varepsilon/2$-close to a hybrid where the secret $s'$ is shared. This completes the proof.

■

## 4.2 Perfect Hash Function Family

In this subsection, we recall the definition of the combinatorial objects called as *perfect hash function family* and give an efficient randomized construction for constant $k$.

**Definition 4.4 (Perfect Hash Function Family [FK84])** *For every $n, k \in \mathbb{N}$, a set of hash functions $\{h_i\}_{i \in [\ell]}$ where $h_i : [n] \to [k]$ is said to be $(n, k)$-perfect hash function family if for each subset $S \subseteq [n]$ of size $k$ there exists an $i \in [\ell]$ such that $h_i$ is injective on $S$.*

Before we give the randomized construction, we will state and prove the following useful lemma.

**Lemma 4.5** *For every $\varepsilon > 0$, $n, k \in \mathbb{N}$, the set of functions $\{h_i\}_{i\in[\ell]}$ where each $h_i$ is chosen randomly from the set of all functions mapping $[n] \to [k]$ is a perfectly hash function family with probability $1-\varepsilon$ when $\ell = \frac{\log\binom{n}{k}+\log\frac{1}{\varepsilon}}{\log\frac{1}{1-\frac{k!}{k^k}}}$. Specifically, when $k$ is constant, we can set $\ell = O(\log n + \log\frac{1}{\varepsilon})$.*

**Proof** Let us first fix a subset $S \subseteq [n]$ of size $k$. Let us choose a function $h$ uniformly at random from the set of all functions mapping $[n] \to [k]$.

$$\Pr[h \text{ is not injective over } S] = 1 - \frac{k!}{k^k}$$

Let us now choose $h_1, \ldots, h_\ell$ uniformly at random from the set of all functions mapping $[n] \to [k]$.

$$\Pr[\forall\ i \in [\ell],\ h_i \text{ is not injective over } S] = (1 - \frac{k!}{k^k})^\ell$$

By union bound,

$$\Pr[\exists\ S \text{ s.t.,} \forall\ i \in [\ell],\ h_i \text{ is not injective over } S] = \binom{n}{k}(1 - \frac{k!}{k^k})^\ell$$

We want $\binom{n}{k}(1-\frac{k!}{k^k})^\ell = \varepsilon$. We get the bound for $\ell$ by rearranging this equation. ■

**Randomized Construction for constant $k$.** For any $k, n$ and some error parameter $\varepsilon$, set $\ell$ as in Lemma 4.5. Choose a function $h_i : [n] \to [k]$ uniformly at random for each $i \in [\ell]$. From Lemma 4.5, we infer that $\{h_i\}_{i\in[\ell]}$ is a perfect hash function family except with probability $\varepsilon$. The construction is efficient since the number of random bits needed for choosing each $h_i$ is $n \log k$ which is polynomial in $n$ when $k$ is a constant.

**Explicit Construction.** Building on the work of Schmidt and Siegal [SS90], Alon et al. [AYZ95] gave an explicit construction of $(n, k)$-perfect hash function family of size $2^{O(k)} \log n$. We now recall the lemma from [AYZ95].

**Lemma 4.6 ( [AYZ95, SS90])** *For every $n, k \in \mathbb{N}$, there exists an explicit and efficiently computable construction of $(n, k)$-perfect hash function family $\{h_i\}_{i\in[\ell]}$ where $\ell = 2^{O(k)} \log n$.*

The explicit construction is obtained by brute forcing over a small bias probability space [NN93] and finding such a family is not as efficient as our randomized construction. On the positive side, the explicit construction is error-free unlike our randomized construction.

### 4.3 Construction of $k$-out-$n$ Leakage Resilient Secret Sharing

In this subsection, we will use a $k$-out-of-$k$ leakage resilient secret sharing scheme from Section 4.1 and a perfect hash function family from Section 4.2 to construct a $k$-out-of-$n$ leakage resilient secret sharing scheme against $\mathcal{F}_{t,k-1,\overrightarrow{\mu}}$ for an arbitrary $t \leq n$ (recall the definition of $\mathcal{F}_{k,\overline{k},\overrightarrow{\mu}}$ from Definition 4.1). We give the description in Figure 2.

Let $(\mathsf{LRShare}_{(k,k)}, \mathsf{LRRec}_{(k,k)})$ be a $k$-out-of-$k$ leakage resilient secret sharing scheme.

$\mathsf{LRShare}_{(k,n)}$ : To share a secret $s$:

1. For each $\mathsf{trial} \in [1, \log(1/\varepsilon_c)]$ do:
    (a) Set $\varepsilon = 1/2$ and $\ell = O(\log n)$. Sample a (candidate) $(n,k)$-perfect hash function family $\{h_i\}_{i\in[\ell]}$ as described in Section 4.2
    (b) Check if $\{h_i\}_{i\in[\ell]}$ is a family of $(n,k)$-perfect hash functions. That is, for each set $S \subset [n]$ and $|S| = k$, check if there exists an $i \in [\ell]$ such that $h_i$ is injective on $S$.
    (c) If yes, exit the loop. Otherwise, go to the beginning.
2. If the above loop fails to find a perfect hash function family then abort.
3. For each $i \in [\ell]$, sample $\overline{\mathsf{share}}_{i,1}, \ldots, \overline{\mathsf{share}}_{i,k} \leftarrow \mathsf{LRShare}_{(k,k)}(s)$.
4. For each $j \in [n]$, set $\mathsf{share}_j = (h_1(j), \overline{\mathsf{share}}_{1,h_1(j)}) \circ (h_2(j), \overline{\mathsf{share}}_{2,h_2(j)}) \circ \ldots \circ (h_\ell(j), \overline{\mathsf{share}}_{\ell,h_\ell(j)})$.

$\mathsf{LRRec}_{(k,n)}$ : Given the shares $\mathsf{share}_{j_1}, \mathsf{share}_{j_2}, \ldots, \mathsf{share}_{j_k}$ do:

1. Choose an $i \in [\ell]$, such that $\{h_i(j_1), h_i(j_2), \ldots, h_i(j_k)\} = \{1, \ldots, k\}$.
2. Recover $s$ as $\mathsf{LRRec}_{(k,k)}(\overline{\mathsf{share}}_{i,1}, \ldots, \overline{\mathsf{share}}_{i,k})$.

**Figure 1**: $(k, n, \varepsilon_c, 0)$ Leakage Resilient Secret Sharing Scheme

**Theorem 4.7** *For every $\varepsilon_c, \varepsilon_s > 0$, $n, k, m \in \mathbb{N}$ and $\overrightarrow{\mu} \in \mathbb{N}^n$, the construction given in Figure 2 is a $(k, n, \varepsilon_c, 0)$ secret sharing scheme for sharing $m$-bit secrets that is $\varepsilon_s$-leakage resilient against leakage functions $\mathcal{F}_{t,k-1,\overrightarrow{\mu}}$ for any $t \le n$. The running times of the sharing and reconstruction algorithms are $\mathrm{poly}(n, m, \sum_i \mu_i, \log(1/\varepsilon_c\varepsilon_s))$ when $k$ is a constant. In particular, when $\varepsilon_s = \varepsilon_c = 2^{-m}$, the running times are $\mathrm{poly}(n, m, \sum_i \mu_i)$. The size of each share when $k$ is a constant is $O((m + \max_T \sum_{i\in T, T\subseteq[n], |T|=t} \mu_i + \log(\log n/\varepsilon_s)) \log n)$.*

**Proof** We first argue correctness. That is, we show that the reconstruction always succeeds except with probability $\varepsilon_c$ (over the randomness of the sharing procedure). We then prove perfect privacy and the leakage resilience.

**Correctness.** We first note that if we set $\ell = O(\log n)$ and $\varepsilon = 1/2$, then each trial of the for loop (lines 1.(a)-(c) in Figure 2) fails to find a perfect hash function with probability 1/2. It now follows that the probability that $\log(1/\varepsilon_c)$ independent trials fail to find a perfect hash function family is at most $\varepsilon_c$. Thus, with probability at least $1 - \varepsilon_c$, we find a perfect hash function family at the end of the for loop. It now follows from the definition of perfect hash function family that for every set of $k$-shares $\mathsf{share}_{j_1}, \ldots, \mathsf{share}_{j_k}$, there exists an $i \in [\ell]$ s.t. $h_i$ is injective on $\{j_1, \ldots, j_k\}$. In this case, we infer that $\mathsf{share}_{j_1}, \ldots, \mathsf{share}_{j_k}$ contains $\overline{\mathsf{share}}_{i,1}, \overline{\mathsf{share}}_{i,2}, \ldots, \overline{\mathsf{share}}_{i,k}$. The correctness now follows from the correctness of $k$-out-of-$k$ leakage resilient secret sharing scheme.

**Perfect Privacy.** We first observe that for any set of at most $k-1$ shares $(\mathsf{share}_{j_1}, \ldots, \mathsf{share}_{j_{k-1}})$, we have that for each $i \in [\ell]$, $|\{h_i(j_1), \ldots, h_i(j_{k-1})\}| \leq k-1$. We can now use the perfect privacy of $\mathsf{LRShare}_{(k,k)}$ to argue the perfect privacy of our construction.

**Leakage Resilience.** We instantiate the $k$-out-of-$k$ $\varepsilon'$-leakage resilient secret sharing scheme against leak functions $\mathcal{F}_{k,k-1,\mu'}$ with $\mu' = \max_T \sum_{i \in T, T \subseteq [n], |T|=t} \mu_i$ and $\varepsilon' = \varepsilon_s/\ell$. We now show the construction given in Figure 2 is $\varepsilon$-secure against leakage function family $\mathcal{F}_{t,k-1,\mu}$ for $t \leq n$.

Let us fix a function $f_{K,\overline{K},\mu} \in \mathcal{F}_{t,k-1,\mu}$. We assume without loss of generality that $|\overline{K}| = k-1$ as the distinguisher sees strictly more information in this case. Let $K = \{j_1, \ldots, j_t\}$ and $\overline{K} = \{j_1, \ldots, j_{k-1}\}$.

Let us assume that the $\mathsf{Share}$ function samples a perfect hash function family $\{h_i\}_{i \in [\ell]}$ as otherwise, leakage resilience trivially holds. Let $\mathsf{share}_j = (h_1(j), \overline{\mathsf{share}}_{1,h_1(j)}) \circ (h_2(j), \overline{\mathsf{share}}_{2,h_2(j)}) \circ \ldots \circ (h_\ell(j), \overline{\mathsf{share}}_{\ell,h_\ell(j)})$ for every $j \in K$. We partition the set $[\ell]$ into $S_1$ and $S_2$ defined as follows. $S_1$ consists of the set of indices $i \in [\ell]$ s.t., $|\{h_i(j_1), \ldots, h_i(j_t)\}| \leq k-1$ and $S_2 = [\ell] \setminus S_1$. Intuitively, for indexes in $S_1$, the $i$-th component of $\{\mathsf{share}_j\}_{j \in K}$ perfectly hides the secret since only at most $k-1$ shares are available. We argue that the secret $s$ is hidden in indexes in $S_2$ from the leakage resilience of $k$-out-of-$k$ secret sharing scheme. We now formalize this argument.

We define a sequence of hybrids $\mathsf{Hyb}_i$ where we use the modified sharing procedure $\mathsf{LRShare}'$ described below.

$\mathsf{LRShare}'_{(i,k,n)}$ :

1. For each $i' < i$, sample $\overline{\mathsf{share}}_{i',1}, \ldots, \overline{\mathsf{share}}_{i',k} \leftarrow \mathsf{LRShare}_{(k,k)}(s')$. We will collectively call $\overline{\mathsf{share}}_{i',1}, \ldots, \overline{\mathsf{share}}_{i',k}$ as $\overline{\mathsf{share}}_{i'}$.
2. For all $i \leq i' \leq \ell$, sample $\overline{\mathsf{share}}_{i',1}, \ldots, \overline{\mathsf{share}}_{i',k} \leftarrow \mathsf{LRShare}_{(k,k)}(s)$.
3. For each $j \in [n]$, set $\mathsf{share}_j = (h_1(j), \overline{\mathsf{share}}_{1,h_1(j)}) \circ (h_2(j), \overline{\mathsf{share}}_{2,h_2(j)}) \circ \ldots \circ (h_\ell(j), \overline{\mathsf{share}}_{\ell,h_\ell(j)})$.

The output of $\mathsf{Hyb}_i$ is $f_{K,\overline{K},\mu}(\mathsf{share}_1, \ldots, \mathsf{share}_j)$. Notice that in $\mathsf{Hyb}_1$ the distribution of the shares given as input to $f_{K,\overline{K},\mu}$ is identical to a valid secret sharing of $s$ and in $\mathsf{Hyb}_{\ell+1}$ is distribution of the shares given as input to $f_{K,\overline{K},\mu}$ is identical to a valid secret sharing of $s'$. In order to prove the leakage resilience property, it is sufficient to show that $\mathsf{Hyb}_1 \approx_{\varepsilon_s} \mathsf{Hyb}_{\ell+1}$. We now show the following claim.

**Claim 4.8** *For every $i \in [\ell]$, we have $\mathsf{Hyb}_i \approx_{\varepsilon'} \mathsf{Hyb}_{i+1}$ where $\varepsilon' = \varepsilon_s/\ell$.*

**Proof** We consider two cases whether $i \in S_1$ or if $i \in S_2$.

- **Case-1:** $i \in S_1$**.** In this case, the number of shares of $\overline{\mathsf{share}}_i$ present in $\{\mathsf{share}_j\}_{j \in K}$ is at most $k-1$ and thus it follows from the perfect privacy of $\mathsf{LRShare}_{(k,k)}$ that $\mathsf{Hyb}_i \equiv \mathsf{Hyb}_{i+1}$.
- **Case-2:** $i \in S_2$**.** Assume for the sake of contradiction the statistical distance between $\mathsf{Hyb}_i$ and $\mathsf{Hyb}_{i+1}$ is greater than $\varepsilon'$. We will construct a leak function $g_{U,\overline{U},\mu'} \in \mathcal{F}_{k,k-1,\mu'}$ against $\mathsf{LRShare}_{(k,k)}$. The leak function $g_{U,\overline{U},\mu'}$ is defined as follows:
  - Let us define the set $\overline{U}$ to be $\{h_i(j)\}_{j \in \overline{K}}$. By definition, $|\overline{U}| \leq k-1$ since $|\overline{K}| = k-1$. The leak function $g_{U,\overline{U},\mu'}$ leaks all the shares $\{\overline{\mathsf{share}}_{i,j}\}_{j \in \overline{U}}$ in the clear.

- We define $U = [k] \setminus \overline{U}$. For each index $a \in U$, we do the following. Let $H_a$ be the set of indices $j \in K \setminus \overline{K}$ such that $\overline{\mathsf{share}}_{i,a}$ appears in $\mathsf{share}_j$. Formally, $H_a := \{j \in K \setminus \overline{K} : h_i(j) = a\}$. For every such $j \in H_a$, we leak the output of $f_j$ which is the leak function that takes in $\mathsf{share}_j$ as input and outputs $\mu_j$ bits. Since $|H_a| \leq t$, the amount of leakage is limited to at most $\mu'$ bits where $\mu' = \max_T \sum_{i \in T, T \subseteq [n], |T| = t} \mu_i$.

It follows from the definition of $g_{U,\overline{U},\mu'}$ that (i) $g_{U,\overline{U},\mu'} \in \mathcal{F}_{k,k-1,\mu'}$ and, (ii) any distinguisher between $\mathsf{Hyb}_i$ and $\mathsf{Hyb}_{i+1}$ can be used in conjunction with $g_{U,\overline{U},\mu'}$ to break the security of $\mathsf{LRShare}_{(k,n)}$.

This completes the proof of the claim. ■

Thus, by repeated application of Claim 4.13, we infer that $\mathsf{Hyb}_1 \approx_{\ell\varepsilon'} \mathsf{Hyb}_{\ell+1}$. This completes the proof of the statistical privacy.

**Running time.** Note that sampling a family of $(n, k)$-perfect hash functions and checking if it is indeed a perfect hash function family can be done in time $\text{poly}(n)$ if $k$ is a constant. Since $\mathsf{LRShare}_{(k,k)}, \mathsf{LRRec}_{(k,k)}$ are efficient procedures (i.e., their running times are $\text{poly}(n, m, \sum_i \mu_i, \log(1/\varepsilon_s))$), the running time of $\mathsf{LRShare}_{(k,n)}$ and $\mathsf{LRRec}_{(k,n)}$ is $\text{poly}(n, m, \sum_i \mu_i, \log(1/\varepsilon_c\varepsilon_s))$ when $k$ is a constant.

**Share Size.** We set $\ell = O(\log n)$, $\mu' = \max_T \sum_{i \in T, T \subseteq [n], |T| = t} \mu_i$ and $\varepsilon' = \varepsilon_s/\ell$. From Lemma 4.3, we infer that the size of $\overline{\mathsf{share}}_{i,j}$ for every $i \in [\ell]$ and $j \in [k]$ is $O(m + \mu' + \log(\log n/\varepsilon))$ and thus the size of each share is $O((m + \mu' + \log(\log n/\varepsilon_s)) \log n)$.

■

**Remark 4.9** *In Figure 2, we cannot directly set the size $\ell = O(\log n + \log \frac{1}{\varepsilon_c})$ and perform a single sampling to find a perfect hash function family. This is because when we want $\varepsilon_c = 2^{-m}$, the size of the function family grows with $m$ and this affects the rate significantly. That is why, it is important to set $\varepsilon = 1/2$ and do $\log \frac{1}{\varepsilon_c}$ independent repetitions in the $\mathsf{LRShare}_{(k,n)}$ function to reduce the error to $\varepsilon_c$.*

## 4.4 Conditional Independence

For constructing NMSS, we require a slightly stronger version of LRSS called as LRSS with conditional independence. We now give the definition and give a construction that satisfies this.

**Definition 4.10** *A $(k, n, \delta_c, \delta_s)$ secret sharing scheme $(\mathsf{Share}, \mathsf{Rec})$ for a message space $\mathcal{M}$ is said to be $\varepsilon$-leakage resilient against a leakage family $\mathbb{F}_{t,k-1,\mu}$ with conditional independence if the following properties hold:*

- ***Conditional Independence.*** *Let $K \subseteq [n]$ such that $|K| = k - 1$. Let $S \subseteq [n] \setminus K$ and $T = [n] \setminus (K \cup S)$. For every such $K, S, T$ and a message $m$, there exists a function $\mathsf{aux}_{K,S}$ such that the following two distributions are identical.*
  1. $(\mathsf{share}_1, \ldots, \mathsf{share}_n) \leftarrow \mathsf{Share}(m; r)$ *(for uniformly chosen $r$) and output* $\mathsf{share}_{[n]}$.
  2. $(\mathsf{share}_1, \ldots, \mathsf{share}_n) \leftarrow \mathsf{Share}(m; r)$, $\mathsf{aux} \leftarrow \mathsf{aux}_{K,S}(m, r)$. *Sample* $\mathsf{share}'_S$ *such that it is consistent with* $\mathsf{aux}, \mathsf{share}_K, m$ *and output* $(\mathsf{share}'_S, \mathsf{share}_{[n] \setminus S})$

- ***Leakage-Resilience.*** *For every set $K$ such that $|K| = k-1$ and $S \subseteq [n] \setminus K$ and for every function $f_{\overline{K},K,\mu} \in \mathbb{F}_{t,k-1,\mu}$, for every two messages $m_0, m_1 \in \mathcal{M}$,*

$$|(\mathsf{aux}_{K,S}(m_0,r), f(\mathsf{Share}(m_0;r))) - (\mathsf{aux}_{K,S}(m_1,r), f(\mathsf{Share}(m_1;r)))| \leq \varepsilon$$

**Augmented Perfect Hash functions.** To construct a LRSS with conditional independence property, we strengthen the requirement of a perfect hash function.

**Definition 4.11 (Augmented Perfect Hash Function Family)** *For every $n, k \in \mathbb{N}$, a set of hash functions $\{h_i\}_{i \in [\ell]}$ where $h_i : [n] \to [k]$ is said to be $(n,k)$-augmented perfect hash function family if*

1. *For each subset $S \subseteq [n]$ of size $k-1$ there exists an $i \in [\ell]$ such that $\{h_i(j)\}_{j \in S} = [k-1]$.*
2. *For every $i \in [\ell]$, consider the sequence $(h_i(1), \ldots, h_i(n))$. Then, every $j \in [k-1]$ occurs exactly once in this sequence.*

We now show that a slight modification of the construction given in Figure 2 when instantiated with augmented perfect hash functions satisfies conditional independence.

Let $(\mathsf{LRShare}_{(k,k)}, \mathsf{LRRec}_{(k,k)})$ be a $k$-out-of-$k$ leakage resilient secret sharing scheme. Let $\{h_i\}_{i \in [\ell]}$ be a $(n, k+1)$-augmented perfect hash function family.

$\mathsf{LRShare}_{(k,n)}$ : To share a secret $s$:

1. For each $i \in [\ell]$, sample $\overline{\mathsf{share}}_{i,1}, \ldots, \overline{\mathsf{share}}_{i,k} \leftarrow \mathsf{LRShare}_{(k,k)}(s)$. Set $\overline{\mathsf{share}}_{i,k+1} = \perp$.
2. For each $j \in [n]$, set $\mathsf{share}_j = (h_1(j), \overline{\mathsf{share}}_{1,h_1(j)}) \circ (h_2(j), \overline{\mathsf{share}}_{2,h_2(j)}) \circ \ldots \circ (h_\ell(j), \overline{\mathsf{share}}_{\ell,h_\ell(j)})$.

$\mathsf{LRRec}_{(k,n)}$ : Given the shares $\mathsf{share}_{j_1}, \mathsf{share}_{j_2}, \ldots, \mathsf{share}_{j_k}$ do:

1. Choose an $i \in [\ell]$, such that $\{h_i(j_1), h_i(j_2), \ldots, h_i(j_k)\} = \{1, \ldots, k\}$.
2. Recover $s$ as $\mathsf{LRRec}_{(k,k)}(\overline{\mathsf{share}}_{i,1}, \ldots, \overline{\mathsf{share}}_{i,k})$.

**Figure 2**: $(k, n, \varepsilon_c, 0)$ Leakage Resilient Secret Sharing Scheme

**Theorem 4.12** *For every $\varepsilon_s > 0$, $n, k, m \in \mathbb{N}$ and $\overrightarrow{\mu} \in \mathbb{N}^n$, the construction given in Figure 2 is a $(k, n, 0, 0)$ secret sharing scheme for sharing $m$-bit secrets that is $\varepsilon_s$-leakage resilient against leakage functions $\mathcal{F}_{t,k-1,\overrightarrow{\mu}}$ for any $t \leq n$ with conditional independence. There exists an instantiation of augmented perfect hash function such that the size of the share is $O(n^{k-1}(m + \mu + \log n + \log(1/\varepsilon_s)))$*

**Proof** Perfect correctness follows directly from the first property of augmented perfect hash function and the perfect correctness of $\mathsf{LRShare}_{(k,k)}$. Perfect privacy is argued exactly as in the previous case. We now argue conditional independence and leakage-resilience.

**Conditional Independence.** Let us fix sets $K \subseteq [n]$ such that $|K| = k-1$, $S \subseteq [n] \setminus K$ and $T = [n] \setminus (K \cup S)$. We start by giving the description of $\mathsf{aux}_{K,S}$. On input the sharing randomness and the message, $\mathsf{aux}_{K,S}$ does the following:

1. For every $i \in [\ell]$ such that $\{h_i(S)\} \neq \{k+1\}$, $\mathsf{aux}_{K,S}$ outputs $\{\overline{\mathsf{share}}_{i,j}\}_{j \in \{h_i(T)\}}$.

To argue conditional independence, we first fix $\mathsf{share}_K$ and $\mathsf{share}_T$ and prove that for each $i \in [\ell]$, $\{h_i(j), \overline{\mathsf{share}}_{i,h_i(j)}\}_{j \in S}$ is identically distributed to $\{h_i(j), \overline{\mathsf{share}}'_{i,h_i(j)}\}_{j \in S}$. This is sufficient since the randomness for generating the $i$-th component is independent of all other components. To see this, let us consider two cases:

- **Case-1:** $\{h_i(S)\} = \{k+1\}$. In this case, both $\overline{\mathsf{share}}_{i,h_i(j)}$ and $\overline{\mathsf{share}}'_{i,h_i(j)}$ will be equal to $\perp$ for each $j \in S$.

- **Case-2:** $\{h_i(S)\} \neq \{k+1\}$. Let $S' \subseteq S$ such that for each $j \in S'$, $h_i(j) \neq k+1$. Note that $\{\overline{\mathsf{share}}_{i,h_i(j)}\}_{j \in h_i(S')}$ is distributed identically to the corresponding shares of a $k$-out-of-$k$ LRSS of $m$ conditioned on fixing $\mathsf{share}_K, \mathsf{share}_T$. Now since the output of $\mathsf{aux}_{K,S}$ contains $\{\overline{\mathsf{share}}_{i,h_i(j)}\}_{j \in h_i(T)}$, sampling $\{\overline{\mathsf{share}}'_{i,h_i(j)}\}_{j \in h_i(S')}$ consistent with the output of $\mathsf{aux}_{K,S}$ and $\mathsf{share}_K$ is also distributed identically to the corresponding shares of a $k$-out-of-$k$ LRSS of $m$ conditioned on fixing $\mathsf{share}_K, \mathsf{share}_T$

**Leakage-Resilience.** Let us fix two message $s, s'$ and fix a function $f_{\overline{K},K,\mu} \in \mathcal{F}_{t,k-1,\mu}$ and a set $S \subseteq [n] \setminus K$. Let $K = \{j_1, \ldots, j_{k-1}\}$ and $\overline{K} = \{j_1, \ldots, j_t\}$.

Let $\mathsf{share}_j = (h_1(j), \overline{\mathsf{share}}_{1,h_1(j)}) \circ (h_2(j), \overline{\mathsf{share}}_{2,h_2(j)}) \circ \ldots \circ (h_\ell(j), \overline{\mathsf{share}}_{\ell,h_\ell(j)})$ for every $j \in \overline{K}$. We partition the set $[\ell]$ into $S_1$ and $S_2$ defined as follows. $S_1$ consists of the set of indices $i \in [\ell]$ s.t., $[k] \not\subseteq \{h_i(j_1), \ldots, h_i(j_t)\}$ and $S_2 = [\ell] \setminus S_1$. Intuitively, for indexes in $S_1$, the $i$-th component of $\{\mathsf{share}_j\}_{j \in \overline{K}}$ perfectly hides the secret since only at most $k-1$ shares are available. We argue that the secret $s$ is hidden in indexes in $S_2$ from the leakage resilience of $k$-out-of-$k$ secret sharing scheme. We now formalize this argument.

We define a sequence of hybrids $\mathsf{Hyb}_i$ where we use the modified sharing procedure $\mathsf{LRShare}'$ described below.

$\mathsf{LRShare}'_{(i,k,n,K,S)}$ :

1. For each $i' < i$, sample $\overline{\mathsf{share}}_{i',1}, \ldots, \overline{\mathsf{share}}_{i',k} \leftarrow \mathsf{LRShare}_{(k,k)}(s')$. We set $\overline{\mathsf{share}}_{i',k+1} = \perp$. We will collectively call $\overline{\mathsf{share}}_{i',1}, \ldots, \overline{\mathsf{share}}_{i',k+1}$ as $\overline{\mathsf{share}}_{i'}$.

2. For all $i \leq i' \leq \ell$, sample $\overline{\mathsf{share}}_{i',1}, \ldots, \overline{\mathsf{share}}_{i',k} \leftarrow \mathsf{LRShare}_{(k,k)}(s)$ and set $\overline{\mathsf{share}}_{i',k+1} = \perp$.

3. For each $j \in [n]$, set $\mathsf{share}_j = (h_1(j), \overline{\mathsf{share}}_{1,h_1(j)}) \circ (h_2(j), \overline{\mathsf{share}}_{2,h_2(j)}) \circ \ldots \circ (h_\ell(j), \overline{\mathsf{share}}_{\ell,h_\ell(j)})$.

4. For every $i \in [\ell]$ such that $\{h_i(S)\} \neq \{k+1\}$, $\mathsf{aux}_{K,S}$ outputs $\{\overline{\mathsf{share}}_{i,j}\}_{j \in \{h_i(T)\}}$.

The output of $\mathsf{Hyb}_i$ is $(\mathsf{aux}_{K,S}, f_{\overline{K},K,\mu}(\mathsf{share}_1, \ldots, \mathsf{share}_j))$. Notice that the output of $\mathsf{Hyb}_1$ is identical to $(\mathsf{aux}_{K,S}(s,r), f_{K,\overline{K},\mu}(\mathsf{share}(s;r))$ and the output of $\mathsf{Hyb}_{\ell+1}$ is identical $(\mathsf{aux}_{K,S}(s',r), f_{K,\overline{K},\mu}(\mathsf{share}(s';r))$. In order to prove the leakage resilience property, it is sufficient to show that $\mathsf{Hyb}_1 \approx_{\varepsilon_s} \mathsf{Hyb}_{\ell+1}$. We now show the following claim.

**Claim 4.13** *For every $i \in [\ell]$, we have $\mathsf{Hyb}_i \approx_{\varepsilon'} \mathsf{Hyb}_{i+1}$ where $\varepsilon' = \varepsilon_s/\ell$.*

**Proof** We consider two cases whether $i \in S_1$ or if $i \in S_2$.

- **Case-1:** $i \in S_1$**.** In this case, the number of shares of $\overline{\mathsf{share}}_i$ present in $\{\mathsf{share}_j\}_{j \in K}$ is at most $k-1$ and thus it follows from the perfect privacy of $\mathsf{LRShare}_{(k,k)}$ that $\mathsf{Hyb}_i \equiv \mathsf{Hyb}_{i+1}$.
- **Case-2:** $i \in S_2$**.** We will consider two sub-cases.
  - **Case-2.(a):** $\{h_i(S)\} = \{k+1\}$. In this case, $\mathsf{aux}_{K,S}$ does not contain any information about the $\{\overline{\mathsf{share}}_{i,j}\}$. We give a reduction to the leakage-resilience of $\mathsf{LRShare}_{k,k}$. Assume for the sake of contradiction the statistical distance between $\mathsf{Hyb}_i$ and $\mathsf{Hyb}_{i+1}$ is greater than $\varepsilon'$. We will construct a leak function $g_{\overline{U},U,\mu} \in \mathcal{F}_{k,k-1,\mu'}$ against $\mathsf{LRShare}_{(k,k)}$. The leak function $g_{\overline{U},U,\mu}$ is defined as follows:
    - $*$ Let us define the set $U$ to be $\{h_i(j)\}_{j \in K}$. By definition, $|U| \leq k-1$ since $|K| = k-1$. The leak function $g_{\overline{U},U,\mu'}$ leaks all the shares $\{\overline{\mathsf{share}}_{i,j}\}_{j \in U}$ in the clear.
    - $*$ We define $U = [k]$. For each index $a \in U$, we leak the output of $f_j$ which is the leak function that takes in $\mathsf{share}_j$ as input and outputs $\mu$ bits. From the second property of augmented perfect hash functions, each $\overline{\mathsf{share}}_{i,j}$ where $j \in [k]$ occurs in exactly once in the sequence $(h_i(1), \ldots, h_i(n))$. Thus, the leakage function $g_{\overline{U},U,\mu} \in \mathcal{F}_{k,k-1,\mu}$.

    It follows from the definition of $g_{U,\overline{U},\mu}$ that (i) $g_{U,\overline{U},\mu} \in \mathcal{F}_{k,k-1,\mu}$ and, (ii) any distinguisher between $\mathsf{Hyb}_i$ and $\mathsf{Hyb}_{i+1}$ can be used in conjunction with $g_{U,\overline{U},\mu}$ to break the security of $\mathsf{LRShare}_{(k,n)}$.
  - **Case-2.(a):** $\{h_i(S)\} \neq \{k+1\}$. In this case, $\mathsf{aux}_{K,S}$ contains $\{\overline{\mathsf{share}}_{i,j}\}_{j \in h_i(T)}$. However, from the second property of augmented perfect hash function family, $|\{h_i(K)\} \cup \{h_i(T)\}| \leq k-1$. Thus, we can use the exact same reduction as above by defining $U = \{h_i(K)\} \cup \{h_i(T)\}$.

This completes the proof of the claim. ■

**Instantiation.** We now give an instantiation of an $(n, k+1)$-augmented perfect hash function. Fix any $n, k \in \mathbb{N}$. Fix $\ell = \binom{n}{k}$ and each hash function is indexed by a $k$-sized set $\{i_1, \ldots, i_k\} \subseteq [n]$. The hash function $h_{(i_1,\ldots,i_k)}(i) = j$ if $i = i_j$ and is equal to $k+1$ otherwise. This function satisfies both the properties of Definition 4.11. Note that for every $i \in [\ell]$ such that $h_i(j) = k+1$, it is not necessary to give out $h_i(j) \circ \overline{\mathsf{share}}_{i,h_i(j)}$ in $\mathsf{share}_j$ since $\overline{\mathsf{share}}_{i,k+1} = \perp$. Thus, the share size is $O(n^{k-1}(m + \mu + \log(n) + \log(1/\varepsilon_s)))$. ■

# 5 Non-Malleable Secret Sharing for Threshold Access Structures

In this section, we give a construction of $t$-out-of-$n$ (for any $t \geq 4$) Non-Malleable Secret Sharing scheme with rate $\Theta(\frac{1}{t \log^2 n})$ against tampering function family $\mathcal{F}_{\mathsf{ind}}$ that tampers each share independently. We first give the formal description of the tampering function family.

**Individual Tampering Family** $\mathcal{F}_{\mathsf{ind}}$**.** Let $\mathsf{Share}$ be the sharing function of the secret sharing scheme that outputs $n$-shares in $\mathcal{S}_1 \times \mathcal{S}_2 \ldots \times \mathcal{S}_n$. The function family $\mathcal{F}_{\mathsf{ind}}$ is composed of functions $(f_1, \ldots, f_n)$ where each $f_i : \mathcal{S}_i \rightarrow \mathcal{S}_i$.

## 5.1 Construction

**Building Blocks.** The construction uses the following building blocks. We instantiate them with concrete schemes later:

- A 3-split-state non-malleable code $(\mathsf{Enc}, \mathsf{Dec})$ where $\mathsf{Enc} : \mathcal{M} \to \mathcal{L} \times \mathcal{C} \times \mathcal{R}$ and the simulation error of the scheme is $\varepsilon_1$. Furthermore, we assume that for any two messages $m, m' \in \mathcal{M}$, $(\mathsf{C}, \mathsf{R}) \approx_{\varepsilon_2} (\mathsf{C}', \mathsf{R}')$ where $(\mathsf{L}, \mathsf{C}, \mathsf{R}) \leftarrow \mathsf{Enc}(m)$ and $(\mathsf{L}', \mathsf{C}', \mathsf{R}') \leftarrow \mathsf{Enc}(m')$.

- A $(t, n, 0, 0)$ secret sharing scheme $(\mathsf{SecShare}_{(t,n)}, \mathsf{SecRec}_{(t,n)})$ with perfect privacy for message space $\mathcal{L}$. We will assume that the size of each share is $m_1$.

- A $(3, n, \varepsilon_3', 0)$ secret sharing scheme $(\mathsf{LRShare}_{(3,n)}, \mathsf{LRRec}_{(3,n)})$ that is $\varepsilon_3$-leakage resilient against leakage functions $\mathcal{F}_{t,2,m_1}$[8] for message space $\mathcal{C}$ with conditional independence. We assume that the size of each share is $m_2$.

- A $(2, n, \varepsilon_4', 0)$ secret sharing scheme $(\mathsf{LRShare}_{(2,n)}, \mathsf{LRRec}_{(2,n)})$ for message space $\mathcal{R}$ that is $\varepsilon_4$-leakage resilient against leakage functions $\mathcal{F}_{t,1,\mu}$ where $\mu = m_1 + m_2$ and has conditional independence. We assume that the size of each share is $m_3$.

**Construction.** We give the formal description of the construction in Figure 3 and give an informal overview below. To share a secret $s$, we first encode $s$ to $(\mathsf{L}, \mathsf{C}, \mathsf{R})$ using the 3-split-state non-malleable code. We first encode $\mathsf{L}$ to $(\mathsf{SL}_1, \ldots, \mathsf{SL}_n)$ using the $t$-out-of-$n$ threshold secret sharing scheme. We then encode $\mathsf{C}$ into $(\mathsf{SC}_1, \ldots, \mathsf{SC}_n)$ using the 3-out-of-$n$ leakage resilience secret sharing scheme $\mathsf{LRShare}_{(3,n)}$. We finally encode $\mathsf{R}$ into $(\mathsf{SR}_1, \ldots, \mathsf{SR}_n)$ using the 2-out-of-$n$ leakage resilient secret sharing scheme $\mathsf{LRShare}_{(2,n)}$. We set the $i$-th share $\mathsf{share}_i$ to be the concatenation of $\mathsf{SL}_i, \mathsf{SC}_i$ and $\mathsf{SR}_i$. In order to reconstruct, we using the corresponding reconstruction procedures $\mathsf{SecRec}$, $\mathsf{LRRec}_{(3,n)}$ and $\mathsf{LRRec}_{(2,n)}$ to compute $\mathsf{L}$, $\mathsf{C}$ and $\mathsf{R}$ respectively. We finally use the decoding procedure of 3-split-state non-malleable code to reconstruct the secret $s$ from $\mathsf{L}, \mathsf{C}$ and $\mathsf{R}$.

**Theorem 5.1** *For any arbitrary $n \in \mathbb{N}$ and threshold $t \geq 4$, the construction given in Figure 3 is a $(t, n, \varepsilon_3' + \varepsilon_4', \varepsilon_2)$ secret sharing scheme. Furthermore, it is $(\varepsilon_1 + \varepsilon_3 + \varepsilon_4)$-non-malleable against $\mathcal{F}_{\mathsf{ind}}$.*

## 5.2 Proof of Theorem 5.1

We now argue correctness, statistical privacy and non-malleability to complete the proof of Theorem 5.1.

**Correctness.** We notice that except with probability $\varepsilon_3' + \varepsilon_4'$, $\mathsf{SecRec}_{(t,n)}$, $\mathsf{LRRec}_{(3,n)}$ and $\mathsf{LRRec}_{(2,n)}$ will be able to reconstruct $\mathsf{L}, \mathsf{C}$, and $\mathsf{R}$ respectively. The correctness now follows directly from the correctness of the decoder $\mathsf{Dec}$ of 3-split-state non-malleable codes.

[8] Recall that this denotes that the function can choose to leak at most $m_1$ bits from each share in a set of size $t-2$ apart from the two that are completely leaked.

$\mathsf{Share}(m)$ : To share a secret $s \in \mathcal{M}$ do:

1. Encode the secret $s$ as $(\mathsf{L}, \mathsf{C}, \mathsf{R}) \leftarrow \mathsf{Enc}(s)$.
2. Compute the shares
$$(\mathsf{SL}_1, \ldots, \mathsf{SL}_n) \leftarrow \mathsf{SecShare}_{(t,n)}(\mathsf{L})$$
$$(\mathsf{SC}_1, \ldots, \mathsf{SC}_n) \leftarrow \mathsf{LRShare}_{(3,n)}(\mathsf{C})$$
$$(\mathsf{SR}_1, \ldots, \mathsf{SR}_n) \leftarrow \mathsf{LRShare}_{(2,n)}(\mathsf{R})$$
3. For each $i \in [n]$, set $\mathsf{share}_i$ as $(\mathsf{SL}_i, \mathsf{SC}_i, \mathsf{SR}_i)$ and output $(\mathsf{share}_1, \ldots, \mathsf{share}_n)$ as the shares.

$\mathsf{Rec}(\mathsf{Share}(m)_T)$ : To reconstruct the secret from the shares in an authorized set $T$ of size $t$ do:

1. Let the shares corresponding to the set $T$ be $(\mathsf{share}_{i_1}, \ldots, \mathsf{share}_{i_t})$.
2. For each $j \in \{i_1, \ldots, i_t\}$, parse $\mathsf{share}_j$ as $(\mathsf{SL}_j, \mathsf{SC}_j, \mathsf{SR}_j)$.
3. Reconstruct
$$\mathsf{L} := \mathsf{SecRec}_{(t,n)}(\mathsf{SL}_{i_1}, \ldots, \mathsf{SL}_{i_t})$$
$$\mathsf{C} := \mathsf{LRRec}_{(3,n)}(\mathsf{SC}_{i_1}, \mathsf{SC}_{i_2}, \mathsf{SC}_{i_3})$$
$$\mathsf{R} := \mathsf{LRRec}_{(2,n)}(\mathsf{SR}_{i_1}, \mathsf{SR}_{i_2})$$
4. Output the secret $s$ as $\mathsf{Dec}(\mathsf{L}, \mathsf{C}, \mathsf{R})$.

**Figure 3**: Construction of $t$-out-of-$n$ Non-Malleable Secret Sharing Scheme

**Statistical Privacy.** To argue statistical privacy, we consider a sequence of hybrids.

- $\mathsf{Hyb}_1$ : In this hybrid, the secret $s$ is shared using $\mathsf{Share}(s)$.
- $\mathsf{Hyb}_2$ : This hybrid is same as $\mathsf{Hyb}_1$ except that we do the following. Let $(\mathsf{L}, \mathsf{C}, \mathsf{R}) \leftarrow \mathsf{Enc}(s)$. We reset $\mathsf{L} = \perp$ and encode this fake $\mathsf{L}$ using $\mathsf{SecShare}_{(t,n)}$.
- $\mathsf{Hyb}_3$ : This hybrid is same as $\mathsf{Hyb}_2$ except that we sample $(\mathsf{L}', \mathsf{C}', \mathsf{R}') \leftarrow \mathsf{Enc}(s')$ and encode $\mathsf{C}', \mathsf{R}'$ instead of $\mathsf{C}, \mathsf{R}$ using $\mathsf{LRShare}_{(3,n)}$ and $\mathsf{LRShare}_{(2,n)}$ respectively.
- $\mathsf{Hyb}_4$ : This hybrid is distributed identically to $\mathsf{Share}(s')$.

We first claim that $\mathsf{Hyb}_1$ is distributed identically to $\mathsf{Hyb}_2$ and $\mathsf{Hyb}_3$ is distributed identically to $\mathsf{Hyb}_4$. This actually follows directly from the security of $\mathsf{SecShare}_{(t,n)}$ since at most $t-1$ shares perfectly hide $\mathsf{L}$.

We now argue that $\mathsf{Hyb}_2 \approx_{\varepsilon_2} \mathsf{Hyb}_3$. Note that in $\mathsf{Hyb}_2$, $(\mathsf{C}, \mathsf{R})$ is generated as part of an encoding of $s$ and in $\mathsf{Hyb}_3$ it is generated as part of an encoding of $s'$. From the property of our 3-state non-malleable code, we know that $(\mathsf{C}, \mathsf{R})$ statistically hide the message, and hence we infer that $\mathsf{Hyb}_2 \approx_{\varepsilon_2} \mathsf{Hyb}_3$.

**Non-Malleability.** We show the non-malleability of our scheme by transforming a tampering attack on the shares of our scheme to a tampering attack on the 3-state non-malleable code. In particular, we will use the tampering functions $(f_1, \ldots, f_n)$ that attack the secret sharing scheme to design a split state tampering function $(f, g, h)$ against the underlying non-malleable code. Note that the split-state functions $f, g$ and $h$ need not be efficiently computable. We then use the security of the underlying non-malleable code to come up with the simulator for our scheme.

Let $(f_1, \ldots, f_n) \in \mathcal{F}_{\mathsf{ind}}$ be a set of tampering functions and $T = \{i_1, \ldots, i_t\}$ be an authorized set. The split state functions $(f, g, h)$ that attack the underlying code are constructed as follows.

- **Shared Randomness.** Let $s_{\$}$ be an arbitrary secret and let $(\mathsf{L}_{\$}, \mathsf{C}_{\$}, \mathsf{R}_{\$}) \leftarrow \mathsf{Enc}(s_{\$})$. Run the sharing function $\mathsf{SecShare}_{(t,n)}, \mathsf{LRShare}_{(3,n)}$ and $\mathsf{LRShare}_{(2,n)}$ on $(\mathsf{L}_{\$}, \mathsf{C}_{\$}, \mathsf{R}_{\$})$ using randomness $r_L, r_C, r_R$ respectively to obtain $(\mathsf{SL}^{\$}_1, \ldots, \mathsf{SL}^{\$}_n)$, $(\mathsf{SC}^{\$}_1, \ldots, \mathsf{SC}^{\$}_n)$ and $(\mathsf{SR}^{\$}_1, \ldots, \mathsf{SR}^{\$}_n)$. For each $i \in [n]$, set $\mathsf{share}^{\$}_i = (\mathsf{SL}^{\$}_i, \mathsf{SC}^{\$}_i, \mathsf{SR}^{\$}_i)$. For $i \in \{i_1, i_2, i_3\}$, run the tampering function $f_i$ on input $(\mathsf{SL}^{\$}_i, \mathsf{SC}^{\$}_i, \mathsf{SR}^{\$}_i)$ to obtain the tampered values $(\widetilde{\mathsf{SL}}^{\$}_i, \widetilde{\mathsf{SC}}^{\$}_i, \widetilde{\mathsf{SR}}^{\$}_i)$. Let $\mathsf{aux}^{(3,n)}_{\{i_1,i_2\},\{i_3\}}$ and $\mathsf{aux}^{(2,n)}_{\{i_3\},\{i_1,i_2\}}$ be the functions guaranteed by the conditional independence of $\mathsf{LRShare}_{(3,n)}$ and $\mathsf{LRShare}_{(2,n)}$. Let $\mathsf{aux}^{(3,n)} \leftarrow \mathsf{aux}^{(3,n)}_{\{i_1,i_2\},\{i_3\}}(r_C)$ and $\mathsf{aux}^{(2,n)} \leftarrow \mathsf{aux}^{(2,n)}_{\{i_3\},\{i_1,i_2\}}(r_R)$. The shared randomness between $f, g$ and $h$ comprises of the following:

$$(\mathsf{SL}^{\$}_{i_1}, \mathsf{SL}^{\$}_{i_2}, \mathsf{SL}^{\$}_{i_3})$$

$$(\widetilde{\mathsf{SL}}^{\$}_{i_1}, \widetilde{\mathsf{SL}}^{\$}_{i_2}, \widetilde{\mathsf{SL}}^{\$}_{i_3})$$

$$(\mathsf{SC}^{\$}_{i_1}, \mathsf{SC}^{\$}_{i_2}, \mathsf{SC}^{\$}_{i_4}, \ldots, \mathsf{SC}^{\$}_{i_t})$$

$$(\widetilde{\mathsf{SC}}^{\$}_{i_1}, \widetilde{\mathsf{SC}}^{\$}_{i_2})$$

$$(\mathsf{SR}^{\$}_{i_3}, \ldots, \mathsf{SR}^{\$}_{i_t})$$

$$(\mathsf{aux}^{(3,n)}, \mathsf{aux}^{(2,n)})$$

- **Function $f$.** The tampering function $f$ on input $\mathsf{L}$ does the following:

  1. It chooses $\mathsf{SL}_{i_4}, \ldots, \mathsf{SL}_{i_t}$ such that $(\mathsf{SL}^{\$}_{i_1}, \mathsf{SL}^{\$}_{i_2}, \mathsf{SL}^{\$}_{i_3}, \mathsf{SL}_{i_4}, \ldots, \mathsf{SL}_{i_t})$ is a valid $t$-out-of-$n$ secret sharing of $\mathsf{L}$.
  2. For every $i \in \{i_4, \ldots, i_t\}$, it runs the tampering function $f_i$ on input $(\mathsf{SL}_i, \mathsf{SC}^{\$}_i, \mathsf{SR}^{\$}_i)$ to obtain $(\widetilde{\mathsf{SL}}_i, \widetilde{\mathsf{SC}}^{\$}_i, \widetilde{\mathsf{SR}}^{\$}_i)$.
  3. It runs $\mathsf{SecRec}_{(t,n)}$ on inputs $(\widetilde{\mathsf{SL}}^{\$}_{i_1}, \widetilde{\mathsf{SL}}^{\$}_{i_2}, \widetilde{\mathsf{SL}}^{\$}_{i_3}, \widetilde{\mathsf{SL}}_{i_4}, \ldots, \widetilde{\mathsf{SL}}_{i_t})$ to obtain $\widetilde{\mathsf{L}}$ and outputs it.

- **Function $g$.** The tampering function $g$ on input $\mathsf{C}$ does the following:

  1. Sample $\mathsf{SC}_{i_3}$ such that the following two conditions are satisfied:
     (a) $\mathsf{SC}_{i_3}$ is consistent with $(\mathsf{SC}^{\$}_{i_1}, \mathsf{SC}^{\$}_{i_2}, \mathsf{aux}^{(3,n)}, \mathsf{C})$.
     (b) $f_{i_3}(\mathsf{SL}^{\$}_{i_3}, \mathsf{SC}_{i_3}, \mathsf{SR}^{\$}_{i_3}) = (\widetilde{\mathsf{SL}}^{\$}_{i_3}, \cdot, \cdot)$. That is, the first component of the output of $f_{i_3}$ on input $\mathsf{SL}^{\$}_{i_3}, \mathsf{SC}_{i_3}, \mathsf{SR}^{\$}_{i_3}$ is equal to $\widetilde{\mathsf{SL}}^{\$}_{i_3}$ (which is part of the shared randomness).

2. In case such a sampling is not possible, it outputs the special symbol $\mathsf{abort}_1$ (it can be thought as some specific symbol in $\mathcal{C}$).
3. Else, it runs $f_{i_3}$ in input $\mathsf{SL}^{\$}_{i_3}, \mathsf{SC}_{i_3}, \mathsf{SR}^{\$}_{i_3}$ to obtain $(\widetilde{\mathsf{SL}}^{\$}_{i_3}, \widetilde{\mathsf{SC}}_{i_3}, \widetilde{\mathsf{SR}'}^{\$}_{i_3})$.
4. Reconstructs $\widetilde{C}$ by running $\mathsf{LRRec}_{(3,n)}(\widetilde{\mathsf{SC}}^{\$}_{i_1}, \widetilde{\mathsf{SC}}^{\$}_{i_2}, \widetilde{\mathsf{SC}}_{i_3})$ and outputs it.

- **Function $h$.** The tampering function $h$ on input $\mathsf{R}$ does the following:
  1. Samples $\mathsf{SR}_{i_1}, \mathsf{SR}_{i_2}$ such that the following three conditions are satisfied:
     (a) $\mathsf{SR}_{i_1}, \mathsf{SR}_{i_2}$ is consistent with $(\mathsf{SR}^{\$}_{i_3}, \mathsf{aux}^{(2,n)}, \mathsf{R})$.
     (b) $f_{i_1}(\mathsf{SL}^{\$}_{i_1}, \mathsf{SC}^{\$}_{i_1}, \mathsf{SR}_{i_1}) = (\widetilde{\mathsf{SL}}^{\$}_{i_1}, \widetilde{\mathsf{SC}}^{\$}_{i_1}, \cdot)$. In other words, the first two components of the output of $f_{i_1}$ on input $\mathsf{SL}^{\$}_{i_1}, \mathsf{SC}^{\$}_{i_1}, \mathsf{SR}_{i_1}$ is same as $\widetilde{\mathsf{SL}}^{\$}_{i_1}, \widetilde{\mathsf{SC}}^{\$}_{i_1}$ (which are part of shared randomness).
     (c) $f_{i_2}(\mathsf{SL}^{\$}_{i_2}, \mathsf{SC}^{\$}_{i_2}, \mathsf{SR}_{i_2}) = (\widetilde{\mathsf{SL}}^{\$}_{i_2}, \widetilde{\mathsf{SC}}^{\$}_{i_2}, \cdot)$.
  2. If such a sampling is not possible, it outputs the special symbol $\mathsf{abort}_2$ (again, it is some specific symbol in $\mathcal{R}$).
  3. Otherwise, for $i \in \{i_1, i_2\}$, it runs $f_i(\mathsf{SL}^{\$}_i, \mathsf{SC}^{\$}_i, \mathsf{SR}_i)$ to obtain $(\widetilde{\mathsf{SL}}^{\$}_i, \widetilde{\mathsf{SC}}^{\$}_i, \widetilde{\mathsf{SR}}_i)$.
  4. Reconstructs $\widetilde{\mathsf{R}}$ by running $\mathsf{LRRec}_{(2,n)}(\widetilde{\mathsf{SR}}_{i_1}, \widetilde{\mathsf{SR}}_{i_2})$ and outputs it.

The functions $f, g, h$ described above constitute a split state tampering function family. In order to reduce the tampering attack on the shares of our secret sharing scheme to the shares of the underlying code, we need to show that the distribution of the shares given as inputs to $f_{\mathsf{ind}} = (f_{i_1}, \ldots, f_{i_t})$ in the tampering experiment $\mathsf{Tamper}_s^{f_{\mathsf{ind}},T}$ is statistically close to the distribution of the shares that $f, g, h$ give as input to these functions. We show this via a hybrid argument.

$\mathsf{Hyb}_1$ : This is same as the experiment where $f, g, h$ are described as above and the output of the experiment is $\mathsf{Dec}(\widetilde{\mathsf{L}}, \widetilde{\mathsf{C}}, \widetilde{\mathsf{R}})$ where $\widetilde{\mathsf{L}}, \widetilde{\mathsf{C}}, \widetilde{\mathsf{R}}$ are the outputs of $f, g, h$ respectively.

$\mathsf{Hyb}_2$ : In this hybrid, we generate the shared randomness between $f, g, h$ differently. In particular, instead of fixing the shares $\mathsf{SL}^{\$}_{i_1}, \mathsf{SL}^{\$}_{i_2}, \mathsf{SL}^{\$}_{i_3}$ as the respective shares of a $t$-out-of-$n$ secret sharing of $\mathsf{L}^{\$}$, we will fix them to be the respective shares of a $t$-out-of-$n$ secret sharing of $\mathsf{L}$.

We now claim that $\mathsf{Hyb}_1$ is identically distributed to $\mathsf{Hyb}_2$ and we will show this by using the perfect privacy of a $t$-out-of-$n$ secret sharing scheme. We now give the formal reduction below.

**Claim 5.2** $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$

**Proof** Since $t \geq 4$, we query the challenger of the $t$-out-of-$n$ secret sharing scheme for the shares $\mathsf{SL}_{i_1}, \mathsf{SL}_{i_2}, \mathsf{SL}_{i_3}$ and use them to generate the shared randomness. The rest of the experiment proceeds exactly like in $\mathsf{Hyb}_1$. Note that if the shares correspond to the sharing of $\mathsf{L}_{\$}$, then the output corresponds to $\mathsf{Hyb}_1$ , otherwise, it is distributed identically to $\mathsf{Hyb}_2$. ■

$\underline{\mathsf{Hyb}_3}$ : In this hybrid, we make the following changes with respect to $\mathsf{Hyb}_2$. In generating the shared randomness between $(f, g, h)$, we secret share the real $\mathsf{C}$ using $\mathsf{LRShare}_{(3,n)}$ instead of the fake $\mathsf{C}_\$$. That is, the shares $\mathsf{SC}^\$_{i_1}, \ldots, \mathsf{SC}^\$_{i_t}$ now correspond to the secret sharing of $\mathsf{C}$. We also let the tampering function $f$ to extract $\widetilde{\mathsf{L}}$ using the secret shares of $\mathsf{C}$ instead of $\mathsf{C}_\$$.

We now show that $\mathsf{Hyb}_2 \approx_{\varepsilon_3} \mathsf{Hyb}_3$ by giving a reduction to the leakage resilience property of $(\mathsf{LRShare}_{(3,n)}, \mathsf{LRRec}_{(3,n)})$.

**Claim 5.3** $\mathsf{Hyb}_2 \approx_{\varepsilon_3} \mathsf{Hyb}_3$.

**Proof** Assume for the sake of contradiction that the statistical distance between $\mathsf{Hyb}_2$ and $\mathsf{Hyb}_3$ is greater than $\varepsilon_3$. We will use this to break the leakage resilience property of $(\mathsf{LRShare}_{(3,n)}, \mathsf{LRRec}_{(3,n)})$.

The reduction works as follows:

1. It generates $(\mathsf{L}, \mathsf{C}, \mathsf{R}) \leftarrow \mathsf{Enc}(s)$ and $(\mathsf{L}_\$, \mathsf{R}_\$, \mathsf{C}_\$) \leftarrow \mathsf{Enc}(s_\$)$.

2. It generates $(\mathsf{SL}_1, \ldots, \mathsf{SL}_n)$ as a valid $t$-out-of-$n$ secret sharing of $\mathsf{L}$.

3. It generates $(\mathsf{SR}^\$_1, \ldots, \mathsf{SR}^\$_n)$ as the output of $\mathsf{LRShare}_{(2,n)}(\mathsf{R}_\$)$.

4. It gives $\mathsf{C}$ and $\mathsf{C}_\$$ as the two messages to the leakage resilience challenger and defines the leakage functions as follows:
   - For $i \in \{i_1, i_2\}$, the function outputs $\mathsf{SC}_{i_1}, \mathsf{SC}_{i_2}$ in the clear.
   - For all $i \in \{i_3, \ldots, i_t\}$, the leakage function takes in $\mathsf{SC}_i$ as input, and computes $(\widetilde{\mathsf{SL}}_i, \cdot, \cdot) := f_i(\mathsf{SL}_i, \mathsf{SC}_i, \mathsf{SR}^\$_i)$. It outputs $\widetilde{\mathsf{SL}}_i$.
   - It also receives $\mathsf{aux}^{(3,n)}$ from the challenger.

5. For $i \in \{i_1, i_2\}$, using the values $\mathsf{SC}_i$ from the leakage, it computes $(\widetilde{\mathsf{SL}}_i, \widetilde{\mathsf{SC}}_i, \cdot) := f_i(\mathsf{SL}_i, \mathsf{SC}_i, \mathsf{SR}^\$_i)$.

6. It reconstructs $\widetilde{\mathsf{L}}$ using $\widetilde{\mathsf{SL}}_{i_1}, \ldots, \widetilde{\mathsf{SL}}_{i_t}$.

7. It runs the tampering functions $g$ and $h$ exactly as in $\mathsf{Hyb}_2$ to get $\widetilde{\mathsf{C}}$ and $\widetilde{\mathsf{R}}$.

8. It outputs $\mathsf{Dec}(\widetilde{\mathsf{L}}, \widetilde{\mathsf{C}}, \widetilde{\mathsf{R}})$.

Note that since $|\widetilde{\mathsf{SL}}_i| = m_1$, the leakage functions defined by the reduction belongs to $\mathcal{F}_{t,2,m_1}$. Note that if the leakage was with respect to the sharing of $\mathsf{C}_\$$ then the output of the reduction is identical to $\mathsf{Hyb}_2$ and otherwise, it is distributed identically to $\mathsf{Hyb}_3$. Thus, we break the leakage resilience property of $(\mathsf{LRShare}_{(3,n)}, \mathsf{LRRec}_{(3,n)})$. ■

$\underline{\mathsf{Hyb}_4}$ : In this hybrid, we make a syntactic change with respect to $\mathsf{Hyb}_3$. Instead of the tampering function $g$ sampling $\mathsf{SC}_{i_3}, \ldots, \mathsf{SC}_{i_t}$ again such that it satisfies the two consistency conditions, we let $g$ to use the same shares $\mathsf{SC}_{i_3}, \ldots, \mathsf{SC}_{i_t}$ that were used to generate the shared randomness. $\mathsf{Hyb}_3$ is identical to $\mathsf{Hyb}_4$ from the conditional independence of $\mathsf{LRShare}_{(3,n)}$.

$\underline{\mathsf{Hyb}_5}$ : In this hybrid, we make the following changes with respect to $\mathsf{Hyb}_4$. In constructing the shared randomness between $(f, g, h)$, we set $(\mathsf{SR}^{\$}_{i_1}, \ldots, \mathsf{SR}^{\$}_{i_t})$ as a valid secret sharing of the real $\mathsf{R}$ instead of fake $\mathsf{R}_{\$}$. Additionally, the tampering functions $f, g$, use these shares in order to extract $\widetilde{\mathsf{L}}, \widetilde{\mathsf{C}}$.

We now argue that $\mathsf{Hyb}_4 \approx_{\varepsilon_4} \mathsf{Hyb}_5$ using the leakage resilience property of $(\mathsf{LRShare}_{(2,n)}, \mathsf{LRRec}_{(2,n)})$.

**Claim 5.4** $\mathsf{Hyb}_4 \approx_{\varepsilon_4} \mathsf{Hyb}_5$.

**Proof** Assume for the sake of contradiction that the statistical distance between $\mathsf{Hyb}_4$ and $\mathsf{Hyb}_5$ is greater than $\varepsilon_4$. We will use this to break the leakage resilience property of $(\mathsf{LRShare}_{(2,n)}$, $\mathsf{LRRec}_{(2,n)})$.

The reduction works as follows:

1. It generates $(\mathsf{L}, \mathsf{C}, \mathsf{R}) \leftarrow \mathsf{Enc}(s)$ and $(\mathsf{L}_{\$}, \mathsf{R}_{\$}, \mathsf{C}_{\$}) \leftarrow \mathsf{Enc}(s_{\$})$.
2. It shares $\mathsf{L}$ using $\mathsf{SecShare}_{(t,n)}$ and $\mathsf{C}$ using $\mathsf{LRShare}_{(3,n)}$ to get the shares $(\mathsf{SL}_1, \ldots, \mathsf{SL}_n)$ and $(\mathsf{SC}_1, \ldots, \mathsf{SC}_n)$ respectively.
3. It gives $\mathsf{R}$ and $\mathsf{R}_{\$}$ as the challenge messages to the leakage resilience challenger and defines the leakage functions as follows:
   - The function outputs $\mathsf{SR}_{i_3}$ in the clear.
   - For $i \in \{i_1, i_2\}$, the leakage function takes in $\mathsf{SR}_i$ as input and computes $(\widetilde{\mathsf{SL}}_i, \widetilde{\mathsf{SC}}_i, \cdot) := f_i(\mathsf{SL}_i, \mathsf{SC}_i, \mathsf{SR}_i)$. It then outputs $\widetilde{\mathsf{SL}}_i, \widetilde{\mathsf{SC}}_i$.
   - For each $i \in \{i_4, \ldots, i_t\}$, the leakage function takes in $\mathsf{SR}_i$ as input and computes $(\widetilde{\mathsf{SL}}_i, \cdot, \cdot) := f_i(\mathsf{SL}_i, \mathsf{SC}_i, \mathsf{SR}_i)$. It then outputs $\widetilde{\mathsf{SL}}_i$.
   - It also receives $\mathsf{aux}^{(2,n)}$.
4. For $i = i_3$, using the value $\mathsf{SR}_{i_3}$ from the leakage, it computes $(\widetilde{\mathsf{SL}}_i, \widetilde{\mathsf{SC}}_i, \cdot) := f_i(\mathsf{SL}_i, \mathsf{SC}_i, \mathsf{SR}_i)$.
5. Using the output of the leakage functions, the reduction reconstructs $\widetilde{\mathsf{L}}$ as $\mathsf{SecRec}_{(t,n)}(\widetilde{\mathsf{SL}}_{i_1}, \ldots, \widetilde{\mathsf{SL}}_{i_t})$ and $\widetilde{\mathsf{C}}$ as $\mathsf{LRRec}_{(3,n)}(\widetilde{\mathsf{SC}}_{i_1}, \widetilde{\mathsf{SC}}_{i_2}, \widetilde{\mathsf{SC}}_{i_3})$.
6. It runs the tampering function $h$ exactly as in $\mathsf{Hyb}_4$ to get $\widetilde{\mathsf{R}}$.

Notice that the leakage function defined by the above reduction belongs to the function family $\mathcal{F}_{t,1,\mu}$ since the leakage from each share is $O(m_1 + m_2)$ bits. If the input to the leakage functions where the secret shares of $\mathsf{R}_{\$}$ then the distribution of the reduction's output is identical to $\mathsf{Hyb}_4$. Else, it is distributed identically to $\mathsf{Hyb}_5$. Thus, we break the leakage resilience property of $(\mathsf{LRShare}_{(2,n)}$, $\mathsf{LRRec}_{(2,n)})$. ■

$\underline{\mathsf{Hyb}_6}$ : We again make a syntactic change with respect to $\mathsf{Hyb}_5$. In particular, we let the tampering function $h$ use the same shares $(\mathsf{SR}_{i_1}, \ldots, \mathsf{SR}_{i_t})$ that were used to generate the shared randomness. $\mathsf{Hyb}_5$ is identical to $\mathsf{Hyb}_6$ from the conditional independence property of $\mathsf{LRShare}_{(2,n)}$.

Notice that the distribution of the experiment's output in $\mathsf{Hyb}_6$ is identical to the value of the tampering experiment $\mathsf{Tamper}_s^{f_{\mathsf{ind}},T}$. We know from the split state security of the underlying non-malleable code that there exists a distribution $\mathcal{D}_{f,g,h}$ such that the output of $\mathsf{Hyb}_1$ is $\varepsilon_1$ close to $\text{copy}(\mathcal{D}_{f,g,h}, s)$. Thus, we infer that

$$\text{copy}(\mathcal{D}_{f,g,h}, s) \approx_{\varepsilon_1+\varepsilon_3+\varepsilon_4} \mathsf{Tamper}_s^{f_{\mathsf{ind}},T}$$

which completes the proof of non-malleability.

### 5.3 Rate Analysis

We now instantiate the primitives and provide the rate analysis.

1. We instantiate the three split state non-malleable code from the works of [KOS18, GMW17] (see Theorem 3.19). Using their construction, the $|\mathsf{L}| = |\mathsf{C}| = |\mathsf{R}| = O(m)$ bits and the error $\varepsilon_1 = 2^{-\Omega(m/\log^{1+\rho}(m))}$ for any $\rho > 0$.

2. We use Shamir's secret sharing [Sha79] as the $t$-out-of-$n$ secret sharing scheme. We get $m_1 = O(m)$ whenever $m > \log n$.

3. We instantiate $(\mathsf{LRShare}_{(3,n)}, \mathsf{LRRec}_{(3,n)})$ and $(\mathsf{LRShare}_{(2,n)}, \mathsf{LRRec}_{(2,n)})$ from Theorem 4.12. We get $m_2 = O(n^2(m + \log n))$ and $m_3 = O(n^3(m + \log n))$ by setting $\varepsilon_3$ and $\varepsilon_4$ to be $2^{-\Omega(m/\log m)}$.

Thus the rate of our construction is $\Theta(\frac{1}{n^3})$ and the error is $2^{-\Omega(m/\log^{1+\rho}(m))}$.

### 5.4 Concrete Optimization of Parameters

In this subsection, we will concretely optimize the rate of our construction.

Let us say that we want to share a secret that is $m$ bits long. The construction of 3-split-state non-malleable codes in [KOS18, GMW17] has a rate of $\frac{1}{3}$. Thus, the length of $\mathsf{L}, \mathsf{C}, \mathsf{R}$ is equal to $m$. Since Shamir's secret sharing has rate exactly 1, we deduce that $|\mathsf{SL}_i| = m$. Let us now calculate the size of $|\mathsf{SC}_i|$. The first building block of $\mathsf{SC}_i$ is $(\mathsf{LRShare}_{(2,2)}, \mathsf{LRRec}_{(2,2)})$. The construction of this primitive is based on the inner product two source extractor of Chor and Goldreich [CG88]. To share a message of length $m$ and to tolerate a leakage of $\mu$ bits, the size of each share is $(4m + \mu)$ bits (for an error of $2^{-m}$). We then used this to construct $(\mathsf{LRShare}_{(3,3)}, \mathsf{LRRec}_{(3,3)})$ which has a share size of 2 times the share size of $\mathsf{LRShare}_{(2,2)}$ which is equal to $(8m + 2\mu)$. We then extended this to $(\mathsf{LRShare}_{(3,n)}, \mathsf{LRShare}_{(3,n)})$ using perfect hash function family. From Lemma 4.5, the size of a $(n, 3)$ perfect hash function family turns out to be $(8.27 \log n + 1)$ (when we set $\varepsilon = 1 - 3!/3^3$). We set $\mu = (t-2)|\mathsf{SL}_i| = (t-2) \cdot m$. Thus, the size of $|\mathsf{SC}_i|$ is $(2t+4) \times m \times (8.27 \log n + 1)$. Let us now calculate the size of $\mathsf{SR}_i$. We know explicit $(n, 2)$ perfect hash functions [9] of length $\log n$. For the case of $\mathsf{SR}_i$, we set $\mu = 3(m + |\mathsf{SC}_i|) + (t-3)m = 3|\mathsf{SC}_i| + tm$. Thus, $|\mathsf{SR}_i| = (3|\mathsf{SC}_i| + (t+4)m) \log n$. The error from the work of [KOS18] was $5 \cdot 2^{-\frac{m}{\log^{1+\rho} m}}$ (see Section 4.5.2 and 5.3.1 in their paper). Thus, the total error in our construction is $2^{-m+1} + 5 \cdot 2^{-\frac{m}{\log^{1+\rho} m}} \leq 6 \cdot 2^{-\frac{m}{\log^{1+\rho} m}}$.

[9] The family of $\log n$ functions where the $i$-th function in the family outputs the $i$-th bit of the binary representation of the input is a perfect hash function family.

- $f_{i,j}$ is composed of individual tampering functions $f_1, \ldots, f_n$ where $f_i$ tampers the share $X_i$. Since we have fixed the authorized set to be $\{1, \ldots, t\}$, it is sufficient to consider the tampering functions $f_1, \ldots, f_t$.
- **Shared Randomness.** We choose $(Y_1, \ldots, Y_{i-1}, Y_i, Y_{i+1}, \ldots, Y_t)$ and $(Y_1, \ldots, Y_{i-1}, Y_i', Y_{i+1}, \ldots, Y_t)$ such that $\mathsf{Rec}(Y_1, \ldots, Y_{i-1}, Y_i, Y_{i+1}, \ldots, Y_t) \neq \mathsf{Rec}(Y_1, \ldots, Y_{i-1}, Y_i', Y_{i+1}, \ldots, Y_t)$. Note that by statistical privacy of $t$-out-of-$n$ secret sharing scheme such values must exist. For all $k \neq i$, the function $f_k$ has $Y_k$ hardwired. $f_i$ has $Y_i$ and $Y_i'$ hardwired.
- On input $\mathsf{share}_k$ for each $k \neq i$, $f_k$ outputs $Y_k$. On input $\mathsf{share}_i$, $f_i$ outputs $Y_i$ if the $j$-th bit of $\mathsf{share}_i$ is 0 and otherwise, outputs $Y_i'$.

**Figure 4**: Description of the Tampering Function $f_{i,j}$

# 6 Unbounded Tamperings: Impossibility Result

We now consider the stronger non-malleability requirement wherein multiple tampering functions can tamper the shares of a secret and we require that the joint distribution of reconstructed tampered shares to be independent of the original secret. In this section, we give an impossibility result for the case of apriori-unbounded number of tamperings. Then, in Section 7, we give a matching positive result for the bounded tampering setting and for any general access structure. This impossibility result generalizes a similar impossibility result for the case of split-state non-malleable codes [GLM$^+$04, FMNV14].

## 6.1 Proof

We now give the formal description of a set of tampering functions along with a distinguisher that can break the non-malleability property for every $t$-out-of-$n$ secret sharing scheme when the number of tampering functions is allowed to grow with the threshold $t$ and the size of each share.

Let us assume that we are given a $t$-out-of-$n$ secret sharing scheme $(\mathsf{Share}, \mathsf{Rec})$. For an arbitrary secret $m \in \mathcal{M}$, let $(\mathsf{share}_1, \ldots, \mathsf{share}_n) \leftarrow \mathsf{Share}(m)$. We assume w.l.o.g. that the size of each $\mathsf{share}_i$ is exactly the same and this is equal to $s$.

We will set the authorized set $T$ to be $\{1, \ldots, t\}$ and this will be the same for each tampering function. We give a set of tampering functions $f_{1,1}, f_{1,2}, \ldots, f_{t,s} \in \mathcal{F}_{\mathsf{ind}}$ that have the property that the secret reconstructed from tampering by $f_{i,j}$ reveals the $j$-th bit of $\mathsf{share}_i$. Thus, given all the $t \cdot s$ reconstructed secrets, the distinguisher can trivially learn the message by running the reconstructed algorithm on $\mathsf{share}_1, \ldots, \mathsf{share}_t$. We now give the description of $f_{i,j}$ in Figure 4.

It follows from the description of $f_{i,j}$ that if the $j$-th bit of $\mathsf{share}_i$ is 0 then the tampered message is $\mathsf{Rec}(Y_1, \ldots, Y_{i-1}, Y_i, Y_{i+1}, \ldots, Y_t)$ and otherwise the tampered message is $\mathsf{Rec}(Y_1, \ldots, Y_{i-1}, Y_i', Y_{i+1}, \ldots, Y_t)$. Since these two values are not equal, the distinguisher can infer the $j$-th bit of $\mathsf{share}_i$ based on the tampered message.

# 7 NMSS for General Access Structures with Multiple Tampering

We first define the notion of non-malleable secret sharing for general access structures in the next subsection. This is followed by the construction and proof in the subsequent subsections.

## 7.1 Definitions

First, we recall the definition of a secret sharing scheme for a general monotone access structure $\mathcal{A}$ - a generalization of the one defined for threshold access structures in Definition 3.10.

**Definition 7.1 ($(\mathcal{A}, n, \varepsilon_c, \varepsilon_s)$-Secret Sharing Scheme)** *Let $\mathcal{M}$ be a finite set of secrets, where $|\mathcal{M}| \geq 2$. Let $[n] = \{1, 2, \ldots, n\}$ be a set of identities (indices) of $n$ parties. A sharing function* $\mathsf{Share}$ *with domain of secrets $\mathcal{M}$ is a $(\mathcal{A}, n, \varepsilon_c, \varepsilon_s)$-secret sharing scheme with respect to monotone access structure $\mathcal{A}$ if the following two properties hold :*

- ***Correctness:*** *The secret can be reconstructed by any set of parties that are part of the access structure $\mathcal{A}$. That is, for any set $T \in \mathcal{A}$, there exists a deterministic reconstruction function* $\mathsf{Rec} : \otimes_{i \in T} \mathcal{S}_i \to \mathcal{M}$ *such that for every $m \in \mathcal{M}$,*

$$\Pr[\mathsf{Rec}(\mathsf{Share}(m)_T) = m] = 1 - \varepsilon_c$$

  *where the probability is over the randomness of the* $\mathsf{Share}$ *function. We will slightly abuse the notation and denote* $\mathsf{Rec}$ *as the reconstruction procedure that takes in $T \in \mathcal{A}$ and* $\mathsf{Share}(m)_T$ *as input and outputs the secret.*

- ***Statistical Privacy:*** *Any collusion of parties not part of the access structure should have "almost" no information about the underlying secret. More formally, for any unauthorized set $U \subseteq [n]$ such that $U \notin \mathcal{A}$, and for every pair of secrets $m_0, m_1 \in M$, for any distinguisher $D$ with output in $\{0, 1\}$, the following holds :*

$$|\Pr[D(\mathsf{Share}(m_0)_U) = 1] - \Pr[D(\mathsf{Share}(m_1)_U) = 1]| \leq \varepsilon_s$$

*We define the rate of the secret sharing scheme as* $\lim_{|m| \to \infty} \frac{|m|}{\max_{i \in [n]} |\mathsf{Share}(m)_i|}$

We now define the notion of a non-malleable secret sharing scheme for general access structures which is a generalization of the definition for threshold access structures given in Definition 3.11.

**Definition 7.2 (Non-Malleable Secret Sharing for General Access Structures [GK18b])** *Let* $(\mathsf{Share}, \mathsf{Rec})$ *be a $(\mathcal{A}, n, \varepsilon_c, \varepsilon_s)$-secret sharing scheme for message space $\mathcal{M}$ and access structure $\mathcal{A}$. Let $\mathcal{F}$ be a family of tampering functions. For each $f \in \mathcal{F}$, $m \in \mathcal{M}$ and authorized set $T \in \mathcal{A}$, define the tampered distribution* $\mathsf{Tamper}_m^{f,T}$ *as* $\mathsf{Rec}(f(\mathsf{Share}(m))_T)$ *where the randomness is over the sharing function* $\mathsf{Share}$. *We say that the $(\mathcal{A}, n, \varepsilon_c, \varepsilon_s)$-secret sharing scheme,* $(\mathsf{Share}, \mathsf{Rec})$ *is $\varepsilon'$-non-malleable w.r.t. $\mathcal{F}$ if for each $f \in \mathcal{F}$ and any authorized set $T \in \mathcal{A}$, there exists a distribution $D^{f,T}$ over $\mathcal{M} \cup \{\mathsf{same}^\star\}$ such that:*

$$|\mathsf{Tamper}_m^{f,T} - \mathrm{copy}(D^{f,T}, m)| \leq \varepsilon'$$

*where* copy *is defined by* $\mathrm{copy}(x, y) = \begin{cases} x & \text{if } x \neq \mathsf{same}^\star \\ y & \text{if } x = \mathsf{same}^\star \end{cases}$.

**Many Tampering Extension.** Similar to the threshold case, we now extend the above definition to capture multiple tampering attacks.

**Definition 7.3** *Let* $(\mathsf{Share}, \mathsf{Rec})$ *be a* $(\mathcal{A}, n, \varepsilon_c, \varepsilon_s)$*-secret sharing scheme for message space* $\mathcal{M}$*. Let* $\mathcal{F}$ *be some family of tampering functions. For* $\overrightarrow{f} = (f_1, \dots, f_{\mathsf{K}}) \in \mathcal{F}^{\mathsf{K}}$*,* $m \in \mathcal{M}$ *and authorized set* $T \in \mathcal{A}$*, we define the tampered distribution* $\mathsf{Tamper}_m^{\overrightarrow{f},T}$ *as* $\big(\mathsf{Rec}(f_1(\mathsf{shares})_T), \dots, \mathsf{Rec}(f_t(\mathsf{shares})_T) : \mathsf{shares} \leftarrow \mathsf{Share}(m)\big)$ *where the randomness is over the sharing function* $\mathsf{Share}$*. We say that the* $(\mathcal{A}, n, \varepsilon_c, \varepsilon_s)$*-secret sharing scheme,* $(\mathsf{Share}, \mathsf{Rec})$ *is* $\varepsilon'$*-non-malleable with tampering degree* $\mathsf{K}$ *w.r.t.* $\mathcal{F}$ *if for each* $\overrightarrow{f} \in \mathcal{F}^{\mathsf{K}}$ *and any authorized set* $T \in \mathcal{A}$*, there exists a distribution* $D^{\overrightarrow{f},T}$ *over* $(\mathcal{M} \cup \{\mathsf{same}^\star\})^{\mathsf{K}}$ *such that:*

$$|\mathsf{Tamper}_m^{\overrightarrow{f},T} - \widetilde{\text{copy}}(D^{\overrightarrow{f},T}, m)| \leq \varepsilon'$$

*where* $\widetilde{\text{copy}}$ *is defined by* $\widetilde{\text{copy}}(\overrightarrow{x}, y) = (z_1, ..., z_n)$ *where* $z_i = \begin{cases} x_i & \text{if } x_i \neq \mathsf{same}^\star \\ y & \text{if } x_i = \mathsf{same}^\star \end{cases}$ *..*

**Remark 7.4** *As in the threshold case, it is possible to further strengthen the above definition by requiring the output of every tampering function* $f_i$ *to use a different authorized set* $T_i$ *for reconstruction. We once again note that our construction does not satisfy this stronger definition. However, recall that the impossibility of apriori unbounded number of tamperings holds even with respect to the weakened definition of using the same authorized set for reconstruction in every tampering and even in the case of just threshold access structures.*

## 7.2 Construction

In this section, we show how to build a one-many non-malleable secret sharing scheme for general access structures.

First, let $(\mathsf{SecShare}_{(\mathcal{A},n)}, \mathsf{SecRec}_{(\mathcal{A},n)})$ be any statistically private secret sharing scheme with rate $\mathsf{R}$ for a 4-monotone access structure $\mathcal{A}$ over $n$ parties. We refer the reader to [KW93, LV18] for explicit constructions.

Let $\mathsf{t}_{\mathsf{max}}$ denote the maximum size of a minimal authorized set of $\mathcal{A}$.[10] We give a construction of a Non-Malleable Secret Sharing scheme with tampering degree $\mathsf{K}$ for a 4-monotone access structure $\mathcal{A}$ with rate $O(\frac{\mathsf{R}}{\mathsf{K}^3 \mathsf{t}_{\mathsf{max}} \log^2 n})$ with respect to a individual tampering function family $\mathcal{F}_{ind}$.

**Building Blocks.** The construction uses the following building blocks. We instantiate them with concrete schemes later:

- A one-many 3-split-state non-malleable code $(\mathsf{Enc}, \mathsf{Dec})$ where $\mathsf{Enc} : \mathcal{M} \to \mathcal{L} \times \mathcal{C} \times \mathcal{R}$, the simulation error of the scheme is $\varepsilon_1$ and the scheme is secure against $\mathsf{K}$ tamperings. Furthermore, we assume that for any two messages $m, m' \in \mathcal{M}$, $(\mathsf{C}, \mathsf{R}) \approx_{\varepsilon_2} (\mathsf{C}', \mathsf{R}')$ where $(\mathsf{L}, \mathsf{C}, \mathsf{R}) \leftarrow \mathsf{Enc}(m)$ and $(\mathsf{L}', \mathsf{C}', \mathsf{R}') \leftarrow \mathsf{Enc}(m')$.
- A $(\mathcal{A}, n, 0, 0)$ (where $\mathcal{A}$ is 4-monotone) secret sharing scheme $(\mathsf{SecShare}_{(\mathcal{A},n)}, \mathsf{SecRec}_{(\mathcal{A},n)})$ with perfect privacy for message space $\mathcal{L}$.[11] We will assume that the size of each share is $m_1$.

[10] We refer the reader to Definition 1.5, Definition 1.6 for definitions of 4-monotone access structures and minimal authorized set.

[11] We note that our proof of security goes through even if this secret sharing scheme only has statistical privacy.

- A $(3, n, \varepsilon'_3, 0)$ secret sharing scheme $(\mathsf{LRShare}_{(3,n)}, \mathsf{LRRec}_{(3,n)})$ that is $\varepsilon_3$-leakage resilient against leakage functions $\mathcal{F}_{\mathsf{t_{max}},2,\mathsf{K}m_1}$ for message space $\mathcal{C}$. We assume that the size of each share is $m_2$.
- A $(2, n, \varepsilon'_4, 0)$ secret sharing scheme $(\mathsf{LRShare}_{(2,n)}, \mathsf{LRRec}_{(2,n)})$ for message space $\mathcal{R}$ that is $\varepsilon_4$-leakage resilient against leakage functions $\mathcal{F}_{\mathsf{t_{max}},1,\vec{\mu}}$ where $\max_T \sum_{i \in T, T \in \mathcal{A}, |T| = \mathsf{t_{max}}} \mu_i = O(\mathsf{K}m_2 + \mathsf{K}\mathsf{t_{max}}m_1)$. We assume that the size of each share is $m_3$.

**Construction.** The construction is very similar to the construction of non-malleable secret sharing for threshold access structures given in Section 5 with the only difference being that we now use the $(\mathcal{A}, n, 0, 0)$ secret sharing scheme. Note that in the construction we additionally need a procedure to find a minimal authorized set from any authorized set. This procedure is efficient if we can efficiently test the membership in $\mathcal{A}$. We point the reader to [GK18b] for details of this procedure. We give the formal description of the construction in Figure 5 for completeness.

$\mathsf{Share}(m)$ : To share a secret $s \in \mathcal{M}$ do:

1. Encode the secret $s$ as $(\mathsf{L}, \mathsf{C}, \mathsf{R}) \leftarrow \mathsf{Enc}(s)$.
2. Compute the shares
$$(\mathsf{SL}_1, \ldots, \mathsf{SL}_n) \leftarrow \mathsf{SecShare}_{(\mathcal{A},n)}(\mathsf{L})$$
$$(\mathsf{SC}_1, \ldots, \mathsf{SC}_n) \leftarrow \mathsf{LRShare}_{(3,n)}(\mathsf{C})$$
$$(\mathsf{SR}_1, \ldots, \mathsf{SR}_n) \leftarrow \mathsf{LRShare}_{(2,n)}(\mathsf{R})$$
3. For each $i \in [n]$, set $\mathsf{share}_i$ as $(\mathsf{SL}_i, \mathsf{SC}_i, \mathsf{SR}_i)$ and output $(\mathsf{share}_1, \ldots, \mathsf{share}_n)$ as the set of shares.

$\mathsf{Rec}(\mathsf{Share}(m)_T)$ : Given a set of shares in an authorized set $T' \in \mathcal{A}$, let $T \subseteq T'$ denote a minimal authorized set. To reconstruct the secret from the shares in set $T$, (of size at most $\mathsf{t_{max}}$) do:

1. Let the shares corresponding to the set $T$ be $(\mathsf{share}_{i_1}, \ldots, \mathsf{share}_{i_{\mathsf{t_{max}}}})$.
2. For each $j \in \{i_1, \ldots, i_{\mathsf{t_{max}}}\}$, parse $\mathsf{share}_j$ as $(\mathsf{SL}_j, \mathsf{SC}_j, \mathsf{SR}_j)$.
3. Reconstruct
$$\mathsf{L} := \mathsf{SecRec}_{(\mathcal{A},n)}(\mathsf{SL}_{i_1}, \ldots, \mathsf{SL}_{i_{\mathsf{t_{max}}}})$$
$$\mathsf{C} := \mathsf{LRRec}_{(3,n)}(\mathsf{SC}_{i_1}, \mathsf{SC}_{i_2}, \mathsf{SC}_{i_3})$$
$$\mathsf{R} := \mathsf{LRRec}_{(2,n)}(\mathsf{SR}_{i_1}, \mathsf{SR}_{i_2})$$
4. Output the secret $s$ as $\mathsf{Dec}(\mathsf{L}, \mathsf{C}, \mathsf{R})$.

**Figure 5**: Construction of Non-Malleable Secret Sharing Scheme for General Access Structures against Multiple Tampering

**Theorem 7.5** *There exists a constant $\gamma > 0$ such that, for any arbitrary $n, \mathsf{K} \in \mathbb{N}$ and 4-monotone access structure $\mathcal{A}$, the construction given in Figure 5 is a $(\mathcal{A}, n, \varepsilon_3' + \varepsilon_4', \varepsilon_2)$ secret sharing scheme for messages of length $m$ where $m \geq \mathsf{K}^{\gamma}$. Furthermore, it is $(\varepsilon_1 + \varepsilon_3 + \varepsilon_4)$ one-many non-malleable with tampering degree $\mathsf{K}$ with respect to tampering function family $\mathcal{F}_{ind}$.*

## 7.3 Proof of Theorem 7.5

In this section, we argue correctness, statistical privacy and non-malleability to complete the proof of Theorem 7.5.

**Correctness.** Similar to Section 5, we notice that except with probability $\varepsilon_3' + \varepsilon_4'$, $\mathsf{SecRec}_{(\mathcal{A},n)}$, $\mathsf{LRRec}_{(3,n)}$ and $\mathsf{LRRec}_{(2,n)}$ will be able to reconstruct $\mathsf{L}, \mathsf{C}$, and $\mathsf{R}$ respectively. The correctness now follows directly from the correctness of the decoder $\mathsf{Dec}$ of one-many 3-split-state non-malleable codes.

**Statistical Privacy.** To argue statistical privacy, we consider a sequence of hybrids very similar to Section 5.

- $\mathsf{Hyb}_1$ : In this hybrid, the secret $s$ is shared using $\mathsf{Share}(s)$.
- $\mathsf{Hyb}_2$ : This hybrid is same as $\mathsf{Hyb}_1$ except that we do the following. Let $(\mathsf{L}, \mathsf{C}, \mathsf{R}) \leftarrow \mathsf{Enc}(s)$. We reset $\mathsf{L} = \perp$ and encode this fake $\mathsf{L}$ using $\mathsf{SecShare}_{(\mathcal{A},n)}$.
- $\mathsf{Hyb}_3$ : This hybrid is same as $\mathsf{Hyb}_2$ except that we sample $(\mathsf{L}', \mathsf{C}', \mathsf{R}') \leftarrow \mathsf{Enc}(s')$ and encode $\mathsf{C}', \mathsf{R}'$ instead of $\mathsf{C}, \mathsf{R}$ using $\mathsf{LRShare}_{(3,n)}$ and $\mathsf{LRShare}_{(2,n)}$ respectively.
- $\mathsf{Hyb}_4$ : This hybrid is distributed identically to $\mathsf{Share}(s')$.

We first claim that $\mathsf{Hyb}_1$ is distributed identically to $\mathsf{Hyb}_2$ and $\mathsf{Hyb}_3$ is distributed identically to $\mathsf{Hyb}_4$. This actually follows directly from the security of $\mathsf{SecShare}_{(\mathcal{A},n)}$ since a set of shares belonging to an unauthorized set perfectly hide $\mathsf{L}$.

We now argue that $\mathsf{Hyb}_2$ is statistically close to $\mathsf{Hyb}_3$. Note that in $\mathsf{Hyb}_2$, $(\mathsf{C}, \mathsf{R})$ is generated as part of an encoding of $s$ and in $\mathsf{Hyb}_3$ it is generated as part of an encoding of $s'$. From the property of our one-many 3-state non-malleable code, we know that $(\mathsf{C}, \mathsf{R})$ statistically hide the message, and hence we infer that $\mathsf{Hyb}_2$ is statistically close to $\mathsf{Hyb}_3$.

**Non-Malleability.** We show the non-malleability of our scheme by transforming a tampering attack on the shares of our scheme to a tampering attack on the one-many 3-state non-malleable code. In particular, we will use the set of $\mathsf{K}$ tampering functions that attack the secret sharing scheme to design a set of $\mathsf{K}$ split state tampering functions against the underlying non-malleable code. We then use the security of the underlying non-malleable code to come up with the simulator for our scheme.

Let $\overrightarrow{\mathsf{fun}} = (\mathsf{fun}_1, \ldots, \mathsf{fun}_{\mathsf{K}}) \in \mathcal{F}_{ind}^{\mathsf{K}}$ be a set of tampering functions where, for each $k \in [\mathsf{K}]$, $\mathsf{fun}_k$ consists of a set $(f_{k,1}, \ldots, f_{k,n}) \in \mathcal{F}_{\mathsf{ind}}$. Let $T = \{i_1, \ldots, i_{\mathsf{t}_{\max}}\}$ be a minimal authorized set. The split state functions $\{(f_k, g_k, h_k)\}_{k \in [\mathsf{K}]}$ that attack the underlying code are constructed as follows.

- **Shared Randomness.** Let $s_{\$}$ be an arbitrary secret and let $(\mathsf{L}_{\$}, \mathsf{C}_{\$}, \mathsf{R}_{\$}) \leftarrow \mathsf{Enc}(s_{\$})$. Run the sharing function $\mathsf{SecShare}_{(\mathcal{A},n)}$, $\mathsf{LRShare}_{(3,n)}$ and $\mathsf{LRShare}_{(2,n)}$ on $(\mathsf{L}_{\$}, \mathsf{C}_{\$}, \mathsf{R}_{\$})$ respectively to obtain $(\mathsf{SL}^{\$}_1, \ldots, \mathsf{SL}^{\$}_n)$, $(\mathsf{SC}^{\$}_1, \ldots, \mathsf{SC}^{\$}_n)$ and $(\mathsf{SR}^{\$}_1, \ldots, \mathsf{SR}^{\$}_n)$. For each $i \in [n]$, set $\mathsf{share}^{\$}_i = (\mathsf{SL}^{\$}_i, \mathsf{SC}^{\$}_i, \mathsf{SR}^{\$}_i)$.

  For each $k \in [\mathsf{K}]$, for $i \in \{i_1, i_2, i_3\}$, run the tampering functions $f_{k,i}$ on input $(\mathsf{SL}^{\$}_i, \mathsf{SC}^{\$}_i, \mathsf{SR}^{\$}_i)$ to obtain the tampered values $(\widetilde{\mathsf{SL}}^{\$}_{k,i}, \widetilde{\mathsf{SC}}^{\$}_{k,i}, \widetilde{\mathsf{SR}}^{\$}_{k,i})$. The shared randomness between $f, g$ and $h$ comprises of the following:

$$(\mathsf{SL}^{\$}_{i_1}, \mathsf{SL}^{\$}_{i_2}, \mathsf{SL}^{\$}_{i_3})$$

$$\{(\widetilde{\mathsf{SL}}^{\$}_{k,i_1}, \widetilde{\mathsf{SL}}^{\$}_{k,i_2}, \widetilde{\mathsf{SL}}^{\$}_{k,i_3})\}_{k \in [\mathsf{K}]}$$

$$(\mathsf{SC}^{\$}_{i_1}, \mathsf{SC}^{\$}_{i_2}, \mathsf{SC}^{\$}_{i_4}, \ldots, \mathsf{SC}^{\$}_{i_{\mathsf{tmax}}})$$

$$\{(\widetilde{\mathsf{SC}}^{\$}_{k,i_1}, \widetilde{\mathsf{SC}}^{\$}_{k,i_2})\}_{k \in [\mathsf{K}]}$$

$$(\mathsf{SR}^{\$}_{i_3}, \ldots, \mathsf{SR}^{\$}_{i_{\mathsf{tmax}}})$$

- **Functions** $\{f_k\}_{k \in [\mathsf{K}]}$**.** For each $k \in [\mathsf{K}]$, the tampering function $f_k$ on input $\mathsf{L}$ does the following:

  1. It chooses $\mathsf{SL}_{i_4}, \ldots, \mathsf{SL}_{i_{\mathsf{tmax}}}$ such that $(\mathsf{SL}^{\$}_{i_1}, \mathsf{SL}^{\$}_{i_2}, \mathsf{SL}^{\$}_{i_3}, \mathsf{SL}_{i_4}, \ldots, \mathsf{SL}_{i_{\mathsf{tmax}}})$ is a valid $(\mathcal{A}, n)$ secret sharing of $\mathsf{L}$.
  2. For every $i \in \{i_4, \ldots, i_{\mathsf{tmax}}\}$, it runs the tampering function $f_{k,i}$ on input $(\mathsf{SL}_i, \mathsf{SC}^{\$}_i, \mathsf{SR}^{\$}_i)$ to obtain $(\widetilde{\mathsf{SL}}_{k,i}, \widetilde{\mathsf{SC}}^{\$}_{k,i}, \widetilde{\mathsf{SR}}^{\$}_{k,i})$.
  3. It runs $\mathsf{SecRec}_{(\mathcal{A},n)}$ on inputs $(\widetilde{\mathsf{SL}}^{\$}_{k,i_1}, \widetilde{\mathsf{SL}}^{\$}_{k,i_2}, \widetilde{\mathsf{SL}}^{\$}_{k,i_3}, \widetilde{\mathsf{SL}}_{k,i_4}, \ldots, \widetilde{\mathsf{SL}}_{k,i_{\mathsf{tmax}}})$ to obtain $\widetilde{\mathsf{L}}_k$ and outputs it.

- **Functions** $\{g_k\}_{k \in [\mathsf{K}]}$**.** For each $k \in [\mathsf{K}]$, the tampering function $g_k$ on input $\mathsf{C}$ does the following:

  1. Samples $\mathsf{SC}_{i_3}, \ldots, \mathsf{SC}_{i_{\mathsf{tmax}}}$ such that the following two conditions are satisfied:
     (a) $(\mathsf{SC}^{\$}_{i_1}, \mathsf{SC}^{\$}_{i_2}, \mathsf{SC}_{i_3}, \ldots, \mathsf{SC}_{i_{\mathsf{tmax}}})$ are valid shares of $\mathsf{LRShare}_{(3,n)}(\mathsf{C})$.
     (b) $f_{k,i_3}(\mathsf{SL}^{\$}_{i_3}, \mathsf{SC}_{i_3}, \mathsf{SR}^{\$}_{i_3}) = (\widetilde{\mathsf{SL}}^{\$}_{k,i_3}, \cdot, \cdot)$. That is, the first component of the output of $f_{k,i_3}$ on input $\mathsf{SL}^{\$}_{i_3}, \mathsf{SC}_{i_3}, \mathsf{SR}^{\$}_{i_3}$ is equal to $\widetilde{\mathsf{SL}}^{\$}_{k,i_3}$ (which is part of the shared randomness).
  2. In case such a sampling is not possible, it outputs the special symbol $\mathsf{abort}_1$.
  3. Else, it runs $f_{k,i_3}$ in input $\mathsf{SL}^{\$}_{i_3}, \mathsf{SC}_{i_3}, \mathsf{SR}^{\$}_{i_3}$ to obtain $(\widetilde{\mathsf{SL}}^{\$}_{k,i_3}, \widetilde{\mathsf{SC}}_{k,i_3}, \widetilde{\mathsf{SR}'}^{\$}_{k,i_3})$.
  4. Reconstructs $\widetilde{C}_k$ by running $\mathsf{LRRec}_{(3,n)}(\widetilde{\mathsf{SC}}^{\$}_{k,i_1}, \widetilde{\mathsf{SC}}^{\$}_{k,i_2}, \widetilde{\mathsf{SC}}_{k,i_3})$ and outputs it.

- **Functions** $\{h_k\}_{k \in [\mathsf{K}]}$**.** For each $k \in [\mathsf{K}]$, the tampering function $h_k$ on input $\mathsf{R}$ does the following:

  1. Samples $\mathsf{SR}_{i_1}, \mathsf{SR}_{i_2}, \mathsf{SR}_{i_4}, \ldots, \mathsf{SR}_{i_{\mathsf{tmax}}}$ such that the following three conditions are satisfied:

(a) $\mathsf{SR}_{i_1}, \mathsf{SR}_{i_2}, \mathsf{SR}^{\$}_{i_3}, \mathsf{SR}_{i_4}, \ldots, \mathsf{SR}_{i_{\mathsf{tmax}}}$ are valid shares of $\mathsf{LRShare}_{(2,n)}(\mathsf{R})$.

(b) $f_{k,i_1}(\mathsf{SL}^{\$}_{i_1}, \mathsf{SC}^{\$}_{i_1}, \mathsf{SR}_{i_1}) = (\widetilde{\mathsf{SL}}^{\$}_{k,i_1}, \widetilde{\mathsf{SC}}^{\$}_{k,i_1}, \cdot)$. In other words, the first two components of the output of $f_{k,i_1}$ on input $\mathsf{SL}^{\$}_{i_1}, \mathsf{SC}^{\$}_{i_1}, \mathsf{SR}_{i_1}$ is same as $\widetilde{\mathsf{SL}}^{\$}_{k,i_1}, \widetilde{\mathsf{SC}}^{\$}_{k,i_1}$ (which are part of shared randomness).

(c) $f_{k,i_2}(\mathsf{SL}^{\$}_{i_2}, \mathsf{SC}^{\$}_{i_2}, \mathsf{SR}_{i_2}) = (\widetilde{\mathsf{SL}}^{\$}_{k,i_2}, \widetilde{\mathsf{SC}}^{\$}_{k,i_2}, \cdot)$.

2. If such a sampling is not possible, it outputs the special symbol $\mathsf{abort}_2$.
3. Otherwise, for $i \in \{i_1, i_2\}$, it runs $f_{k,i}(\mathsf{SL}^{\$}_{i}, \mathsf{SC}^{\$}_{i}, \mathsf{SR}_{i})$ to obtain $(\widetilde{\mathsf{SL}}^{\$}_{k,i}, \widetilde{\mathsf{SC}}^{\$}_{k,i}, \widetilde{\mathsf{SR}}_{k,i})$.
4. Reconstructs $\widetilde{\mathsf{R}}_k$ by running $\mathsf{LRRec}_{(2,n)}(\widetilde{\mathsf{SR}}_{k,i_1}, \widetilde{\mathsf{SR}}_{k,i_2})$ and outputs it.

The functions $\{f_k, g_k, h_k\}_{k \in [\mathsf{K}]}$ described above constitute a split state tampering function family for the non-malleable code. In order to reduce the tampering attack on the shares of our secret sharing scheme to the shares of the underlying code, we need to show that the distribution of the shares given as inputs to $(\mathsf{fun}_1, \ldots, \mathsf{fun}_{\mathsf{K}})$ in the tampering experiment $\mathsf{Tamper}_s^{f_{(\mathrm{many},\mathsf{K})}, T}$ is statistically close to the distribution of the shares that $\{f_k, g_k, h_k\}_{k \in [\mathsf{K}]}$ give as input to these functions. We show this via a hybrid argument that is very similar to Section 5.

$\underline{\mathsf{Hyb}_1}$ : This is same as the experiment where $\{f_k, g_k, h_k\}_{k \in [\mathsf{K}]}$ are described as above and the output of the experiment is $\{\mathsf{Dec}(\widetilde{\mathsf{L}}_k, \widetilde{\mathsf{C}}_k, \widetilde{\mathsf{R}}_k)\}_{k \in [\mathsf{K}]}$ where $\widetilde{\mathsf{L}}_k, \widetilde{\mathsf{C}}_k, \widetilde{\mathsf{R}}_k$ are the outputs of $f_k, g_k, h_k$ respectively.

$\underline{\mathsf{Hyb}_2}$ : In this hybrid, we generate the shared randomness between $\{f_k, g_k, h_k\}_{k \in [\mathsf{K}]}$ differently. In particular, instead of fixing the shares $\mathsf{SL}^{\$}_{i_1}, \mathsf{SL}^{\$}_{i_2}, \mathsf{SL}^{\$}_{i_3}$ as the respective shares of a $(\mathcal{A}, n, 0, 0)$ secret sharing of $\mathsf{L}^{\$}$, we will fix them to be the respective shares of a $(\mathcal{A}, n, 0, 0)$ secret sharing of $\mathsf{L}$.

We now claim that $\mathsf{Hyb}_1$ is identically distributed to $\mathsf{Hyb}_2$ and we will show this by using the perfect privacy of $(\mathcal{A}, n, 0, 0)$ secret sharing scheme. We now give the formal reduction below.

**Claim 7.6** $\mathsf{Hyb}_1 \equiv \mathsf{Hyb}_2$

**Proof** Since the access structure $\mathcal{A}$ is 4-monotone, we query the challenger of the $(\mathcal{A}, n, 0, 0)$ secret sharing scheme for the shares $\mathsf{SL}_{i_1}, \mathsf{SL}_{i_2}, \mathsf{SL}_{i_3}$ and use them to generate the shared randomness. The rest of the experiment proceeds exactly like in $\mathsf{Hyb}_1$. Note that if the shares correspond to the sharing of $\mathsf{L}_{\$}$, then the output corresponds to $\mathsf{Hyb}_1$. Otherwise, it is distributed identically to $\mathsf{Hyb}_2$. ■

$\underline{\mathsf{Hyb}_{2.5}}$ : In this hybrid, we make a syntactic change with respect to $\mathsf{Hyb}_2$. In particular, instead of allowing each $f_k$ to sample its own shares $\mathsf{SL}_{i_4}, \ldots, \mathsf{SL}_{i_{\mathsf{tmax}}}$ such that it is a valid secret sharing of $\mathsf{L}$, we will make them use the same shares $\mathsf{SL}_{i_4}, \ldots, \mathsf{SL}_{i_{\mathsf{tmax}}}$ that were used to generate the shared randomness. That is, each $f_k$ will use the same set of shares to extract $\widetilde{\mathsf{L}}_k$. This change is only syntactic and $\mathsf{Hyb}_{2.5}$ is identically distributed to $\mathsf{Hyb}_3$.

$\underline{\mathsf{Hyb}_3}$ : In this hybrid, we make the following changes with respect to $\mathsf{Hyb}_{2.5}$. In generating the shared randomness between $\{f_k, g_k, h_k\}_{k\in[\mathsf{K}]}$, we secret share the real $\mathsf{C}$ using $\mathsf{LRShare}_{(3,n)}$ instead of the fake $\mathsf{C}_{\$}$. That is, the shares $\mathsf{SC}^{\$}_{i_1}, \ldots, \mathsf{SC}^{\$}_{i_{\mathsf{tmax}}}$ now correspond to the secret sharing of $\mathsf{C}$. We also let the tampering function $f_k$ for each $k \in [\mathsf{K}]$ to extract $\widetilde{\mathsf{L}}_k$ using the secret shares of $\mathsf{C}$ instead of $\mathsf{C}_{\$}$.

We now show that $\mathsf{Hyb}_{2.5} \approx_{\varepsilon_3} \mathsf{Hyb}_3$ by giving a reduction to the leakage resilience property of $(\mathsf{LRShare}_{(3,n)}, \mathsf{LRRec}_{(3,n)})$.

**Claim 7.7** $\mathsf{Hyb}_{2.5} \approx_{\varepsilon_3} \mathsf{Hyb}_3$.

**Proof** Assume for the sake of contradiction that the statistical distance between $\mathsf{Hyb}_{2.5}$ and $\mathsf{Hyb}_3$ is greater than $\varepsilon_3$. We will use this to break the leakage resilience property of $(\mathsf{LRShare}_{(3,n)}, \mathsf{LRRec}_{(3,n)})$.

The reduction works as follows:

1. It generates $(\mathsf{L}, \mathsf{C}, \mathsf{R}) \leftarrow \mathsf{Enc}(s)$ and $(\mathsf{L}_{\$}, \mathsf{R}_{\$}, \mathsf{C}_{\$}) \leftarrow \mathsf{Enc}(s_{\$})$.

2. It generates $(\mathsf{SL}_1, \ldots, \mathsf{SL}_n)$ as a valid $(\mathcal{A}, n, 0, 0)$ secret sharing of $\mathsf{L}$.

3. It generates $(\mathsf{SR}^{\$}_1, \ldots, \mathsf{SR}^{\$}_n)$ as the output of $\mathsf{LRShare}_{(2,n)}(\mathsf{R}_{\$})$.

4. It gives $\mathsf{C}$ and $\mathsf{C}_{\$}$ as the two messages to the leakage resilience challenger and defines the leakage functions as follows:

   - For $i \in \{i_1, i_2\}$, the function outputs $\mathsf{SC}_{i_1}, \mathsf{SC}_{i_2}$ in the clear.
   - For all $i \in \{i_3, \ldots, i_{\mathsf{tmax}}\}$: The leakage function takes in $\mathsf{SC}_i$ as input, computes $\{(\widetilde{\mathsf{SL}}_{k,i}, \cdot, \cdot) := f_{k,i}(\mathsf{SL}_i, \mathsf{SC}_i, \mathsf{SR}^{\$}_i)\}_{k\in[\mathsf{K}]}$ and outputs $\{\widetilde{\mathsf{SL}}_{k,i}\}_{k\in[\mathsf{K}]}$.

5. For $i \in \{i_1, i_2\}$, using the values $\mathsf{SC}_i$ from the leakage, for each $k \in [\mathsf{K}]$, it computes $(\widetilde{\mathsf{SL}}_{k,i}, \widetilde{\mathsf{SC}}_{k,i}, \cdot) := f_{k,i}(\mathsf{SL}_i, \mathsf{SC}_i, \mathsf{SR}^{\$}_i)$.

6. For each $k \in [\mathsf{K}]$, it reconstructs $\widetilde{\mathsf{L}}_k$ using $\widetilde{\mathsf{SL}}_{k,i_1}, \ldots, \widetilde{\mathsf{SL}}_{k,i_{\mathsf{tmax}}}$.

7. For each $k \in [\mathsf{K}]$, it runs the tampering functions $g_k$ and $h_k$ exactly as in $\mathsf{Hyb}_2$ to get $\widetilde{\mathsf{C}}_k$ and $\widetilde{\mathsf{R}}_k$.

8. It outputs $\{\mathsf{Dec}(\widetilde{\mathsf{L}}_k, \widetilde{\mathsf{C}}_k, \widetilde{\mathsf{R}}_k)\}_{k\in[\mathsf{K}]}$.

Note that since $|\widetilde{\mathsf{SL}_{k,i}}| = m_1$, the leakage functions defined by the reduction belongs to $\mathcal{F}_{t,2,\mathsf{K}m_1}$. Note that if the leakage was with respect to the sharing of $\mathsf{C}_{\$}$ then the output of the reduction is identical to $\mathsf{Hyb}_{2.5}$ and otherwise, it is distributed identically to $\mathsf{Hyb}_3$. Thus, we break the leakage resilience property of $(\mathsf{LRShare}_{(3,n)}, \mathsf{LRRec}_{(3,n)})$. ■

$\underline{\mathsf{Hyb}_4}$ : In this hybrid, we make a syntactic change with respect to $\mathsf{Hyb}_3$. Instead of the tampering functions $g_k$, for each $k \in [\mathsf{K}]$, sampling $\mathsf{SC}_{i_3}, \ldots, \mathsf{SC}_{i_{\mathsf{tmax}}}$ again such that it satisfies the two consistency conditions, we let $g_k$ to use the same shares $\mathsf{SC}_{i_3}, \ldots, \mathsf{SC}_{i_{\mathsf{tmax}}}$ that were used to generate the shared randomness. This change is only syntactic and it can be easily seen that $\mathsf{Hyb}_3$ is identical to $\mathsf{Hyb}_4$.

$\underline{\mathsf{Hyb}_5}$ : In this hybrid, we make the following changes with respect to $\mathsf{Hyb}_4$. In constructing the shared randomness between $\{f_k, g_k, h_k\}_{k\in[\mathsf{K}]}$, we set $(\mathsf{SR}^{\$}_{i_1}, \ldots, \mathsf{SR}^{\$}_{i_{\mathsf{t}_{\max}}})$ as a valid secret sharing of the real $\mathsf{R}$ instead of fake $\mathsf{R}_{\$}$. Additionally, the tampering functions $f_k, g_k$ for each $k \in [\mathsf{K}]$, use these shares in order to extract $\widetilde{\mathsf{L}}_k, \widetilde{\mathsf{C}}_k$.

We now argue that $\mathsf{Hyb}_4 \approx_{\varepsilon_4} \mathsf{Hyb}_5$ using the leakage resilience property of $(\mathsf{LRShare}_{(2,n)}, \mathsf{LRRec}_{(2,n)})$.

**Claim 7.8** $\mathsf{Hyb}_4 \approx_{\varepsilon_4} \mathsf{Hyb}_5$.

**Proof** Assume for the sake of contradiction that the statistical distance between $\mathsf{Hyb}_4$ and $\mathsf{Hyb}_5$ is greater than $\varepsilon_4$. We will use this to break the leakage resilience property of $(\mathsf{LRShare}_{(2,n)}$, $\mathsf{LRRec}_{(2,n)})$. The reduction works as follows:

1. It generates $(\mathsf{L}, \mathsf{C}, \mathsf{R}) \leftarrow \mathsf{Enc}(s)$ and $(\mathsf{L}_{\$}, \mathsf{R}_{\$}, \mathsf{C}_{\$}) \leftarrow \mathsf{Enc}(s_{\$})$.
2. It shares $\mathsf{L}$ using $\mathsf{SecShare}_{(\mathcal{A},n)}$ and $\mathsf{C}$ using $\mathsf{LRShare}_{(3,n)}$ to get the shares $(\mathsf{SL}_1, \ldots, \mathsf{SL}_n)$ and $(\mathsf{SC}_1, \ldots, \mathsf{SC}_n)$ respectively.
3. It gives $\mathsf{R}$ and $\mathsf{R}_{\$}$ as the challenge messages to the leakage resilience challenger and defines the leakage functions as follows:
   - The function outputs $\mathsf{SR}_{i_3}$ in the clear.
   - For $i \in \{i_1, i_2\}$, the leakage function takes in $\mathsf{SR}_i$ as input and computes $\{(\widetilde{\mathsf{SL}}_{k,i}, \widetilde{\mathsf{SC}}_{k,i}, \cdot) := f_{k,i}(\mathsf{SL}_i, \mathsf{SC}_i, \mathsf{SR}_i)\}_{k\in[\mathsf{K}]}$. It then outputs $\{\widetilde{\mathsf{SL}}_{k,i}, \widetilde{\mathsf{SC}}_{k,i}\}_{k\in[\mathsf{K}]}$.
   - For each $i \in \{i_4, \ldots, i_{\mathsf{t}_{\max}}\}$, the leakage function takes in $\mathsf{SR}_i$ as input and computes $\{(\widetilde{\mathsf{SL}}_{k,i}, \cdot, \cdot) := f_{k,i}(\mathsf{SL}_i, \mathsf{SC}_i, \mathsf{SR}_i)\}_{k\in[\mathsf{K}]}$. It then outputs $\{\widetilde{\mathsf{SL}}_{k,i}\}_{k\in[\mathsf{K}]}$.
4. For $i = i_3$, using the value $\mathsf{SR}_{i_3}$ from the leakage, it computes $\{(\widetilde{\mathsf{SL}}_{k,i}, \widetilde{\mathsf{SC}}_{k,i}, \cdot) := f_{k,i}(\mathsf{SL}_i, \mathsf{SC}_i, \mathsf{SR}_i)\}_{k\in[\mathsf{K}]}$.
5. For each $k \in [\mathsf{K}]$, using the output of the leakage functions, the reduction reconstructs $\widetilde{\mathsf{L}}_k$ as $\mathsf{SecRec}_{(\mathcal{A},n)}(\widetilde{\mathsf{SL}}_{k,i_1}, \ldots, \widetilde{\mathsf{SL}}_{k,i_{\mathsf{t}_{\max}}})$ and $\widetilde{\mathsf{C}}_k$ as $\mathsf{LRRec}_{(3,n)}(\widetilde{\mathsf{SC}}_{k,i_1}, \widetilde{\mathsf{SC}}_{k,i_2}, \widetilde{\mathsf{SC}}_{k,i_3})$.
6. For each $k \in [\mathsf{K}]$, it runs the tampering function $h_k$ exactly as in $\mathsf{Hyb}_4$ to get $\widetilde{\mathsf{R}}_k$.

Notice that the leakage function defined by the above reduction belongs to the function family $\mathcal{F}_{t,1,\vec{\mu}}$ since the total amount of leakage is restricted to $O(\mathsf{K}m_2 + \mathsf{K}\mathsf{t}_{\max}m_1)$ bits. If the input to the leakage functions where the secret shares of $\mathsf{R}_{\$}$ then the distribution of the reduction's output is identical to $\mathsf{Hyb}_4$. Else, it is distributed identically to $\mathsf{Hyb}_5$. Thus, we break the leakage resilience property of $(\mathsf{LRShare}_{(2,n)}, \mathsf{LRRec}(2,n))$. ■

$\underline{\mathsf{Hyb}_6}$ : We again make a syntactic change with respect to $\mathsf{Hyb}_5$. In particular, we let the tampering function $h_k$, for each $k \in [\mathsf{K}]$, use the same shares $(\mathsf{SR}_{i_1}, \ldots, \mathsf{SR}_{i_{\mathsf{t}_{\max}}})$ that were used to generate the shared randomness. Again, $\mathsf{Hyb}_5$ is identical to $\mathsf{Hyb}_6$.

Notice that the distribution of the experiment's output in $\mathsf{Hyb}_6$ is identical to the value of the tampering experiment $\mathsf{Tamper}_s^{\overrightarrow{\mathsf{fun}},T}$. We know from the one-many split state security of the underlying non-malleable code that there exists a distribution $\mathcal{D}_{\{f_k,g_k,h_k\}_{k\in[\mathsf{K}]}}$ such that the output of $\mathsf{Hyb}_1$ is $\varepsilon_1$ close to $\text{copy}(\mathcal{D}_{\{f_k,g_k,h_k\}_{k\in[\mathsf{K}]}}, s)$. Thus, we infer that

$$\widetilde{\text{copy}}(\mathcal{D}_{\{f_k,g_k,h_k\}_{k\in[\mathsf{K}]}}, s) \approx_{\varepsilon_1+\varepsilon_3+\varepsilon_4} \mathsf{Tamper}_s^{\overrightarrow{\mathsf{fun}},T}$$

which completes the proof of non-malleability.

## 7.4 Rate Analysis

We now instantiate the primitives and provide the rate analysis.

1. We instantiate the three split state non-malleable code from the construction in Appendix B with rate $O(\frac{1}{\mathsf{K}})$. Using that construction, the $|\mathsf{L}| = |\mathsf{C}| = |\mathsf{R}| = O(\mathsf{K}m)$ bits and the error $\varepsilon_1 = 2^{-m^{\Omega(1)}}$. Further, from Theorem 3.22, there exists a constant $1 > \gamma' > 0$ such that the scheme only works for $m^{\gamma'} \geq \mathsf{K}^{(1-\gamma')}$. We will set $\gamma = (1-\gamma')/\gamma'$.

2. We use a secret sharing scheme for access structure $\mathcal{A}$ with rate $R$. We get $m_1 = O(\frac{\mathsf{K}m}{R})$.

3. We instantiate $\mathsf{LRShare}_{(3,n)}$ from Theorem 4.7 to get $m_2 = O(\mathsf{K}m_1\mathsf{t}_{\mathsf{max}}\log n) = O(\mathsf{K}^2m\mathsf{t}_{\mathsf{max}}\log n)$ by setting $\varepsilon_2$ to be $2^{-\Omega(m/\log m)}$.

4. Similarly, we instantiate $\mathsf{LRShare}_{(3,n)}$ from Theorem 4.7 to get $m_3 = O(\mathsf{K}^3m\mathsf{t}_{\mathsf{max}}\log^2 n)$ by setting $\varepsilon_3$ to be $2^{-\Omega(m/\log m)}$.

Thus the rate of our construction is $\Theta(\frac{\mathsf{R}}{\mathsf{K}^3\mathsf{t}_{\mathsf{max}}\log^2 n})$.

**Acknowledgements.** The first author's research supported in part by the IBM PhD Fellowship. The first author's research also supported in part from a DARPA /ARL SAFEWARE award, NSF Frontier Award 1413955, and NSF grant1619348, BSF grant 2012378, a Xerox Faculty Research Award, a Google Faculty Research Award, an equipment grant from Intel, an Okawa Foundation Research Grant, NSF-BSF grant 1619348, DARPA SafeWare subcontract to Galois Inc., DARPA SPAWAR contract N66001-15-1C-4065, US-Israel BSF grant 2012366, OKAWA Foundation Research Award, IBM Faculty Research Award, Xerox Faculty Research Award, B. John Garrick Foundation Award,Teradata Research Award, and Lockheed-Martin Corporation Research Award. This material is based upon work supported by the Defense Advanced Research Projects Agency through the ARL under Contract W911NF-15-C- 0205. The second author's research supported in part from DARPA/ARL SAFEWARE Award W911NF15C0210, AFOSR Award FA9550-15-1-0274, AFOSR YIP Award, DARPA and SPAWAR under contract N66001-15-C-4065, a Hellman Award and research grants by the Okawa Foundation, Visa Inc., and Center for Long-Term Cybersecurity (CLTC, UC Berkeley) of Sanjam Garg. The views expressed are those of the authors and do not reflect the official policy or position of the funding agencies.
The authors thank Pasin Manurangsi for pointing to the work of Alon et al. [AYZ95] for the explicit construction of perfect hash function family. The authors also thank Sanjam Garg, Peihan Miao and Prashant Vasudevan for useful comments on the write-up.

# References

[AAG$^+$16] Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In *TCC*, 2016.

[ADKO15] Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In *STOC*, pages 459–468, 2015.

[ADL14] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In *STOC*, pages 774–783, 2014.

[ADN$^+$18] Divesh Aggarwal, Ivan Damgard, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, Jo ao Ribeiro, and Mark Simkin. Stronger leakage-resilient and non-malleable secret-sharing schemes for general access structures. Cryptology ePrint Archive, Report 2018/1147, 2018. `https://eprint.iacr.org/2018/1147`.

[AGM$^+$15] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In *CRYPTO*, pages 538–557, 2015.

[AYZ95] Noga Alon, Raphael Yuster, and Uri Zwick. Color-coding. *J. ACM*, 42(4):844–856, 1995.

[BDG$^+$18] Marshall Ball, Dana Dachman-Soled, Siyao Guo, Tal Malkin, and Li-Yang Tan. Non-malleable codes for small-depth circuits. *To appear in FOCS*, 2018.

[BDIR18] Fabrice Benhamouda, Akshay Degwekar, Yuval Ishai, and Tal Rabin. On the local leakage resilience of linear secret sharing schemes. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 531–561. Springer, Heidelberg, August 2018.

[BDKM16] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes for bounded depth, bounded fan-in circuits. In *EUROCRYPT*, 2016.

[BDKM18] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes from average-case hardness: Ac0, decision trees, and streaming space-bounded tampering. In *EUROCRYPT*, 2018.

[BDL01] Dan Boneh, Richard A. DeMillo, and Richard J. Lipton. On the importance of eliminating errors in cryptographic computations. *Journal of Cryptology*, 14(2):101–119, 2001.

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

[Bla79] GR Blakley. Safeguarding cryptographic keys. In *Proc. AFIPS 1979 National Computer Conf.*, volume 48, pages 313–317, 1979.

[Bla99] Simon R Blackburn. Combinatorics and threshold cryptography. *CHAPMAN AND HALL CRC RESEARCH NOTES IN MATHEMATICS*, pages 49–70, 1999.

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

[CDF$^+$08] Ronald Cramer, Yevgeniy Dodis, Serge Fehr, Carles Padró, and Daniel Wichs. Detection of algebraic manipulation with applications to robust secret sharing and fuzzy extractors. In Nigel P. Smart, editor, *Advances in Cryptology – EUROCRYPT 2008*, volume 4965 of *Lecture Notes in Computer Science*, pages 471–488. Springer, Heidelberg, April 2008.

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

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

[CGM$^+$16] Nishanth Chandran, Vipul Goyal, Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay. Block-wise non-malleable codes. In *ICALP*, 2016.

[CGMA85] Benny Chor, Shafi Goldwasser, Silvio Micali, and Baruch Awerbuch. Verifiable secret sharing and achieving simultaneity in the presence of faults (extended abstract). In *26th Annual Symposium on Foundations of Computer Science*, pages 383–395. IEEE Computer Society Press, October 1985.

[CKR16] Nishanth Chandran, Bhavana Kanukurthi, and Srinivasan Raghuraman. Information-theoretic local non-malleable codes and their applications. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A: 13th Theory of Cryptography Conference, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 367–392. Springer, Heidelberg, January 2016.

[CL17] Eshan Chattopadhyay and Xin Li. Non-malleable codes and extractors for small-depth circuits, and affine functions. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *49th Annual ACM Symposium on Theory of Computing*, pages 1171–1184. ACM Press, June 2017.

[DDFY94] Alfredo De Santis, Yvo Desmedt, Yair Frankel, and Moti Yung. How to share a function securely. In *26th Annual ACM Symposium on Theory of Computing*, pages 522–533. ACM Press, May 1994.

[DF90] Yvo Desmedt and Yair Frankel. Threshold cryptosystems. In Gilles Brassard, editor, *Advances in Cryptology – CRYPTO'89*, volume 435 of *Lecture Notes in Computer Science*, pages 307–315. Springer, Heidelberg, August 1990.

[DKO13] Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In Ran Canetti and Juan A. Garay, editors, *Advances* *in Cryptology – CRYPTO 2013, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 239–257. Springer, Heidelberg, August 2013.

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

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

[FK84] Michael L. Fredman and János Komlós. On the size of separating systems and families of perfect hash functions. *SIAM Journal on Algebraic Discrete Methods*, 5(1):61–68, 1984.

[FMNV14] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In Yehuda Lindell, editor, *TCC*, volume 8349 of *Lecture Notes in Computer Science*, pages 465–488. Springer, 2014.

[FMNV15] Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. A tamper and leakage resilient von neumann architecture. In Jonathan Katz, editor, *PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography*, volume 9020 of *Lecture Notes in Computer Science*, pages 579–603. Springer, Heidelberg, March / April 2015.

[FMVW14] Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology – EUROCRYPT 2014*, volume 8441 of *Lecture Notes in Computer Science*, pages 111–128. Springer, Heidelberg, May 2014.

[Fra90] Yair Frankel. A practical protocol for large group oriented networks. In Jean-Jacques Quisquater and Joos Vandewalle, editors, *Advances in Cryptology – EUROCRYPT'89*, volume 434 of *Lecture Notes in Computer Science*, pages 56–61. Springer, Heidelberg, April 1990.

[FRR$^+$10] Sebastian Faust, Tal Rabin, Leonid Reyzin, Eran Tromer, and Vinod Vaikuntanathan. Protecting circuits from leakage: the computationally-bounded and noisy cases. In Henri Gilbert, editor, *Advances in Cryptology – EUROCRYPT 2010*, volume 6110 of *Lecture Notes in Computer Science*, pages 135–156. Springer, Heidelberg, May / June 2010.

[GK18a] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In *STOC*, pages 685–698, 2018.

[GK18b] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing for general access structures. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part I*, volume 10991 of *Lecture Notes in Computer Science*, pages 501–530. Springer, Heidelberg, August 2018.

[GKP$^+$18] Vipul Goyal, Ashutosh Kumar, Sunoo Park, Silas Richelson, and Akshayaram Srinivasan. Non-malleable commitments from non-malleable extractors. Manuscript, accessed via personal communication, 2018.

[GLM$^+$04] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (atp) security: Theoretical foundations for security against hardware tampering. In *Theory of Cryptography Conference*, pages 258–277. Springer, 2004.

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

[GMW17] Divya Gupta, Hemanta K. Maji, and Mingyuan Wang. Constant-rate non-malleable codes in the split-state model. Cryptology ePrint Archive, Report 2017/1048, 2017. `https://eprint.iacr.org/2017/1048`.

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

[ISW03] Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *Advances in Cryptology – CRYPTO 2003*, volume 2729 of *Lecture Notes in Computer Science*, pages 463–481. Springer, Heidelberg, August 2003.

[JKS93] Thomas Johansson, Gregory Kabatianskii, and Ben J. M. Smeets. On the relation between a-codes and codes correcting independent errors. In *EUROCRYPT*, pages 1–11, 1993.

[JW15] Zahra Jafargholi and Daniel Wichs. Tamper detection and continuous non-malleable codes. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015: 12th Theory of Cryptography Conference, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 451–480. Springer, Heidelberg, March 2015.

[KLT18] Aggelos Kiayias, Feng-Hao Liu, and Yiannis Tselekounis. Non-malleable codes for partial functions with manipulation detection. In Hovav Shacham and Alexandra Boldyreva, editors, *Advances in Cryptology – CRYPTO 2018, Part III*, volume 10993 of *Lecture Notes in Computer Science*, pages 577–607. Springer, Heidelberg, August 2018.

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

[KOS17] Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Four-state non-malleable codes with explicit constant rate. In Yael Kalai and Leonid Reyzin, editors, *TCC 2017: 15th Theory of Cryptography Conference, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 344–375. Springer, Heidelberg, November 2017.

[KOS18] Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Non-malleable randomness encoders and their applications. In *EUROCRYPT*, pages 589–617, 2018.

[KW93] Mauricio Karchmer and Avi Wigderson. On span programs. In *Proceedings of the Eigth Annual Structure in Complexity Theory Conference, San Diego, CA, USA, May 18-21, 1993*, pages 102–111, 1993.

[Li17] Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. *STOC*, 2017.

[LL12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 517–532. Springer, Heidelberg, August 2012.

[LV18] Tianren Liu and Vinod Vaikuntanathan. Breaking the circuit-size barrier in secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *50th Annual ACM Symposium on Theory of Computing*, pages 699–708. ACM Press, June 2018.

[NN93] Joseph Naor and Moni Naor. Small-bias probability spaces: Efficient constructions and applications. *SIAM J. Comput.*, 22(4):838–856, 1993.

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

[Rot12] Guy N. Rothblum. How to compute under $\mathcal{AC}^0$ leakage without secure hardware. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, volume 7417 of *Lecture Notes in Computer Science*, pages 552–569. Springer, Heidelberg, August 2012.

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

[SNW01] Rei Safavi-Naini and Huaxiong Wang. Robust additive secret sharing schemes over zm. In Kwok-Yan Lam, Igor Shparlinski, Huaxiong Wang, and Chaoping Xing, editors, *Cryptography and Computational Number Theory*, pages 357–368, Basel, 2001. Birkhäuser Basel.

[SS90] Jeanette P. Schmidt and Alan Siegel. The spatial complexity of oblivious k-probe hash functions. *SIAM J. Comput.*, 19(5):775–786, 1990.

# A 3-Split-State Non-Malleable Code

In this section, we recall the construction of 3-split-state non-malleable codes from the work of Kanukurthi et al. [KOS18] and Gupta et al. [GMW17] and show that it satisfies the additional property that given the second and third state, the message is statistically hidden. We first recall their encoding scheme. The encoding procedure uses the following tools (we don't specify the relations between the parameters and we refer the reader to [KOS18] for the details):

- $(\mathsf{Tag}, \mathsf{Verify})$ and $(\mathsf{Tag}', \mathsf{Verify}')$ be two message authentication codes. The key length, message length and the tag length of $(\mathsf{Tag}, \mathsf{Verify})$ and $(\mathsf{Tag}', \mathsf{Verify}')$ are $(\tau, \ell, \delta)$ and $(\tau_1, n, \delta_1)$ respectively.
- $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ be an average-case strong seeded extractor.
- $(\mathsf{Enc}^+, \mathsf{Dec}^+)$ be an augmented 2-split-state non-malleable code (see Definition 3.17).

We describe the encoding and decoding procedure in Figure 6.

```
• $\mathsf{Enc}(m)$: To encode a message $m$,
    1. Sample $s \leftarrow U_d$, $w \leftarrow U_n$, $k_1 \leftarrow U_{\tau_1}$.
    2. Compute $k_e \| k_2 = \mathrm{Ext}(w, s)$ and set $c := m \oplus k_e$.
    3. Compute $t_2 = \mathsf{Tag}_{k_2}(c)$ and $t_1 = \mathsf{Tag}'_{k_1}(w)$.
    4. Compute $(L, R) \leftarrow \mathsf{Enc}^+(k_1, t_1, s)$.
    5. Output the three states as $((w, L), R, c\|t_2)$.
• $\mathsf{Dec}(\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3)$: To decode a codeword $\mathsf{S}_1, \mathsf{S}_2, \mathsf{S}_3$:
    1. Parse $\mathsf{S}_1$ as $(w, L)$, $\mathsf{S}_2$ as $R$ and $\mathsf{S}_3$ as $c\|t_2$.
    2. Compute $(k_1, t_1, s) := \mathsf{Dec}^+(L, R)$. If $(k_1, t_1, s) = \perp$ then output $\perp$.
    3. Compute $k_e \| k_2 = \mathrm{Ext}(w, s)$.
    4. If, $\mathsf{Verify}_{k_1}(t_1, w) = 0$ or $\mathsf{Verify}'_{k_2}(t_2, c) = 0$, output $\perp$.
    5. Else, output $c \oplus k_e$.
```

**Figure 6**: Encoding and Decoding of 3-split-state Non-Malleable Code from the work of Kanukurthi et al. [KOS18]

We now argue that the second state $c\|t_2$ and the third state $R$ statistically hide the message $m$. It is known from [ADKO15] that any 2-split-state non-malleable code is a 2-out-of-2 secret sharing scheme. Hence, given $R$, the message $k_1, t_1, s$ is statistically hidden. This implies that both the source $w$ and the seed $s$ are statistically hidden given $R$. It now follows from the property of extractor Ext that $m \oplus k_e$ is statistically close to random and hence the message $m$ is statistically hidden. We formally argue this via a hybrid argument given below.

- $\mathsf{Hyb}_1$ : This corresponds to the distribution of $(R, c\|t_2)$ when a message $m$ is encoded.
- $\mathsf{Hyb}_2$ : This corresponds to a distribution of $(R, c\|t_2)$ where $c$ is generated honestly as given in Figure 6 but $R$ is generated as a right state that encodes $(k_1, t_1, \perp_d)$ where $\perp_d$ denotes a fixed string of length $d$. It follows from the fact that any 2-split-state non-malleable code is a 2-out-of-2 secret sharing scheme that $\mathsf{Hyb}_1$ is statistically close to $\mathsf{Hyb}_2$.

- $\mathsf{Hyb}_3$ : This corresponds to a distribution of $(R, c\|t_2)$ where $R$ is generated as in the previous hybrid but $c$ is chosen randomly from $U_\ell$. It follows from the property of the extractor Ext that $\mathsf{Hyb}_2$ and $\mathsf{Hyb}_3$ are statistically close. Notice that $\mathsf{Hyb}_3$ is independent of the message.

# B 3-split-state Non-malleable Code against Multiple Tampering

In this section, we give a construction of one-many 3-split-state Non-Malleable Code with tampering degree $k$ and rate $= O(1/k)$. This is obtained by replacing the one-one augmented non-malleable code in the construction of Kanukurthi et al. [KOS18] and Gupta et al. [GMW17] with an one-many augmented non-malleable code of [CGL16, GKP+18]. We detail the construction and the proof of security below.

**Building Blocks.** We use the following building blocks:

- $(\mathsf{Tag}, \mathsf{Verify})$ and $(\mathsf{Tag}', \mathsf{Verify}')$ be two message authentication codes. The key length, message length and the tag length of $(\mathsf{Tag}, \mathsf{Verify})$ and $(\mathsf{Tag}', \mathsf{Verify}')$ are $(\tau', \ell, \delta')$ and $(\tau, n, \delta)$ respectively. The MACs $(\mathsf{Tag}, \mathsf{Verify})$ and $(\mathsf{Tag}', \mathsf{Verify}')$ are unforgeable except with probability $\varepsilon_1$ and $\varepsilon'_1$ respectively.
- $\mathrm{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^{\ell+\tau}$ be a strong average-case seeded extractor for average-case min-entropy $t + \log(1/\mu)$ and error $\varepsilon_2 + \mu$.
- $(\mathsf{Enc}^+, \mathsf{Dec}^+)$ be an augmented, one-many 2-split-state non-malleable code with tampering degree $k$ and simulation error $\varepsilon_3$.
- We will set $n - (\ell + \tau + 3)k \geq t + \log(1/\mu)$.

**Construction.** The construction is exactly same as Figure 6 except that we use an augmented, one-many 2-split state non-malleable code.

**Proof of Non-Malleability.** The correctness of the construction is easy to verify and we also note that via the same argument given in Appendix A, it can be shown that given the second and the third state of the codeword, the message is statistically hidden. We now give the proof of one-many non-malleability. For every $i \in [k]$, let $(f_i, g_i, h_i)$ be a function belonging to the 3-split state tampering family. We give the description of the simulator $D_{(f_1,g_1,h_1),\ldots,(f_k,g_k,h_k)}$ in Figure 7 and this uses the simulator $\mathcal{S}_{(f'_1,g'_1),\ldots,(f'_k,g'_k)}$ (for some split state functions $(f'_i, g'_i)$ for every $i \in [k]$) of the underlying 2-split state non-malleable code.

Recall that $\mathsf{Tamper}^m_{(f_1,g_1,h_1),\ldots,(f_k,g_k,h_k)}$ be the outcome of the experiment wherein the message $m$ is encoded, the codeword is tampered with the functions $(f_i, g_i, h_i)$ for every $i \in [k]$ and the output corresponds decoding of the tampered codewords. For convenience, we give the description of this experiment in Figure 8. In order to prove non-malleability, we need to show that

$$\mathsf{Tamper}^m_{(f_1,g_1,h_1),\ldots,(f_k,g_k,h_k)} \approx \widetilde{\mathrm{copy}}(D_{(f_1,g_1,h_1),\ldots,(f_k,g_k,h_k)}, m)$$

We show this through a sequence of hybrids.

$\underline{\mathsf{Hyb}_1}$ : In this hybrid, we change the function $\mathsf{Tamper}'_{(f_1,g_1),\ldots,(f_k,g_k)}$ in Figure 8 as follows.

```
The simulator $D_{(f_1,g_1,h_1),\ldots,(f_k,g_k,h_k)}$ works as follows:
  1. Sample $k_e \| k_2 \leftarrow U_{\ell+\tau}$.
  2. $(\widetilde{\beta}_1, \ldots, \widetilde{\beta}_k) \leftarrow \mathcal{S}'$ where $\mathcal{S}'$ is described below.
  3. Set $c = k_e \oplus 0$ and compute $t_2 = \mathsf{Tag}_{k_2}(c)$.
  4. For each $i \in [k]$,
     (a) Define $(\widetilde{c}^i, \widetilde{t}^i) := h_i(c \| t_2)$.
     (b) If $\widetilde{\beta}_i = \mathsf{same}^\star$ then:
         i. If $\widetilde{c}^i = c$, set $\widetilde{\gamma}_i = \mathsf{same}^\star$. Else, set $\widetilde{\gamma}_i = \perp$.
     (c) Else, parse $\widetilde{\beta}_i$ as $\widetilde{k}_e^i, \widetilde{k}_2^i$.
     (d) If $\mathsf{Verify}_{\widetilde{k}_2^i}(\widetilde{c}^i, \widetilde{t}_2^i) = 1$, set $\widetilde{\gamma}_i = \widetilde{c}^i \oplus \widetilde{k}_e^i$. Else, set $\widetilde{\gamma}_i = \perp$.
  5. Output $(\widetilde{\gamma}_1, \ldots, \widetilde{\gamma}_k)$.
```

```
The function $\mathcal{S}'$
  1. Sample $w \leftarrow U_n$.
  2. $(\mathsf{L}, \widetilde{\alpha_1}, \ldots, \widetilde{\alpha_k}) \leftarrow \mathcal{S}_{(f_1^{(1)}[w],g_1),\ldots,(f_k^{(1)}[w],g_k)}$ where $\widetilde{\alpha}_i := (\widetilde{k}_1^i, \widetilde{t}_1^i, \widetilde{s}^i)$ or $\widetilde{\alpha}_i = \mathsf{same}^\star$ and $f_i^{(1)}[w]$
     is equal to $f_i(w, \cdot)$ except that it only outputs the last $|\mathsf{L}|$ bits of the output of $f_i$.
  3. For each $i \in [k]$, define $\widetilde{w}^i = f_i^{(2)}[\mathsf{L}](w)$ where $f_i^{(2)}[\mathsf{L}]$ is equal to $f_i(\cdot, \mathsf{L})$ except that it only
     outputs the first $|w|$ bits of the output of $f_i$.
  4. For each $i \in [k]$,
     (a) If $\widetilde{\alpha}_i = \mathsf{same}^\star$
         i. If $\widetilde{w}^i = w$: set $\widetilde{\beta}_i = \mathsf{same}^\star$.
         ii. Else, set $\widetilde{\beta}_i = \perp$.
     (b) Else if, $\mathsf{Verify}'(\widetilde{t}_1^i, \widetilde{w}^i) = 1$, set $\beta_i = \mathrm{Ext}(\widetilde{w}^i, \widetilde{s}^i)$. Else, set $\widetilde{\beta}_i = \perp$.
  5. Output $(\widetilde{\beta}_1, \ldots, \widetilde{\beta}_k)$.
```

**Figure 7**: Description of the simulator $D_{(f_1,g_1,h_1),\ldots,(f_k,g_k,h_k)}$

1. Compute $t_1 = \mathsf{Tag}'_{k_1}(w)$.

1. Sample $s \leftarrow U_d$, $w \leftarrow U_n$, $k_1 \leftarrow U_{\tau_1}$.
2. Sample $(k_e \| k_2, \widetilde{\beta}_1, \ldots, \widetilde{\beta}_k) \leftarrow \mathsf{Tamper'}_{(f_1,g_1),\ldots,(f_k,g_k)}(s, w, k_1)$ where $\mathsf{Tamper'}_{(f_1,g_1),\ldots,(f_k,g_k)}$ is described below.
3. Set $c = k_e \oplus m$ and compute $t_2 = \mathsf{Tag}_{k_2}(c)$.
4. For each $i \in [k]$,
   (a) Define $(\widetilde{c}^i, \widetilde{t}^i) := h_i(c \| t_2)$.
   (b) Parse $\widetilde{\beta}_i$ as $\widetilde{k}_e^i, \widetilde{k}_2^i$.
   (c) If $\mathsf{Verify}_{\widetilde{k}_2^i}(\widetilde{c}^i, \widetilde{t}_2^i) = 1$, set $\widetilde{\gamma}_i = \widetilde{c}^i \oplus \widetilde{k}_e^i$. Else, set $\widetilde{\gamma}_i = \perp$.
5. Output $(\widetilde{\gamma}_1, \ldots, \widetilde{\gamma}_k)$.

$\mathsf{Tamper'}_{(f_1,g_1),\ldots,(f_k,g_k)}$

On input $s, w, k_1$ do:

1. Compute $t_1 = \mathsf{Tag}'_{k_1}(w)$.
2. $(\mathsf{L}, \widetilde{\alpha_1}, \ldots, \widetilde{\alpha_k}) \leftarrow \overline{\mathsf{Tamper}}^{k_1 \| t_1 \| s}_{(f_1^{(1)}[w],g_1),\ldots,(f_k^{(1)}[w],g_k)}$ where $\widetilde{\alpha}_i := (\widetilde{k}_1^i, \widetilde{t}_1^i, \widetilde{s}^i)$ and $f_i^{(1)}[w]$ is equal to $f_i(w, \cdot)$ except that it only outputs the last $|\mathsf{L}|$ bits of the output of $f_i$ and $\overline{\mathsf{Tamper}}$ denotes the tampering experiment for the underlying 2-split-state non-malleable code.
3. For each $i \in [k]$, define $\widetilde{w}^i = f_i^{(2)}[\mathsf{L}](w)$ where $f_i^{(2)}[\mathsf{L}]$ is equal to $f_i(\cdot, \mathsf{L})$ except that it only outputs the first $|w|$ bits of the output of $f_i$..
4. Set $k_e \| k_2 := \mathrm{Ext}(w, s)$.
5. For each $i \in [k]$,
   (a) If, $\mathsf{Verify'}(\widetilde{t}_1^i, \widetilde{w}^i) = 1$, set $\beta_i = \mathrm{Ext}(\widetilde{w}^i, \widetilde{s}^i)$. Else, set $\widetilde{\beta}_i = \perp$.
6. Output $(k_e \| k_2, \widetilde{\beta}_1, \ldots, \widetilde{\beta}_k)$.

**Figure 8**: Description of the tampering Experiment $\mathsf{Tamper}^m_{(f_1,g_1,h_1),\ldots,(f_k,g_k,h_k)}$

2. $(\mathsf{L}, \widetilde{\alpha_1}, \ldots, \widetilde{\alpha_k}) \leftarrow \mathcal{S}_{(f_1^{(1)}[w],g_1),\ldots,(f_k^{(1)}[w],g_k)}$ where $\widetilde{\alpha}_i := (\widetilde{k}_1^i, \widetilde{t}_1^i, \widetilde{s}^i)$ and $f_i^{(1)}[w]$ is equal to $f_i(w, \cdot)$ except that it only outputs the last $|\mathsf{L}|$ bits of the output of $f_i$ and $\mathcal{S}$ denotes the simulated distribution for the underlying augmented, 2-split-state non-malleable code.

3. For each $i \in [k]$, if $\widetilde{\alpha}_i = \mathsf{same}^\star$, reset $\widetilde{\alpha}_i = k_1 \| t_1 \| s$.

4. The rest of the steps are same as in Figure 8.

Note that the only change between $\mathsf{Tamper}^m_{(f_1,g_1,h_1),\ldots,(f_k,g_k,h_k)}$ and $\mathsf{Hyb}_1$ is that we use the simulator of the underlying augmented 2-split-state non-malleable code. It now follows directly from the security of the non-malleable code that $\mathsf{Hyb}_1$ is $\varepsilon_3$ close to $\mathsf{Tamper}^m_{(f_1,g_1,h_1),\ldots,(f_k,g_k,h_k)}$.

$\underline{\mathsf{Hyb}_2}$ : In thus hybrid, we make some changes to the function $\mathsf{Tamper'}_{(f_1,g_1),\ldots,(f_k,g_k)}$ from the previous hybrid. The new function $\mathsf{Tamper'}^{(2)}_{(f_1,g_1),\ldots,(f_k,g_k)}$ takes as input $s, w$ and is defined as follows:

1. $(\mathsf{L}, \widetilde{\alpha_1}, \ldots, \widetilde{\alpha_k}) \leftarrow \mathcal{S}_{(f_1^{(1)}[w],g_1),\ldots,(f_k^{(1)}[w],g_k)}$ where $\widetilde{\alpha}_i := (\widetilde{k_1^i}, \widetilde{t_1^i}, \widetilde{s^i})$ and $f_i^{(1)}[w]$ is equal to $f_i(w,\cdot)$ except that it only outputs the last $|\mathsf{L}|$ bits of the output of $f_i$ and $\mathcal{S}$ denotes the simulated distribution for the underlying 2-split-state non-malleable code.

2. For each $i \in [k]$, define $\widetilde{w}^i = f_i^{(2)}[\mathsf{L}](w)$ where $f_i^{(2)}[\mathsf{L}]$ is equal to $f_i(\cdot, \mathsf{L})$ except that it only outputs the first $|w|$ bits of the output of $f_i$.

3. Set $k_e \| k_2 := \mathrm{Ext}(w, s)$.

4. For each $i \in [k]$,

   (a) If $\widetilde{\alpha}_i = \mathsf{same}^\star$

      i. If $\widetilde{w}^i = w$: set $\widetilde{\beta}_i = k_e \| k_2$.

      ii. Else, set $\widetilde{\beta}_i = \perp$.

   (b) Else If, $\mathsf{Verify'}(\widetilde{t_1^i}, \widetilde{w}^i) = 1$, set $\beta_i = \mathrm{Ext}(\widetilde{w}^i, \widetilde{s}^i)$. Else, set $\widetilde{\beta}_i = \perp$.

5. Output $(k_e \| k_2, \widetilde{\beta}_1, \ldots, \widetilde{\beta}_k)$.

Note that the only change between $\mathsf{Hyb}_1$ and $\mathsf{Hyb}_2$ is that when $\widetilde{\alpha}_i = \mathsf{same}^\star$, we will check if $\widetilde{w}^i = w$ instead of running the MAC verification. It now follows from the security of MAC unforgeability that $\mathsf{Hyb}_1 \approx_{k\varepsilon'_1} \mathsf{Hyb}_2$. The formal reduction follows directly from Claim 2 in [KOS18].

$\underline{\mathsf{Hyb}_3}$ : In this hybrid, we make some changes to the function $\mathsf{Tamper'}^{(2)}_{(f_1,g_1),\ldots,(f_k,g_k)}$. The new function $\mathsf{Tamper'}^{(3)}_{(f_1,g_1),\ldots,(f_k,g_k)}$ takes $w$ as input and is defined as follows:

1. $(\mathsf{L}, \widetilde{\alpha_1}, \ldots, \widetilde{\alpha_k}) \leftarrow \mathcal{S}_{(f_1^{(1)}[w],g_1),\ldots,(f_k^{(1)}[w],g_k)}$ where $\widetilde{\alpha}_i := (\widetilde{k_1^i}, \widetilde{t_1^i}, \widetilde{s^i})$ and $f_i[w]$ is same as $f_i^{(1)}$ with $w$ being hardwired that outputs the last $|\mathsf{L}|$ bits of $f_i$ and $\mathcal{S}$ denotes the simulated distribution for the underlying 2-split-state non-malleable code.

2. For each $i \in [k]$, define $\widetilde{w}^i = f_i^{(2)}[\mathsf{L}](w)$ where $f_i^{(2)}[\mathsf{L}]$ is same as $f_i$ except that it has $\mathsf{L}$ hardwired and outputs the last $|w|$ bits of $f_i$.

3. Sample $k_e \| k_2 \leftarrow U_{\ell+\tau}$.

4. For each $i \in [k]$,

   (a) If $\widetilde{\alpha}_i = \mathsf{same}^\star$
      i. If $\widetilde{w}^i = w$: set $\widetilde{\beta}_i = k_e \| k_2$.
      ii. Else, set $\widetilde{\beta}_i = \perp$.
   (b) Else If, $\mathsf{Verify}'(\widetilde{t}^i_1, \widetilde{w}^i) = 1$, set $\beta_i = \mathrm{Ext}(\widetilde{w}^i, \widetilde{s}^i)$. Else, set $\widetilde{\beta}_i = \perp$.

5. Output $(k_e \| k_2, \widetilde{\beta}_1, \ldots, \widetilde{\beta}_k)$.

We now show that $\mathsf{Hyb}_2$ is statistically close to $\mathsf{Hyb}_3$ by a straightforward generalization of Claim 3 from [KOS18].

**Claim B.1** $\mathsf{Hyb}_2 \approx_{\varepsilon_2} \mathsf{Hyb}_3$

**Proof** Most parts of this proof is taken verbatim from [KOS18] and we only make relevant changes so that it works in the many tampering setting. We start by defining a few random variables that capture the auxiliary information. We will then invoke the extractor security to show that $\mathsf{Hyb}_2$ is statistically close to $\mathsf{Hyb}_3$.

We fix the output $(\mathsf{L}, \widetilde{\alpha_1}, \ldots, \widetilde{\alpha_k}) \leftarrow \mathcal{S}_{(f_1^{(1)}[w], g_1), \ldots, (f_k^{(1)}[w], g_k)}$ and define

$$b^i_{\mathsf{same}^\star} := \begin{cases} 1 & \text{if } \widetilde{\alpha}_i = \mathsf{same}^\star \\ 0 & \text{otherwise} \end{cases}$$

$$b^i_\perp := \begin{cases} 1 & \text{if } \widetilde{\alpha}_i = \perp \\ 0 & \text{otherwise} \end{cases}$$

$$eq^i(w) := \begin{cases} 1 & \text{if } f_i^{(2)}[\mathsf{L}](w) = w \\ 0 & \text{otherwise} \end{cases}$$

$$V^i(w) = \mathsf{Verify}'_{\widetilde{k}^i_1}(f_i^{(2)}[\mathsf{L}](w), \widetilde{t}^i_1)$$

Finally, we define:

$$Y^i(w, b_1, b_2) := \begin{cases} eq^i(w) & \text{if } b_1 = 1 \\ (V^i(w), \mathrm{Ext}(\widetilde{w}^i, \widetilde{s}^i)) & \text{if } b_1 = 0 \wedge b_2 = 0 \\ \perp & \text{otherwise} \end{cases}$$

We define the auxiliary information by $\widehat{E} = (b^i_{\mathsf{same}^\star}, b^i_\perp, Y^i(w, b^i_{\mathsf{same}^\star}, b^i_\perp))_{i \in [k]}$. We now define the new function $G$ that takes $\widehat{E}$ and a string $k \in \{0,1\}^{\ell+\tau}$ and works as follows:

- Parse $\widehat{E}$ as $(b^i_{\mathsf{same}^\star}, b^i_\perp, y^i)_{i \in [k]}$.
- For each $i \in [k]$
  1. If $b^i_{\mathsf{same}^\star} = 1$:
     (a) If $y^i = 1$, output $(k, k)$.
     (b) Else, output $(k, \perp)$.
  2. Else,

(a) If $b_\perp = 1$, output $(k, \perp)$.
(b) Else, parse $y^i$ as $V^i(w), \mathrm{Ext}(\widetilde{w}^i, \widetilde{s}^i)$.
  i. If $V^i(w) = 1$, output $(k, \mathrm{Ext}(\widetilde{w}^i, \widetilde{s}^i)$.
  ii. Else, output $(k, \perp)$

Note that $G$ on input $\widehat{E}, Ext(W; S)$ is distributed identically to $\mathsf{Tamper}'^{(2)}_{(f_1,g_1),\ldots,(f_k,g_k)}$ on input $S, W$ and $G$ on input $\widehat{E}, U_{\ell+\tau}$ is distributed identically to $\mathsf{Tamper}'^{(3)}_{(f_1,g_1),\ldots,(f_k,g_k)}$ on input $W$. Thus, to prove that $\mathsf{Hyb}_2 \approx_{\varepsilon_2} \mathsf{Hyb}_3$ it is sufficient to prove that

$$\widehat{E}, Ext(W; S) \approx_{\varepsilon_2} \widehat{E}, U_{\ell+\tau}$$

Notice that $\widehat{E}$ depends only on the string $W$ is is independent of the seed $S$. Further, $\widehat{E}$ takes at most $2^{(3+\ell+\tau)k}$ values. Hence, $\widetilde{H}_\infty(W|\widehat{E}) \geq n - (3 + \ell + \tau)k$. By our choice of parameters, it follows from the average-case strong extractor property of Ext that $\widehat{E}, Ext(W; S) \approx_{\varepsilon_2} \widehat{E}, U_{\ell+\tau}$. ■

$\underline{\mathsf{Hyb}_4}$ : In this hybrid, we make a syntactic change with respect to $\mathsf{Hyb}_3$. The formal description of $\mathsf{Hyb}_4$ is given in Figure 9. Notice that the change is only syntactic and $\mathsf{Hyb}_3$ is identical to $\mathsf{Hyb}_4$.

$\underline{\mathsf{Hyb}_5}$ : In this hybrid, we make the following changes with respect to $\mathsf{Hyb}_4$.

1. Sample $k_e \| k_2 \leftarrow U_{\ell+\tau}$.
2. Sample $(\widetilde{\beta}_1, \ldots, \widetilde{\beta}_k) \leftarrow \mathsf{Tamper}'^{(4)}_{(f_1,g_1),\ldots,(f_k,g_k)}$.
3. Set $c = k_e \oplus m$ and compute $t_2 = \mathsf{Tag}_{k_2}(c)$.
4. For each $i \in [k]$,
   (a) Define $(\widetilde{c}^i, \widetilde{t}^i) := h_i(c \| t_2)$.
   (b) If $\widetilde{\beta}_i = \mathsf{same}^\star$
      i. If $\widetilde{c}^i = c$, set $\widetilde{\gamma}_i = m$.
      ii. Else, output $\perp$.
   (c) Else if, parse $\widetilde{\beta}_i$ as $\widetilde{k}^i_e, \widetilde{k}^i_2$. If $\mathsf{Verify}_{\widetilde{k}^i_2}(\widetilde{c}^i, \widetilde{t}^i_2) = 1$, set $\widetilde{\gamma}_i = \widetilde{c}^i \oplus \widetilde{k}^i_e$. Else, set $\widetilde{\gamma}_i = \perp$.
5. Output $(\widetilde{\gamma}_1, \ldots, \widetilde{\gamma}_k)$.

The statistical closeness between $\mathsf{Hyb}_4$ and $\mathsf{Hyb}_5$ follows from the unforgeability of $(\mathsf{Tag}, \mathsf{Verify})$ and follows identically to the proof of closeness between $\mathsf{Hyb}_1$ and $\mathsf{Hyb}_2$. We infer that $\mathsf{Hyb}_4 \approx_{k\varepsilon_1} \mathsf{Hyb}_5$.

$\underline{\mathsf{Hyb}_6}$ : In this hybrid, we replace $c = k_e \oplus m$ with $c = k_e \oplus 0$. We infer from the security of one-time pad that $\mathsf{Hyb}_5$ is identically distributed to $\mathsf{Hyb}_6$. Note that $\mathsf{Hyb}_6$ is distributed identically to $\widetilde{\mathrm{copy}}(D_{(f_1,g_1,h_1),\ldots,(f_k,g_k,h_k)}, m)$.

```
1. Sample $k_e \| k_2 \leftarrow U_{\ell+\tau}$.
2. Sample $(\widetilde{\beta}_1, \ldots, \widetilde{\beta}_k) \leftarrow \mathsf{Tamper'}^{(4)}_{(f_1,g_1),\ldots,(f_k,g_k)}$ where $\mathsf{Tamper'}^{(4)}_{(f_1,g_1),\ldots,(f_k,g_k)}$ is described
   below.
3. For each $i \in [k]$, if $\widetilde{\beta}_i = \mathsf{same}^\star$ then reset $\widetilde{\beta}_i = k_e \| k_2$.
4. Set $c = k_e \oplus m$ and compute $t_2 = \mathsf{Tag}_{k_2}(c)$.
5. For each $i \in [k]$,
   (a) Define $(\widetilde{c}^i, \widetilde{t}^i) := h_i(c \| t_2)$.
   (b) Parse $\widetilde{\beta}_i$ as $\widetilde{k}_e^i, \widetilde{k}_2^i$.
   (c) If $\mathsf{Verify}_{\widetilde{k}_2^i}(\widetilde{c}^i, \widetilde{t}_2^i) = 1$, set $\widetilde{\gamma}_i = \widetilde{c}^i \oplus \widetilde{k}_e^i$. Else, set $\widetilde{\gamma}_i = \perp$.
6. Output $(\widetilde{\gamma}_1, \ldots, \widetilde{\gamma}_k)$.
```

$$\mathsf{Tamper'}^{(4)}_{(f_1,g_1),\ldots,(f_k,g_k)}$$

```
1. Sample $w \leftarrow U_n$.
2. $(\mathsf{L}, \widetilde{\alpha_1}, \ldots, \widetilde{\alpha_k}) \leftarrow \mathcal{S}_{(f_1^{(1)}[w],g_1),\ldots,(f_k^{(1)}[w],g_k)}$ where $\widetilde{\alpha}_i := (\widetilde{k}_1^i, \widetilde{t}_1^i, \widetilde{s}^i)$ and $f_i[w]$ is same as $f_i^{(1)}$
   with $w$ being hardwired that outputs the last $|\mathsf{L}|$ bits of $f_i$ and $\mathcal{S}$ denotes the simulated
   distribution for the underlying 2-split-state non-malleable code.
3. For each $i \in [k]$, define $\widetilde{w}^i = f_i^{(2)}[\mathsf{L}](w)$ where $f_i^{(2)}[\mathsf{L}]$ is same as $f_i$ except that it has $\mathsf{L}$
   hardwired and outputs the last $|w|$ bits of $f_i$.
4. For each $i \in [k]$,
   (a) If $\widetilde{\alpha}_i = \mathsf{same}^\star$
       i. If $\widetilde{w}^i = w$: set $\widetilde{\beta}_i = \mathsf{same}^\star$.
       ii. Else, set $\widetilde{\beta}_i = \perp$.
   (b) Else If, $\mathsf{Verify'}(\widetilde{t}_1^i, \widetilde{w}^i) = 1$, set $\beta_i = \mathrm{Ext}(\widetilde{w}^i, \widetilde{s}^i)$. Else, set $\widetilde{\beta}_i = \perp$.
5. Output $(\widetilde{\beta}_1, \ldots, \widetilde{\beta}_k)$.
```

**Figure 9**: Description of $\mathsf{Hyb}_4$

## B.1 Instantiation

We now instantiate the building blocks and give bounds on the rate of our construction.

1. We set the error rate $\mu = \varepsilon_1 = \varepsilon_1' = \varepsilon_2 = \varepsilon_3 = 2^{-\lambda}$.
2. We will instantiate the MAC from the work of Johansson et al. [JKS93] (see also [GMW17] for a construction). For authenticating $n$ bit strings, the tag length and the key length are respectively $(\log n + \lambda)$ and $2(\log n + \lambda)$.
3. We will instantiate the strong average case extractor from the work of Guruswami et al. [GUV09] (see Theorem 3.8). Fixing the output length of the extractor to be $\ell + \tau$ and the average min-entropy $t < n - k(3 + \tau + \ell) - \log(1/\mu)$, we get the length $n = O(k(\lambda + \ell))$.
4. We instantiate the underlying non-malleable code from [GKP+18]. Let $2\beta$ be the length of the codeword. The length of the message that is encoded is $O(\log \ell + \lambda)$. By Theorem 3.21, we get that there exists a constant $\delta > 0$ such that $2\beta = O(\lambda^{1+\delta} + \log^{1+\delta} \ell)$. For some constant $\gamma > 0$ and $\gamma < \delta$, we can set the tampering degree to be $\lambda^{\gamma}$.

We now calculate the rate $R$.

$$\begin{aligned} R &= \frac{\ell}{2\beta + n + \ell + \log \ell + \lambda} \\ &= \frac{\ell}{O(k\ell + \lambda^{1+\delta})} \end{aligned}$$

We set $\lambda^{1+\delta} = O(\ell)$ and we get the rate to be $\Theta(\frac{1}{k})$. The error of our scheme is $2^{-O(\lambda)} = 2^{-\ell^{\Omega(1)}}$.