Survey: Non-malleable code in the split-state model

Divesh Aggarwal* Marshall Ball† Maciej Obremski‡

October 6, 2022

Abstract

Non-malleable codes are a natural relaxation of error correction and error detection codes applicable in scenarios where error-correction or error-detection is impossible.

Over the last decade, non-malleable codes have been studied for a wide variety of tampering families. Among the most well studied of these is the split-state family of tampering channels, where the codeword is split into two or more parts and each part is tampered independently.

We survey various constructions and applications of non-malleable codes in the split-state model.

*Department of Computer Science and Center for Quantum Technologies, National University of Singapore. Email: dcsdiva@nus.edu.sg.

†Courant Institute of Mathematics, New York University. Email: marshall@cs.nyu.edu

‡Center for Quantum Technologies, National University of Singapore. Email: obremski.math@gmail.com.

1 Introduction

Motivated by applications in tamper-resilient hardware, Dziembowski, Pietrzak, and Wichs [DPW10] introduced non-malleable codes as a natural generalization of error correction and error detection codes.

The error correction and the error detection codes are the most basic objects in the codes theory. They do, however, have significant drawbacks, which makes them unsuitable for the applications to tamper-resilient cryptography. In the case of error correction codes, the message can be retrieved as long as only a limited number of positions of the codeword have been flipped. However it is hard to find a scenario where an adversary would limit himself to flipping only a few positions when given access to the whole codeword. The error detection codes face a different interesting challenge, namely whatever tampering limitations we impose on the adversary (be it polynomial time, bounded memory or some structure limitations like split-state), the adversary can not be allowed to overwrite the codeword. Overwriting a codeword with another valid pre-computed codeword makes the detection of tampering clearly impossible. But overwrites are quite simple attacks, the adversary wipes the memory of the device and uploads some new data. While this attack seems irrational, there are scenarios when partial overwrites are realistic attacks on the scheme[1]. Naturally, we would like to allow for a wide spectrum of attacks including overwrite attacks.

Motivated by this, Dziembowski, Pietrzak and Wichs [DPW10] considered a notion of non-malleable codes (NMC). It was a weakening of detection/correction codes based on the concept of *non-malleability* introduced by [DDN00].

The model is very natural and clean. We start with the message m, we encode it $\mathsf{Enc}(m) = c$, then we store the encoding on the device, the adversary picks any adversarial function $t \in \mathcal{T}$ (where $\mathcal{T}$ is a class of tampering channels), a codeword is tampered to $c' = t(c)$, and after decoding we get $Dec(t(c)) = m'$. In the *error-correction* codes, we would like $m' = m$, in the *error-detection* codes we would like $m' = m$ or $m' = \perp$ (where $\perp$ is a special symbol denoting detection of tampering). As we already discussed, if the family of channels $\mathcal{T}$ contains constant functions then neither correction nor detection is possible. There is however a meaningful definition that can be considered here. Non-malleable code against the family of channels $\mathcal{T}$ guarantees that after above tampering, $m' = m$ or m' is completely independent of m, for instance, $m' = m + 1$ is not possible.

Dziembowski, Pietrzak, and Wichs formalized this notion using the simulation paradigm: the output of the experiment can be simulated by a simulator that depends only on the adversarial channel t (and not the message m), and is allowed to output a special symbol same which is replaced by the encoded message m.

Definition 1.1 (Non-malleable codes [DPW10]). *A pair of (randomized[2]) algorithms,* $(\mathsf{Enc} : \{0,1\}^k \to \{0,1\}^n, \mathsf{Dec} : \{0,1\}^n \to \{0,1\}^k)$*, is an* ε*-non-malleable code with respect to a family of channels* $\mathcal{T} \subseteq \{f : \{0,1\}^n \to \{0,1\}^n\}$*, if the following properties hold:*

1. *(Correctness)*

   For every message $m \in \{0,1\}^k$*,*

   $$\Pr[\mathsf{Dec}(\mathsf{Enc}(m) = m] = 1$$

   where the probability is over the randomness of the encoding and decoding procedures.

2. *(Security)*

   For every $t \in \mathcal{T}$ *there is a random variable* D_t *supported on* $\{0,1\}^k \cup \{\mathsf{same}\}$ *that is independent of the randomness in* $\mathsf{Enc}, \mathsf{Dec}$*, such that for every message* $m \in \{0,1\}^k$

   $$(\mathsf{Dec}(t(\mathsf{Enc}(m))) \approx_\varepsilon \mathsf{Copy}(D_t, m)),$$

[1] Those attacks often allow the adversary to gradually learn the underlying secret key. They are especially prevalent in the natural scenarios where the adversary gets to tamper the device multiple times.

[2] [DPW10] defined Dec to be deterministic, however here we allow decoding to be randomized. It is not clear if randomized and deterministic decoding are equivalent; in particular, no strong separations are known.

where $\approx_\varepsilon$ denotes statistical distance (total variation distance) at most ε and the function Copy *is defined as*

$$\mathsf{Copy}(x, y) = \begin{cases} x & \text{if } x \neq \mathit{same} \\ y & \text{if } x = \mathit{same}. \end{cases}$$

A few years later, Aggarwal, Dodis, Kazana, and Obremski [ADKO15a] introduced an alternative perspective on non-malleable codes by introducing the notion of non-malleable reductions. To intuitively describe a non-malleable reduction, imagine the scenario discussed earlier, where the message m is encoded as a codeword c, and c is tampered using $t \in \mathcal{T}$ into $c' = t(c)$. The tampered codeword c' decodes to m'. A non-malleable reduction from $\mathcal{T}$ to $\mathcal{G}$ guarantees that $m' = g(m)$, where g is a possibly randomized tampering function sampled from $\mathcal{G}$. In particular, if the family of functions $\mathcal{G}$ contains only the identity function and all constant functions, then the corresponding non-malleable reduction is a non-malleable code for $\mathcal{T}$.

Motivation: tamper-resilient hardware. The relaxed guarantees of a non-malleable code may seem a bit arbitrary at first glance, however the object has natural applications in tamper resilient hardware. In the 90s, high profile side-channel attacks on a number of cryptographic schemes were published that broke security by evaluating the schemes on a sequences of algebraically-related keys [Bih93, KSW96]. A number of ad hoc solutions for these "related-key attacks" were suggested, and eventually theoretical solutions were proposed by Gennaro, Lysyanskaya, Malkin, Micali, and Rabin [GLM+03] as well as Ishai, Prabhakaran, Sahai, and Wagner [IPSW06].

In [GLM+03], the authors addressed tampering attacks with a solution that assumes a (public) tamper and leakage resilient circuit in conjunction with leakage resilient memory. The justification for using tamper and leakage resilient-hardware was two-fold: (1) leakage-resilience had been addressed far more systematically in the literature and existing approaches could be applied off-the-shelf, (2) because the tamper and leakage resilient circuit was public (in particular, contained no secret keys), it was safer to assume appropriately hardened hardware could be responsibly manufactured. Their approach was to sign the contents of memory with strong signature scheme. Unfortunately, this also makes it infeasible to update the memory without a secret key (which would again need to be protected). In [IPSW06], it was shown how to compile a circuit into a tamper-resilient one, building on ideas from secure computation. Unfortunately, the tampering attacks handled by this approach are quite limited and it has proven difficult to extend their results to more general tampering attacks.

Dziembowski, Pietrzak and Wichs motivated non-malleable codes as a means of extending the approach of [GLM+03]. They considered the same model of a tamper and leakage resilient (public) circuit with leakage-resilient memory, but instead of signing the memory (using a secret key), the memory is encoded with a (public) non-malleable code. This allows the tamper and leakage resilient circuit to update any state in the memory itself by decoding, computing, and re-encoding. In contrast with [GLM+03], where a trusted third party holding the secret signing key is needed to sign new memory contents, this achieves non-malleability for stateful functionalities. The downside is that while a strong signature scheme is resilient to arbitrary polynomial time tampering attacks, efficient non-malleable codes (because they are public) have no hope of handling such attacks (See Feasibility below, as well as Section 7 for more details).

Feasibility. It is not difficult to see that non-malleable codes only exist for restricted classes of channels $\mathcal{T}$: otherwise one can always consider the channel that decodes the message, flips a bit, and re-encodes the resulting message. So the natural question to ask is against which classes of tampering channels is it possible to build non-malleable codes. As a first result, Dziembowski, Pietrzak, and Wichs gave an efficient, explicit construction of a non-malleable code against channels that can tamper each codeword bit independently (so called "bitwise tampering"). They additionally provided a non-constructive argument

that ε-non-malleable codes exist for any class of channels that is not too big, i.e. $\log\log|\mathcal{T}| < n - 2\log 1/\varepsilon$. They left constructing explicit codes for larger, richer classes of channels as an open problem.

Split-state non-malleable codes. A well-studied class of tampering functions is the 2-split-state model where the codeword consists of two states L and R, and the adversary tampers with each of these states independently. This is a very large class of tampering channels that, in particular, includes the bitwise tampering family of channels mentioned above. We now sketch the landscape of this area and particularly summarize the results on 2-split-state NMCs in Table 1. In [DPW10], in addition to introducing non-malleable codes, the authors also introduced a model of tampering called the t-split-state model, where the codeword consists of t independently tamperable states. They give the first NMC constructions in the n-split-state model[3] (where n is the codeword length) and the 2-split-state model (using random oracles). Dziembowski, Kazana and Obremski [DKO13] provided the first construction of 2-split-state NMCs without any assumptions. Their construction enabled encoding of 1-bit messages and used two-source extractors. The first NMC in the 2-split-state model for k-bit messages was given by Aggarwal, Dodis and Lovett [ADL14], which used inner product extractors with tools from additive combinatorics. In [CG14a], Cheraghchi and Guruswami studied the optimal rate of the non-malleable codes for various tampering families, where the rate of a code is defined as $\frac{\mathsf{message\ length}}{\mathsf{codeword\ length}}$. In particular, they showed that the optimal achievable rate for the t-split-state family is $1 - 1/t$. Note that in the split-state tampering model, having as few states as possible is most desirable, with 2 states being the best achievable. By the above result, the best possible rate for the 2-split-state model is therefore $\frac{1}{2}$. A long series of works[4] [CG14b, CZ14, ADKO15a, CGL16, Li17, Li19a, KOS17, KOS18, GMW18, AO20, ASK+22] has made significant progress towards achieving this rate. We now discuss some of these results. The work of Cheraghchi and Guruswami [CG14b] gave the first optimal rate non-malleable code in the n-split-state model (where n is the codeword length). More importantly, this work introduced non-malleable two-source extractors and demonstrated that these special extractors can be used to generically build 2-split-state NMCs. This connection has led to several fascinating works [CZ14, CGL16, Li17, Li19a] striving to improve the rate and number of states of non-malleable codes. Most notably, Chattopadhyay and Zuckerman [CZ14] built a 10-state NMC with a constant rate, making this the first constant rate construction with a constant number of states. They achieve their result by first building a non-malleable extractor with 10 sources and then using the connection due to [CG14a] to obtain the corresponding non-malleable code. The work of Kanukurthi, Obbattu and Sekar [KOS17] used seeded extractors to build a compiler that transforms a low rate non-malleable code into one with high rate and, in particular, obtained a rate 1/3, 4−state non-malleable code. This was subsequently improved to three states in the works of Kanukurthi, Obbattu and Sekar [KOS18] as well as Gupta, Maji and Wang [GMW18]. Li [Li19a] obtained 2-split-state NMC with rate $O(\frac{\log\log\log(1/\varepsilon)}{\log\log(1/\varepsilon)})$ (where ε is the error). Particularly, this gave a rate of $O(\log\log(n)/\log(n))$, for negligible error $\varepsilon = 2^{-\Omega(n)}$, and a constant rate for constant error, making this the first constant rate scheme in the 2−split-state model. The concept of non-malleable reductions due to [ADKO15a] was used to build the first constant rate NMC with negligible error in the 2-split-state model [AO20]. In a recent work, [ASK+22], it was shown that the construction from [KOS17] (with rate 1/3) is actually non-malleable even against 2 split-state tampering (and hence is nearly an optimal rate construction for 2 split-state tampering).

Applications of split-state non-malleable codes. The split-state tampering model is a very natural model. In particular, there are cryptographic settings where the separation of states is natural,

[3] We already mentioned this result above as a non-malleable code against bitwise tampering. We mention it again just to emphasize that bitwise tampering is a special case of split-state tampering.

[4] Other works have considered non-malleable codes in models other than the 2-split-state model or under computational assumptions [AAG+16, FMNV14, AGM+15, JW15, AKO17, ADN+19b, DLSZ20, DKS19, CKR16, ADKO15b, CGM+16, CL17, DKO+18, GMW18, BDSKM18, FHMV17].

like in secret sharing or in multiparty computation (MPC) scenarios. Non-malleable codes in the split-state model have found many applications in achieving security against physical (leakage and tampering) attacks [DPW10, LL12], domain extension of encryption schemes [CMTV15, CDTV16], non-malleable commitments [GPR16], non-malleable oblivious transfer [**?**] non-malleable secret sharing [GK18, ADN$^+$19a, BS19, SV19], non-malleable oblivious transfer [IKSS21], and privacy amplification [CKOS19]. We discuss the application to non-malleable commitments in more detail in Section 8.

Additionally, non-malleable codes in the split-state model have found many applications in the construction of non-malleable codes against other important and natural tampering families, as mentioned below.

- Decision tree tampering ([BGW19]): each tampered output symbol is a function of a small polynomial number of (adaptively chosen) queries to codeword symbols.
- Small-depth circuit tampering ([BDSG$^+$18, BGW19]): the tampered codeword is produced by a boolean circuit of polynomial size and nearly logarithmic depth.
- (Bounded) Polynomial-size circuit tampering ([BDL21]): the tampered codeword is produced by circuit of bounded polynomial size, n^d for some constant d where n is the codeword length.
- Continuous NMCs ([ADN$^+$19b]): the tampering is still split-state, but the adversary is allowed to tamper repeatedly until the tampering is detected.

The applications to decision tree tampering, small-depth circuits, and polynomial size circuit tampering are discussed in Section 7.

1.1 Organization of the paper

- Section 2 contains preliminaries and definition of non-malleable reductions, and the reader may refer to it when required.
- Section 3 contains a gentle introduction to different variants of non-malleable codes and their properties such as secret sharing and leakage-resilience.
- In Section 4, we give the first, and arguably the simplest construction of non-malleable codes in the split-state model [ADL14]. The simplicity made it a particularly useful tool for several follow-up works that required non-standard properties from the underlying non-malleable code.
- In Section 5, we briefly mention two-source non-malleable extractors, and their connection to non-malleable codes in the split-state model, as well as to other cryptographic primitives.
- In Section 6, we give an overview of the rate amplification technique from [KOS17, KOS18, AO20, ASK$^+$22] that gives an almost optimal rate non-malleable code in the split-state model.
- In Section 7, we survey some of the applications of split-state non-malleable codes to constructing non-malleable codes against computationally bounded tampering classes. In particular, we give an overview of the techniques in the following works: [BGW19] for small-depth decision trees, [BDSG$^+$18] for small-depth circuit tampering, and [BDL21] for polynomial size circuit tampering.
- In Section 8, we give a construction of a non-malleable commitment scheme due to [GPR16] that is one of the most important applications of non-malleable codes in the split-state model.

Work	Rate
Dziembowski, Pietrzak, Wichs [DPW10]	1/6 (Existential, Random Oracle Model)
Cheraghchi, Guruswami [CG14a]	1/2 (Existential, Lower bound)
Dziembowski, Kazana, Obremski [DKO13]	$\Omega(1/n)$ (Only for 1-bit messages)
Aggarwal, Dodis, Lovett, Briët [ADL14, Agg15, AB16]	$\Omega(1/n^{4/5})$
Chattopadhyay, Goyal, Li [CGL16]	$n^{-\Omega(1)}$
Li [Li17]	$\Omega(1/\log(n))$
Li [Li19a]	$\Omega(\log\log(n)/\log(n))$
Li [Li19a]	$\Omega(1)$ (with constant error)
Aggarwal, Obremski [AO20]	$\approx 1/1,500,000$
Aggarwal, Sekar, Kanukurthi, Obremski, Obbattu [ASK+22]	1/3

Table 1: Prior Work on 2-state NMCs (n is codeword length)

2 Preliminaries

2.1 Notation and Mathematical Preliminaries

For a set T, let U_T denote a uniform distribution over T, and, for an integer ℓ, let U_ℓ denote uniform distribution over ℓ bit strings. We say that $b = a \pm \delta$ if $a - \delta \leq b \leq a + \delta$. For any random variable A and any set $\mathcal{A}$, we denote $A|_{A\in\mathcal{A}}$ to be the random variable A' such that

$$\forall a,\ \Pr[A' = a] = \Pr[A = a \mid A \in \mathcal{A}] \ .$$

The *statistical distance* between two random variables A, B is defined by

$$\Delta(A\ ;\ B) = \frac{1}{2}\sum_v |\Pr[A = v] - \Pr[B = v]| \ .$$

We use $A \approx_\varepsilon B$ as shorthand for $\Delta(A, B) \leq \varepsilon$.

Lemma 2.1. *For any function α, if $\Delta(A\ ;\ B) \leq \varepsilon$, then $\Delta(\alpha(A)\ ;\ \alpha(B)) \leq \varepsilon$.*

The *min-entropy* of a random variable W is $\mathbf{H}_\infty(W) \stackrel{\text{def}}{=} -\log(\max_w \Pr[W = w])$, and the *conditional* min-entropy of W given Z is $\mathbf{H}_\infty(W|Z) \stackrel{\text{def}}{=} -\log\left(\mathbb{E}_{z\leftarrow Z} \max_w \Pr[W = w|Z = z]\right)$.

Definition 2.1. *We say that a function* $\mathsf{Ext} : \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}$ *is a* (k, ε)-2-source extractor *if for all independent sources $X, Y \in \mathbb{F}^n$ such that min-entropy $\mathbf{H}_\infty(X) + \mathbf{H}_\infty(Y) \geq k$, we have $(Y, \mathsf{Ext}(X, Y)) \approx_\varepsilon (Y, U_m)$, and $(X, \mathsf{Ext}(X, Y)) \approx_\varepsilon (X, U_m)$.*

Lemma 2.2. *For all positive integers n and any finite field $\mathbb{F}$, and for all $\varepsilon > 0$, the inner product function $\langle\cdot,\cdot\rangle : \mathbb{F}^n \times \mathbb{F}^n \to \mathbb{F}$ is an efficient $\left((n+1)\log|\mathbb{F}| + 2\log\left(\frac{1}{\varepsilon}\right), \varepsilon\right)$ 2-source extractor.*

In particular, for n being an integer multiple of m, and interpreting elements of $\{0,1\}^m$ as elements from $\mathbb{F}_{2^m}$ and those in $\{0,1\}^n$ to be from $(\mathbb{F}_{2^m})^{n/m}$, we have that for any $\varepsilon > 0$ there exists an efficient $\left(n + m + 2\log\left(\frac{1}{\varepsilon}\right), \varepsilon\right)$ 2-source extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$.

The following is a definition of an ε-almost universal hash function.

Definition 2.2. *A function $C : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^t$ is called an* ε-almost universal hash function *if for any $x, y \in \{0,1\}^n$ such that $x \neq y$,*

$$\Pr_{R \leftarrow \{0,1\}^s}(C(R,x) = C(R,y)) \leq \varepsilon$$

The following is a standard construction of a polynomial evaluation ε-universal hash function. The parameters are from [DW09].

Lemma 2.3. *For any $n, t > 2\log n$, there exists an efficiently computable $2^{-t/2}$-almost univeral hash function $C : \{0,1\}^s \times \{0,1\}^n \to \{0,1\}^t$ with $s = 2t$.*

Lemma 2.4 (Lemma 4 of [DDV10], Lemma 9 of [ADKO15a]). *Let A, B be independent random variables and consider a sequence $V_1, \ldots, V_i$ of random variables, where for some function ϕ, $V_i = \phi_i(C_i) = \phi(V_1, \ldots, V_{i-1}, C_i)$ with each $C_i \in \{A, B\}$. Then A and B are independent conditioned on $V_1, \ldots, V_i$. That is, $I(A; B|V_1, \ldots, V_i) = 0$.*

2.2 Non-malleable Codes and Reductions

Definitions. In [ADKO15a], the notion of non-malleable codes w.r.t. to a tampering family $\mathcal{F}$ (see [DPW10]) was generalized to a more versatile notion of *non-malleable reductions* from $\mathcal{F}$ to $\mathcal{G}$. The following definitions are taken from [ADKO15a].

Definition 2.3 (non-malleable reduction). Let $\mathcal{F} \subset A^A$ and $\mathcal{G} \subset B^B$ be some classes of functions (which we call *manipulation* functions). We will write:

$$(\mathcal{F} \Rightarrow \mathcal{G}, \varepsilon)$$

and say $\mathcal{F}$ *reduces to* $\mathcal{G}$, if there exist an efficient randomized *encoding* function $E : B \to A$, and an efficient deterministic *decoding* function $D : A \to B$, such that (a) for all $x \in B$, we have $D(E(x)) = x$, and (b) for all $f \in \mathcal{F}$, there exists G such that for all $x \in B$,

$$\Delta\Big(D(f(E(x)))\ ;\ G(x)\Big) \leq \varepsilon, \tag{1}$$

where G is a *distribution* over $\mathcal{G}$, and $G(x)$ denotes the distribution $g(x)$, where $g \leftarrow G$.

The pair (E, D) is called $(\mathcal{F}, \mathcal{G}, \varepsilon)$*-non-malleable reduction.*

Intuitively, $(\mathcal{F}, \mathcal{G}, \varepsilon)$-non-malleable reduction allows one to encode a value x by $y \leftarrow E(x)$, so that tampering with y by $\underline{y} = f(y)$ for $f \in \mathcal{F}$ gets "reduced" (by the decoding function $D(\underline{y}) = \underline{x}$) to tampering *with x itself* via some (distribution over) $g \in \mathcal{G}$.

In particular, the notion of *non-malleable code* w.r.t. $\mathcal{F}$, is simply a reduction from $\mathcal{F}$ to the family of "trivial manipulation functions" NM_k defined below.

Definition 2.4. Let NM_k denote the set of *trivial manipulation functions* on k-bit strings, which consists of the identity function $I(x) = x$ and all constant functions $f_c(x) = c$, where $c \in \{0,1\}^k$.

We say that a pair (E, D) defines an $(\mathcal{F}, k, \varepsilon)$*-non-malleable code*, if it defines a $(\mathcal{F}, \mathsf{NM}_k, \varepsilon)$-non-malleable reduction.

The utility of non-malleable reductions comes from the following natural composition theorem that was shown in [ADKO15a], which allows to gradually make our tampering families simpler and simpler, until we eventually end up with a non-malleable code (corresponding to the trivial family NM_k).

Theorem 2.1 (Composition). *If $(\mathcal{F} \Rightarrow \mathcal{G}, \varepsilon_1)$ and $(\mathcal{G} \Rightarrow \mathcal{H}, \varepsilon_2)$, then $(\mathcal{F} \Rightarrow \mathcal{H}, \varepsilon_1 + \varepsilon_2)$.*

We will also need the following trivial observation.

Observation 2.1 (Union). *Let (E, D) be an $(\mathcal{F}, \mathcal{H}, \varepsilon)$ and a $(\mathcal{G}, \mathcal{H}, \varepsilon')$ non-malleable reduction . Then (E, D) is an $(\mathcal{F} \cup \mathcal{G}, \mathcal{H}, \max(\varepsilon, \varepsilon'))$ non-malleable reduction .*

Useful Tampering Families. We define several natural tampering families we will use in this work. For this, we first introduce the following "direct product" operator on tampering families:

Definition 2.5. Given tampering families $\mathcal{F} \subset A^A$ and $\mathcal{G} \subset B^B$, let $\mathcal{F} \times \mathcal{G}$ denote the class of functions h from $(A \times B)^{A \times B}$ such that

$$h(x) = h_1(x_1) \| h_2(x_2)$$

for some $h_1 \in \mathcal{F}$ and $h_2 \in \mathcal{G}$ and $x = x_1 \| x_2$, where $x_1 \in A, x_2 \in B$.

We also let $\mathcal{F}^1 := \mathcal{F}$, and, for $t \geq 1$, $\mathcal{F}^{t+1} := \mathcal{F}^t \times \mathcal{F}$.

We can now define the following tampering families:

- $\mathcal{S}_n = (\{0,1\}^n)^{\{0,1\}^n}$ denotes the class of *all* manipulation functions on n-bit strings.
- $\mathcal{S}_{n,p} = (\mathbb{F}_p^n)^{\mathbb{F}_p^n}$ denotes the class of *all* manipulation functions on $\mathbb{F}_p^n$.
- Given $t > 1$, $\mathcal{S}_{n,p}^t$ denotes the tampering family in the *t-split-state model*, where the attacker can apply t arbitrarily correlated functions $h_1, \ldots, h_t$ to t separate, parts of memory each in $\mathbb{F}_p^n$ (but, of course, each h_i can only be applied to the i-th part individually).
- Given a prime p, $\mathcal{AFF}_p$ denotes the class of all affine functions parametrized by $a, b \in \mathbb{F}_p$ such that $f_{a,b}(x) := ax + b$ for all $x \in \mathbb{F}_p$.

2.3 Basic techniques

The following is a simple result from [ADL14] that says that if two pairs of random variables $(X_1, X_2), (Y_1, Y_2)$ are statistically close to each other then X_1 conditioned on X_2 is statistically close to Y_1 conditioned on Y_2.

Lemma 2.5. *Let $X_1, Y_1 \in \mathcal{A}_1$, and $Y_1, Y_2 \in \mathcal{A}_2$ be random variables such that $\Delta((X_1, X_2)\ ;\ (Y_1, Y_2)) \leq \varepsilon$. Then, for any non-empty set $\mathcal{A}' \subseteq \mathcal{A}_1$, we have*

$$\Delta(X_2 \mid X_1 \in \mathcal{A}'\ ;\ Y_2 \mid Y_1 \in \mathcal{A}') \leq \frac{2\varepsilon}{\Pr(X_1 \in \mathcal{A}')}\ .$$

The following is a variant of a similar simple lemma from [DKO13, ADL14]. The proof is just a simple application of triangle inequality.

Lemma 2.6. *Let S be some random variable distributed over a set $\mathcal{S}$, and let $\mathcal{S}_1, \ldots, \mathcal{S}_j$ be a partition of $\mathcal{S}$. Let $\phi : \mathcal{S} \to \mathcal{T}$ be some function, and let $D_1, \ldots, D_j$ be some random variables over the set $\mathcal{T}$. Assume that for all $1 \leq i \leq j$,*

$$\Delta\left(\phi(S)|_{S \in \mathcal{S}_i}\ ;\ D_i\right) \leq \varepsilon_i.$$

Then

$$\Delta\left(\phi(S)\ ;\ D\right) \leq \sum \varepsilon_i \Pr[S \in S_i]\ ,$$

for some random variable $D \in \mathcal{T}$ such that for all d $\Pr[D = d] = \sum_i \Pr[S \in \mathcal{S}_i] \cdot \Pr[D_i = d]$.

Lemma 2.7. *Let $\mathbb{F}$ be a finite field. Let $X = (X_1, X_2) \in \mathbb{F} \times \mathbb{F}$ be a random variable. Assume that for all $a \in \mathbb{F}$ not both zero, $\Delta(X_1 + aX_2\ ;\ U_{\mathbb{F}}) \leq \varepsilon$. Then $\Delta((X_1, X_2)\ ;\ U_{\mathbb{F}_p}, X_2) \leq \varepsilon|\mathbb{F}|^2$.*

Lemma 2.8. *Let $X \in \mathbb{F}$ be a random variable. Assume that $\Delta(X\ ;\ U_{\mathbb{F}}) \geq \varepsilon$. Then if X' is an i.i.d copy of X then*

$$\Pr[X = X'] \geq \frac{1+\varepsilon^2}{|\mathbb{F}|}\ .$$

Lemma 2.9. *Let $Z = (X, Y) \in \mathbb{F}^n \times \mathbb{F}^n$ be a random variable, and let $Z' = (X', Y')$ be an i.i.d copy of Z. Then*

$$\Pr[\langle X, Y\rangle = \langle X', Y'\rangle] \leq \Pr[\langle X, Y\rangle = \langle X', Y\rangle].$$

3 Basic properties and variants of non-malleable codes and continuous non-malleable codes

In this section we will discuss various basic properties of the 2− split state non-malleable codes, as well as numerous variants of their definitions.

3.1 A few examples.

As a warm up we will go through few basic examples of codes that are not non-malleable.

Example 3.1. *To encode $m \in \mathbb{F}$ we pick $L \in \mathbb{F}$ and $R \in \mathbb{F}$ uniformly random such that $L + R = m$.*

Above clearly is not a non-malleable code: pick $\underline{L} = L+1$ and $\underline{R} = R$ then $\mathsf{Dec}(\underline{L}, \underline{R}) = \mathsf{Dec}(L, R)+1$, we have changed the message but the message is not independent of the original message.

Example 3.2. *To encode $m \in \mathbb{F}_p$, pick $L, R \in \mathbb{F}_p^n$ uniformly random such that $\langle L, R\rangle = m$, where $\langle ., .\rangle$ stands for the inner product over $\mathbb{F}_p$.*

Again, the attack is quite simple: pick $a \notin \{0, 1\}$ and let $\underline{L} = a \cdot L$, $\underline{R} = R$, then $\mathsf{Dec}(\underline{L}, \underline{R}) = a \cdot \mathsf{Dec}(L, R)$. Again the message has changed but remained strongly correlated with the original message. Above attack depends on $a \notin \{0, 1\}$, however over $\mathbb{F}_2$ we won't have any other option. so maybe let's consider the following code:

Example 3.3. *To encode $m \in \mathbb{F}_2$ pick $L, R \in \mathbb{F}_2^n$ uniformly at random such that $\langle L, R\rangle = m$, where $\langle ., .\rangle$ stands for the inner product over $\mathbb{F}_2$.*

Sadly again there is a simple attack let $\underline{L}$ (and $\underline{R}$ respectively) be equal to L (and R respectively) on all positions except the last, we will set the last position to $(\underline{L})_n = 1$ (and $(\underline{R})_n = 1$). Now with probability $\frac{3}{4}$ we have $\mathsf{Dec}(\underline{L}, \underline{R}) = \mathsf{Dec}(L, R) \oplus 1$, and with probability $\frac{1}{4}$ we have $\mathsf{Dec}(\underline{L}, \underline{R}) = \mathsf{Dec}(L, R)$. This can not be a non-malleable code, as [DKO13] showed for single bit message[5]: if we can flip the output of the Decoder with probability greater then $\frac{1}{2}$ (plus some non-negligible factor) then the code can not be non-malleable.

3.2 Secret sharing.

We will show that the 2− split state non-malleable code has to be a 2 out of 2 secret sharing. Let m_0, m_1 be two messages and let $\mathsf{Enc}(m_0) = X_0, Y_0$ and $\mathsf{Enc}(m_1) = X_1, Y_1$. If given X_i we could guess i we would be able to tamper the codewords in a way that $\mathsf{Dec}(\underline{X}_0, \underline{Y}_0) = a_0$ and $\mathsf{Dec}(\underline{X}_1, \underline{Y}_1) = a_1$, where a_0, a_1 are two fixed distinct messages (different than m_0, m_1). This clearly breaks non-malleability since the original messages m_0, m_1 are not preserved but the messages after tampering are correlated with the original messages: tampered message is a_i if and only if the original message was m_i. This conveys the main intuition: is message is not preserved then tampered message should not reveal the original message.

[5]We also assume that $\mathsf{Dec} \neq \perp$, i.e. there are no invalid codewords

Let us construct the above-mentioned attack: find $\ell_0, \ell_1, r, a_0 \neq a_1$ such that $\mathsf{Dec}(\ell_0, r) = a_0$ and $\mathsf{Dec}(\ell_1, r) = a_1$ (we know they must exist else r alone would determine the output of the decoder, and we could carry on the same attack on the right state). Now for the tampering: we will completely overwrite the right state $Y_i \to r$, and given X_i if we think $i = 0$ we will tamper $X_i \to \ell_0$, if we think that $i = 1$ then we tamper $X_i \to \ell_1$, this gives the desired result.

To be more precise we recall theorem from [ADKO15b].

Lemma 3.1 (from [ADKO15b]). *If* $(\mathsf{Enc}, \mathsf{Dec})$ *is an* $\varepsilon-$ *non-malleable code then for any two messages* m_0, m_1*, and for* $\mathsf{Enc}(m_0) = X_0, Y_0$ *and* $\mathsf{Enc}(m_1) = X_1, Y_1$*, we get:*

$$X_0 \approx_{2\varepsilon} X_1 \qquad \text{and} \qquad Y_0 \approx_{2\varepsilon} Y_1$$

3.3 Leakage-resilience.

So far we only discussed active adversary that tampers the states. It is natural to consider its weaker version: a passive adversary.

Long time ago we thought of cryptographic device as a box that holds a secret key and has a strictly defined interface, and the attacker is only allowed to use that well defined input/output interface. But no device is a true blackbox, it consumes electricity, emits electromagnetic radiation, has a heat signature and a running time, those values were not predicted in the clean blackbox-security model, and thus the first wave of *passive attacks* was born. Now the adversary could exploit side-channel information like the one mentioned above and with its help break the security of the device. We often refer to such side information as *leakage*, and adversary that exploits it as a *passive adversary.*

We can start with the following theorem:

Theorem 3.1 (from [AKO17]). *Let* $k \geq 3$*, and let* $\varepsilon < 1/20$*. Let* $\mathsf{Enc} : \{0,1\}^k \to \{0,1\}^n \times \{0,1\}^n$*,* $\mathsf{Dec} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^k$ *be an* $\varepsilon-$*non-malleable code in the split state model. For every sets* $A, B \subset \{0,1\}^n$ *and every messages* $m_0, m_1 \in \{0,1\}^k$

$$|\Pr(\mathsf{Enc}(m_0) \in A \times B) - \Pr(\mathsf{Enc}(m_1) \in A \times B)| \leq \varepsilon\ .$$

Before we get to the proof, notice that one can run the above lemma for following sets: $A \times \{0,1\}^n$ and $\{0,1\}^n \times B$ and $A \times B$ (for the set $\{0,1\}^n \times \{0,1\}^n$ statement is trivial) this means that given the indicators $\mathbf{1}_A(X_i)$, $\mathbf{1}_B(Y_i)$ we can not distinguish between $i = 0$ and $i = 1$ (where (X_i, Y_i) encode message m_i). One should remark that while the above is just a one bit leakage, one can easily leverage it to the arbitrary size t leakage at the price of the 2^t multiplicative penalty in the error, we refer to the similar reasoning below Remark 3.4.

This is only a mild version of leakage resilience and we will expand it further in this section.

Below we go over the proof of theorem 3.1, we remark that similar reasoning forms the core of remark 3.4 and theorem 3.4.

Proof. We claim that there exist $x, y, z, w \in \{0,1\}^n$ such that $m_0, m_1, \mathsf{Dec}(x, w)$, $\mathsf{Dec}(z, w)$, and $\mathsf{Dec}(z, y)$ are all different from $\mathsf{Dec}(x, y)$. Before proving this claim, we show why this implies the given result. Let $\mathsf{Enc}(m) = X, Y$, consider the tampering functions f, g such that $f(X) = x$ if $X \in A$, and $f(X) = z$, otherwise, and $g(Y) = y$ if $Y \in B$, and $g(Y) = w$, otherwise. Thus, for $b = 0, 1$, $\mathsf{Dec}(\underline{X}, \underline{Y}) = \mathsf{Dec}(x, y)$ if and only if $\mathsf{Enc}(m_b) \in A \times B$. The result then follows from the ε-non-malleability of $(\mathsf{Enc}, \mathsf{Dec})$.

Now, to prove the claim, we will use the probabilistic method. Let U be uniform in $\{0,1\}^k$, and let $X, Y \leftarrow \mathsf{Enc}(U)$. Furthermore, let $W, Z \in \{0,1\}^n$ be uniform and independent of X, Y, U. We claim that X, Y, Z, W satisfy the required property with non-zero probability.

It is easy to see that the probability that $\mathsf{Dec}(X, Y) = U$ is either of m_0 or m_1 is at most $2/2^k$. Also, by Lemma 3.1, we have that except with probability 2ε, X is independent of U. Also, W is independent of U. Thus, the probability that $\mathsf{Dec}(X, W) = U$ is at most $2\varepsilon + 1/2^k$. Similarly, the probability that

$\mathsf{Dec}(Z, Y) = U$ is at most $2\varepsilon + 1/2^k$. Finally, W, Z are independent of U, and so the probability that $\mathsf{Dec}(Z, W) = U$ is at most $\frac{1}{2^k}$.

Thus, by union bound, the probability that X, Y, Z, W do not satisfy the condition of the claim is at most $\frac{5}{2^k} + 4\varepsilon \leq \frac{5}{8} + 4\varepsilon < 1$. □

As we already hinted, above is only a mild version of leakage resilience, for full version we would expect for example that the decoded message along with the leakage doesn't reveal anything about the message.

To formalize the above intuitive notion we first have to recall the original definition from [DPW10]:

Definition 3.1. (Non-Malleable Code from [DPW10].) *Let* $(\mathsf{Enc} : \mathcal{M} \to \mathcal{X} \times \mathcal{X}, \mathsf{Dec} : \mathcal{X} \times \mathcal{X} \to \mathcal{M} \cup \{\perp\})$ *be an encoding scheme. For* $f, g : \mathcal{X} \to \mathcal{X}$ *and for any* $m \in \mathcal{M}$ *define the experiment* $\mathsf{DPWTamper}_m^{f,g}$ *as:*

$$\mathsf{DPWTamper}_m^{f,g} = \left\{ \begin{array}{c} (X, Y) \leftarrow \mathsf{Enc}(m), \\ \underline{X} := f(X), \underline{Y} := g(Y) \\ \underline{m} := \mathsf{Dec}(\underline{X}, \underline{Y}) \\ \textit{output: } \underline{m} \end{array} \right\}$$

Above represents the state of the message after tampering. The claim that the message has either not changed or is completely independent of the original message is expressed in a following way: We say that the encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ *is* ε-DPW-non-malleable in split-state model *if for every functions* $f, g : \mathcal{X} \to \mathcal{X}$ *there exists distribution* $D^{f,g}$ *on* $\mathcal{M} \cup \{\mathsf{same}, \perp\}$ *(without the access to the original message) such that for every* $m \in \mathcal{M}$ *we have*

$$\mathsf{DPWTamper}_m^{f,g} \approx_\varepsilon \left\{ \begin{array}{c} d \leftarrow D^{f,g} \\ \textit{if } d = \mathsf{same} \textit{ then output } m \\ \textit{otherwise output } d. \end{array} \right\}$$

In other words there exists a simulator $D^{f,g}$ that can simulate the tampering experiment, the simulator has no access to original message: he can only output special symbol same that will be replaced with original message.

Adding leakage resilience do non-malleable codes. To add a true leakage resilience we have few options:

1. Tampering functions might depend on the leakages (e.g. [LL12, ADKO15b, BGW19]):

$$\mathsf{TamperLeak}_m^{f,g,\mathsf{Leak}_X,\mathsf{Leak}_Y} = \left\{ \begin{array}{c} (X, Y) \leftarrow \mathsf{Enc}(m), \\ \underline{X} := f(X, \mathsf{Leak}_Y(Y)), \underline{Y} := g(Y, \mathsf{Leak}_X(X)) \\ \underline{m} := \mathsf{Dec}(\underline{X}, \underline{Y}) \\ \text{output: } \underline{m} \end{array} \right\}$$

2. We can also consider outputting the leakage along with the tampered message (e.g. [BFO+20, ASK+22]):

$$\mathsf{TamperLeak}_m^{f,g,\mathsf{Leak}_X,\mathsf{Leak}_Y} = \left\{ \begin{array}{c} (X, Y) \leftarrow \mathsf{Enc}(m), \\ \underline{X} := f(X), \underline{Y} := g(Y) \\ \underline{m} := \mathsf{Dec}(\underline{X}, \underline{Y}) \\ \text{output: } \underline{m}, \mathsf{Leak}_X(X), \mathsf{Leak}_Y(Y) \end{array} \right\}$$

3. And of course we can also consider a combination of above, where tampering depends on the leakage and the leakage is also part of the tampering output:

$$\mathsf{TamperLeak}_m^{f,g,\mathsf{Leak}_X,\mathsf{Leak}_Y} = \left\{ \begin{array}{c} (X,Y) \leftarrow \mathsf{Enc}(m), \\ \underline{X} := f(X, \mathsf{Leak}_Y(Y)), \underline{Y} := g(Y, \mathsf{Leak}_X(X)) \\ \underline{m} := \mathsf{Dec}(\underline{X}, \underline{Y}) \\ \text{output: } \underline{m}, \mathsf{Leak}_X(X), \mathsf{Leak}_Y(Y) \end{array} \right\}$$

In all of the above we expect the (modified) simulator to be indistinguishable from the tampering experiment. In the case of the experiment 2 and experiment 3 we slightly modify the simulator: The $D^{f,g,\mathsf{Leak}_X,\mathsf{Leak}_Y}$ simulates not only the message but also the leakage:

$$\mathsf{TamperLeak}_m^{f,g,\mathsf{Leak}_X,\mathsf{Leak}_Y} \approx_\varepsilon \left\{ \begin{array}{c} (d, \ell_X, \ell_Y) \leftarrow D^{f,g,\mathsf{Leak}_X,\mathsf{Leak}_Y} \\ \text{if } d = \mathsf{same} \text{ then output } m, \ell_X, \ell_Y \\ \text{otherwise output } d, \ell_X, \ell_Y. \end{array} \right\}$$

Remark 3.1. *Usually we consider* $\mathsf{Leak}_X, \mathsf{Leak}_Y$ *to be bounded output size leakages. Also* Leak_X *and* Leak_Y *might be a series of adaptive leakages depending on each other- then one has to apply lemma 2.4 to obtain independence of* X *and* Y *given the leakages. We have to remark here that lemma 2.4 states that* X *and* Y *are independent given* $\mathsf{Leak}_X = \ell_X$ *and* $\mathsf{Leak}_Y = \ell_Y$*, however one has to remain vigilant since* $X|\mathsf{Leak}_X = \ell_X$ *and* $Y|\mathsf{Leak}_Y = \ell_Y$ *might not be efficiently samplable sources thus the extension to the adaptive leakage case is straight forward only in the information theoretic world.*

Remark 3.2. *The second definition might seem a bit artificial, however it is quite useful for technical reasons. Sometimes, non-malleable code is merely one of many building blocks of bigger protocol/application, and the leakage is a byproduct of the technical proof- other parts of the protocol might behave differently depending on the non-malleable encoding (which is most conveniently modeled as a leakage), thus non-malleable code is tampered in a usual way while rest of the protocol leaks extra information.*

The first compiler that returns a leakage resilient (with respect to the variant 1) non-malleable code was given by [ADKO15b], it could tolerate up to $\frac{1}{12}$ leakage rate (i.e. output size of leakage functions can be up to $\frac{1}{12}$ of the input size), but it required a symmetric decoder ($\mathsf{Dec}(X,Y) = \mathsf{Dec}(Y,X)$). Later Ball, Guo and Wichs gave a better compiler:

Theorem 3.2 (from [BGW19, BDL21]). *For any* $\alpha \in [0, \frac{1}{4})$ *there exists a compiler that takes any* $2-$*split state non-malleable code and outputs a leakage resilient (with respect to definition variants 1, 2, and 3) non-malleable code with leakage rate* α*. The rate of the new code is* $\Theta(\text{original} - \text{rate})$ *and the error stays the same except for an extra* $\exp(-\Omega(n))$ *factor (where* n *is the new code's length).*

Remark 3.3. *[BGW19] originally only showed that their compiler worked for variant 1. However, [BDL21] later extended their analysis to the latter variants.*

Also for the variants 2 and 3 we have the following result:

Theorem 3.3 (from [BFO$^+$20]). *Any* $2-$*split-state* $\varepsilon-$*non-malleable code is also* $2^t \cdot \varepsilon-$*non-malleable code that tolerates up to* t *bits of leakage (with respect to definition 2 or 3).*

Remark 3.4. *Originally the above paper considered definition 2 only, but simple inspection of the proof gives the security with respect to variant 3 too.*

The idea of the proof is quite simple: we guess the leakage functions (thus the penalty 2^t) and tampering function check if the leakage is consistent with their view, if any of the views does not match the guessed leakage then the tampering aborts (f or g outputs $\perp$, and the decoder aborts). Else, if

the guessed leakage is consistent with the views of the tampering functions the tampering happens as intended. Above expands the power of tampering functions: instead of $f, g : \{0,1\}^n \to \{0,1\}^n$ we have $f, g : \{0,1\}^n \to \{0,1\}^n \cup \{\perp\}$, this is without a loss of generality- in a similar fashion as in theorem 3.1 we can show that adding $\perp$ as a possible output of the tampering functions doesn't break the definition.

Later [ASK+22] expanded the result from theorem 3.3 for augmented (see section 3.4) non-malleable codes.

Theorem 3.4 (from [ASK+22]). *Any $2-$split-state augmented $\varepsilon-$non-malleable code is also an augmented $2^t \cdot \varepsilon-$non-malleable code that tolerates up to t bits of leakage (with respect to definition 2 or 3).*

Similarly, the [BGW19] compiler also preserves the augmented property:

Theorem 3.5 (from [BDL21]). *For any $\alpha \in [0, 1/4)$, and 2-split-state augmented $\varepsilon-$non-malleable code can be compiled into an augmented split-state $\varepsilon+\exp(-\Omega(n))$-non-malleable code with rate $\Theta(\text{original} - \text{rate})$ and leakage rate α (with respect to any variant above).*

3.4 Augmented NMCs

Many applications require an extra property, namely that adversary on top of receiving a tampered message can get one of the states (similar to the leakage resilience discussed above).

Definition 3.2 (Left-augmented NMC). *Let $(\mathsf{Enc} : \mathcal{M} \to \mathcal{X} \times \mathcal{X}, \mathsf{Dec} : \mathcal{X} \times \mathcal{X} \to \mathcal{M} \cup \{\perp\})$ be an encoding scheme. For $f, g : \mathcal{X} \to \mathcal{X}$ and for any $m \in \mathcal{M}$ define the experiment $\mathsf{Tamper}_m^{f,g}$ as:*

$$\mathsf{Tamper}_m^{f,g} = \left\{ \begin{array}{c} (X, Y) \leftarrow \mathsf{Enc}(m), \\ \underline{X} := f(X), \underline{Y} := g(Y) \\ \underline{m} := \mathsf{Dec}(\underline{X}, \underline{Y}) \\ \textit{output: } \underline{m}, X \end{array} \right\}$$

We say that the encoding scheme $(\mathsf{Enc}, \mathsf{Dec})$ is left-augmented ε-non-malleable in $2-$split-state model *if for every functions $f, g : \mathcal{X} \to \mathcal{X}$ there exists distribution $D^{f,g}$ on $\mathcal{M} \cup \{\mathsf{same}, \perp\}$ (without the access to the original message) such that for every $m \in \mathcal{M}$ we have*

$$\mathsf{Tamper}_m^{f,g} \approx_\varepsilon \left\{ \begin{array}{c} (d, x) \leftarrow D^{f,g} \\ \textit{if } d = \mathsf{same} \textit{ then output } m, x \\ \textit{otherwise output } d, x. \end{array} \right\}$$

Symmetrically we can consider right-augmented property where right state is revealed.

Most of the known constructions like [ADL14, CGL16, Li17, Li19b, ASK+22] are augmented (both left and right augmented). Interestingly [KOS18] non-malleable randomness encoder (see section 6.1 for details) is right-augmented but not left-augmented.

3.5 Simulation vs game.

Definition 3.1 is the most common simulation-based definition. However in some situations it is actually more convenient to consider a game based definition, where the adversary picks two messages m_0, m_1, the challenger encodes m_b for uniformly chosen b, and the adversary has to guess b based on the tampering of $\mathsf{Enc}(m_b) = (X_b, Y_b)$.

In particular the following alternative definition of non-malleable code, will give a smoother transition to the subsequent definitions in this section.

The transition from simulator to game is not quite trivial: let $\mathsf{Enc}(m_0) = L, R$ imagine that the tampering $f(X) = L$ and $g(Y) = R$ now both messages have been completely overwritten and both

tampering experiments should output m_0. However, notice that tampering experiment $\mathsf{Tamper}^{f,g}_{m_0}$ has two options: it can answer m_0 or it can answer same, while $\mathsf{Tamper}^{f,g}_{m_1}$ can only answer m_0. In the above example $\mathsf{Tamper}^{f,g}_{m_0}$ can not answer same else it will be distinguishable. To solve the dilemma we have to add an extra "helper" sitting inside the tampering experiment that will decide if the tampering experiment should output same or $\underline{m}$.

Definition 3.3. (Game definition for non-malleable code, from [AKO17].) *We say that an encoding scheme* $(\mathsf{Enc} : \mathcal{M} \to \mathcal{X} \times \mathcal{X}, \mathsf{Dec} : \mathcal{X} \times \mathcal{X} \to \mathcal{M} \cup \{\perp\})$ *is* ε-non-malleable in split-state model *if for every functions* $f, g : \mathcal{X} \to \mathcal{X}$ *there exists family of distributions* $\{D^{f,g}_{x,y}\}_{x,y\in\mathcal{X}}$ *each on* $\{0,1\}$ *such that for every* $m_0, m_1 \in \mathcal{M}$

$$\mathsf{Tamper}^{f,g}_{m_0} \approx_\varepsilon \mathsf{Tamper}^{f,g}_{m_1}$$

where

$$\mathsf{Tamper}^{f,g}_{m} = \left\{ \begin{array}{c} (X,Y) \leftarrow \mathsf{Enc}(m), \\ \textit{output } \mathbf{same} \textit{ if } \mathsf{Dec}(X,Y) = \mathsf{Dec}(f(X), g(Y)) \wedge D^{f,g}_{X,Y} = 0 \\ \textit{else output: } \mathsf{Dec}(f(X), g(Y)) \end{array} \right\}$$

In [AKO17](Appendix A) authors show the equivalence of the definitions 3.1 and 3.3.

Theorem 3.6 (from [AKO17]). *If* $(\mathsf{Enc}, \mathsf{Dec})$ *is an* $\varepsilon-$*non-malleable code according to the game-based definition then it is also an* $\varepsilon-$*non-malleable code according to the definition from [DPW10].*

Theorem 3.7 (from [AKO17]). *If* $(\mathsf{Enc}, \mathsf{Dec})$ *is an* $\varepsilon-$*-non-malleable code according to the definition from [DPW10], then it is* $4\varepsilon-$*non-malleable code according to the game-based definition.*

To prove above authors construct explicit "helper" distribution. There was another game based definition already considered in [DPW10], but the above definition is easier to generalize to the definition for stronger notions of non-malleable codes.

3.6 Strong, super and super-strong variants.

Some results in the literature like [FMNV14, JW15] have considered a notion of super-strong non-malleable codes. We start with the following intermediate notion of super non-malleable codes introduced in [AKO17]. In this variant if the tampering is successful and non-trivial i.e. output is not $\perp$ or same then tampering experiment outputs the whole tampered codeword. In other words we require that valid tampering either tampering doesn't change the message, or even the tampered codeword itself doesn't carry any information about the original message.

Definition 3.4. (Super non-malleable code.) *We say that an encoding scheme* $(\mathsf{Enc} : \mathcal{M} \to \mathcal{X} \times \mathcal{X}, \mathsf{Dec} : \mathcal{X} \times \mathcal{X} \to \mathcal{M} \cup \{\perp\})$ *is* ε-super non-malleable in split-state model *if for every functions* $f, g : \mathcal{X} \to \mathcal{X}$ *there exists family of distributions* $\{D^{f,g}_{x,y}\}_{x,y\in\mathcal{X}}$ *each on* $\{0,1\}$ *such that for every* $m_0, m_1 \in \mathcal{M}$

$$\mathsf{SuperTamper}^{f,g}_{m_0} \approx_\varepsilon \mathsf{SuperTamper}^{f,g}_{m_1}$$

where $\mathsf{SuperTamper}^{f,g}_{m} =$

$$\left\{ \begin{array}{c} (X,Y) \leftarrow \mathsf{Enc}(m), \\ \textit{output } \mathbf{same} \textit{ if } \mathsf{Dec}(X,Y) = \mathsf{Dec}(f(X), g(Y)) \wedge D^{f,g}_{X,Y} = 0 \\ \textit{else if } \mathsf{Dec}(f(X), g(Y)) = \perp \textit{ output } \perp \\ \textit{else output: } (f(X), g(Y)) \end{array} \right\}$$

Remark 3.5. *This definition is clearly stronger then the standard version, since given the tampered codeword we can apply decoder and obtain the tampered message.*

In [DPW10] authors considered a strong variant. This is a variant that follows the standard definition closely except puts a restriction on the use of same- it can only be outputted only if $f(X) = X \wedge g(Y) = Y$. [6] This variant is perhaps the closest to the intuition, if the codeword is tampered then it's either invalid or it decodes to something independent of the original message.

Definition 3.5. (Strong non-malleable code.) *We say that an encoding scheme* $(\mathsf{Enc} : \mathcal{M} \to \mathcal{X} \times \mathcal{X}, \mathsf{Dec} : \mathcal{X} \times \mathcal{X} \to \mathcal{M} \cup \{\perp\})$ *is* ε-strong non-malleable in split-state model *if for every functions* $f, g : \mathcal{X} \to \mathcal{X}$ *and for every* $m_0, m_1 \in \mathcal{M}$

$$\mathsf{StrTamp}^{f,g}_{m_0} \approx_\varepsilon \mathsf{StrTamp}^{f,g}_{m_1}$$

where

$$\mathsf{StrTamp}^{f,g}_{m} = \left\{ \begin{array}{c} (X, Y) \leftarrow \mathsf{Enc}(m), \\ \textit{output } \textbf{same} \textit{ if } (X, Y) = (f(X), g(Y)) \\ \textit{else if } \mathsf{Dec}(f(X), g(Y)) = \perp \textit{ output } \perp \\ \textit{else output: } \mathsf{Dec}(f(X), g(Y)) \end{array} \right\}$$

Notice that above we do not need a "helper" distribution anymore since the condition to output same are so restrictive.

Finally one can consider both the super and strong version. Here we require that same can only be outputted if $f(X) = X \ \wedge \ g(Y) = Y$ and if the codeword is valid and not trivially tampered then the whole tampered codeword doesn't reveal any information about the original message.

Definition 3.6. (Super strong non-malleable code.) *We say that an encoding scheme* $(\mathsf{Enc} : \mathcal{M} \to \mathcal{X} \times \mathcal{X}, \mathsf{Dec} : \mathcal{X} \times \mathcal{X} \to \mathcal{M} \cup \{\perp\})$ *is* ε-super strong non-malleable in split-state model *if for every functions* $f, g : \mathcal{X} \to \mathcal{X}$ *and for every* $m_0, m_1 \in \mathcal{M}$

$$\mathsf{SupStrTamp}^{f,g}_{m_0} \approx_\varepsilon \mathsf{SupStrTamp}^{f,g}_{m_1}$$

where

$$\mathsf{SupStrTamp}^{f,g}_{m} = \left\{ \begin{array}{c} (X, Y) \leftarrow \mathsf{Enc}(m), \\ \textit{output } \textbf{same} \textit{ if } (X, Y) = (f(X), g(Y)) \\ \textit{else if } \mathsf{Dec}(f(X), g(Y)) = \perp \textit{ output } \perp \\ \textit{else output: } (f(X), g(Y)) \end{array} \right\}$$

Examples of the codes:

- [DKO13] is not super and not strong,
- [ADL14] is super but not strong,
- all non-malleable extractors including [CGL16, Li17, Li19b] are strong but not super,
- [ADL14] compiled with [AKO17] is super and strong.

[6] Notice that outputting same in that case is unavoidable.

Informal Theorem 3.1 (from [AKO17]). *There exists a compiler that turns any super non-malleable code (with certain sampling properties which we discuss below) in* $2-$*split state model into a super-strong non-malleable code in a* $2-$*split state model, at the minimal loss to the rate of the code. The above compiler also turns a non-malleable code into a strong non-malleable code.*

The idea behind the compiler is to introduce a certain level of circularity: $\mathsf{Enc}(m||\text{check}_X, \text{check}_Y) = X, Y$ in other words codeword encodes it's own "checks". Notice that the difference between strong and not-strong variant is only in the use of same output. The checks ensure that if the code was tampered with and still decodes to the same message then the checks remain unchanged - this leads to the decoder error since the checks will not match the changed codeword.

This approach has a problem: the circularity introduced above does not necessarily allow for efficient encoding, and thus there are additional requirements on the underlying non-malleable code. The authors show that the extra assumptions are fulfilled by the code from [ADL14], thereby giving a super-strong non-malleable code.

3.7 Continuous non-malleable codes

We can push the definition further, imagine that the codeword is tampered not once, but multiple times. This is the idea behind continuous non-malleable codes. While in principle we can take any of the four variants: standard, strong, super, super-strong and extend the definition to multiple tamperings for various technical reasons[7] the super-strong extension is the one that received attention.

There are again four variants that stem from two possible flags: self-destruct (yes/no) and persistence (yes/no).

Self-destruct decides what happens when one of the tamperings outputs $\perp$- should we stop the experiment, or should we allow adversary to continue tampering? We will discuss later that non-self-destruct codes do not exist for the most of the reasonable tampering families.

Persistence(often referred to as resettability) decides how the tampering is applied. Say codeword c was tampered into c', is the next tampering applied to original c, or should it be applied on top of c'? As long as $c \to c'$ is a bijection that is not a problem, but if the tampering function was very lossy given c' we can't recreate c thus this becomes a non-trivial choice. Indeed later we will discuss impossibility results that strongly separate persistent (not-resettable) and non-persistant (resettable) codes.

Remark 3.6 (Note on two-source non-malleable extractors). *It is important to stress few things: two-source non-malleable extractors do not output* $\perp$ *thus (for the same reason why non-self-destruct codes do not exist for reasonable tampering classes) we can not consider a continuous version of them. However we can, and usually do, consider a* $t-$*times tampering variants where two-source non-malleable extractor is tampered* t *times for some fixed in advance* t*.*

Definition 3.7. (Continuous Non-Malleable Code.) *[JW15] define four types of continuous non-malleable codes based on two flags:* $\mathsf{sd} \in \{0,1\}$ (self-destruct) *and* $\mathsf{prs} \in \{0,1\}$ (persistent)*. We say that an encoding scheme* $(\mathsf{Enc} : \mathcal{M} \to \mathcal{X} \times \mathcal{X}, \mathsf{Dec} : \mathcal{X} \times \mathcal{X} \to \mathcal{M} \cup \{\perp\})$ *is* $(\mathsf{T}, \varepsilon)$-continuous $[\mathsf{sd}, \mathsf{prs}]$ non-malleable in split-state model *if for every Adversary* $\mathcal{A}$ *and for every* $m_0, m_1 \in \mathcal{M}$

$$\mathsf{ConTamper}_{\mathcal{A},\mathsf{T},m_0} \approx_\varepsilon \mathsf{ConTamper}_{\mathcal{A},\mathsf{T},m_1}$$

where $\mathsf{ConTamper}_{\mathcal{A},\mathsf{T},m} =$

[7] Main problem of non-super variants is that immediately after the first tampering X, Y are not independent anymore given $\mathsf{Dec}(f_1(X), g_1(Y))$, this causes huge technical problems, thus in practice it's actually easier to aim for the strongest variant. Intuitively speaking X, Y remain "somewhat-independent" given $f(X), g(Y)$, where by "somewhat-independent" we mean that X, Y still form a valid codeword, but revealing extra information doesn't add any additional correlations.

$$\left\{\begin{array}{l}(X,Y)\leftarrow \mathsf{Enc}(m),\\ f_0,g_0\equiv \mathsf{id},\\ \textbf{Repeat}\quad i=1,2,...,\mathsf{T}\\ \quad \mathcal{A}\text{ chooses functions } f'_i,g'_i\\ \quad \textbf{if } \mathsf{prs}=1 \textbf{ then } f_i=f'_i\circ f_{i-1},\ g_i=g'_i\circ g_{i-1}\\ \qquad \textbf{else } f_i=f'_i,\ g_i=g'_i\\ \quad \textbf{if } (f_i(X),g_i(Y))=(X,Y) \textbf{ then output } \textbf{same}\\ \qquad \textbf{else}\\ \qquad \textbf{if } \mathsf{Dec}(f_i(X),g_i(Y))=\perp \textbf{ then output } \perp \textbf{ if } \mathsf{sd}=1 \textbf{ then experiment stops}\\ \qquad\quad \textbf{else output } (f_i(X),g_i(Y)) \textbf{ if } \mathsf{prs}=1 \textbf{ then experiment stops}\end{array}\right\}$$

Remark 3.7. *In the case of persistent tampering, the above definition by [JW15] assumes that the tampering experiment stops if there is a non-trivial tampering that does not decode to $\perp$ since in this case the adversary learns the entire tampered codeword, and can simulate the remaining tampering experiment himself (since the tampering is persistent).*

Remark 3.8. *In any model allowing bitwise tampering, in particular the $2-$split state model, it is not difficult to conclude that the* non-self-destruct *property is impossible to achieve even in the case of persistent tampering if the space of messages contains at least 3 elements. To see this, notice that one can tamper the codeword $c=(c_1,c_2,c_3,\ldots)$ to obtain $c'_1=(0,c_2,\ldots)$. The adversary then obtains the output of the tampering experiment which is* same *if and only if $c_1=0$. Thus the adversary learns $c^\star_1=c_1$ and continues the tampering experiment with $(c^\star_1,0,c_3,\ldots)$ (note that this tampering is persistent). Thus, the adversary can continue learn the codeword one bit at a time, thereby learning the entire codeword in N steps where N is the length of the codeword.*

The constructions:

Theorem 3.8 (from [AKO17]). *If $(\mathsf{Enc},\mathsf{Dec})$ is an ε-super strong non-malleable code in the $2-$split-state model then $(\mathsf{Enc},\mathsf{Dec})$ is a $(T,(2T+1)\varepsilon)-$continuous self-destruct, persistent non-malleable code in the $2-$split-state model. This combined with [ADL14] compiled with [AKO17] gives an explicit and efficient continuous self-destruct persistent non-malleable code in the $2-$split-state model.*

Remark 3.9. *The number of tampering rounds T does not have to be specified in advance (unlike with two-source non-malleable extractors). We expect the number of tamperings to be polynomial, and ε to be negligible, one can plug those in and obtain a code with unlimited (but polynomial) number of tamperings and security ε^α for any $\alpha<1$.*

The idea behind the theorem is as follows: there are only two output patters that we can observe: either there will be some number of same outputs followed by a $\perp$ or followed by the tampered codeword c'. Authors argue that the long same chain doesn't teach us much thus the only tampering that really matters is the last one (the one that leads to not-same). Thus the continuous tampering is actually reduced to the one non-trivial tampering, we just have to pay a small price in epsilon, since we basically have to guess in which round the non-same tampering will happen.

Remark 3.10. *Above technique was extended and generalized by [BFM+22] for other tampering classes. In particular authors achieve continuous NMC against persistent decision tree tampering.*

Informal Theorem 3.2 (from [ADN+19b]). *There exists an explicit and efficient self-destruct, non-persistent (resettable) continuous non-malleable code in $8-$split state model (i.e. where we have 8 states instead of 2).*

Remark 3.11. *[FMNV14] show that non-persistent continuous non-malleable codes are impossible to construct in 2−split state model. We know that 8 states is enough, we hypothesize that the idea behind [ADN+19b] could be extended to give an existential (not efficient or explicit) 6 state construction. The exact number of states required to construct non-persistent code remains an opened question even in the non-explicit case.*

4 The non-malleable code construction via inner product

In this section we show a construction of non-malleable codes via inner product due to [ADL14, Agg15, AB16].

Theorem 4.1. *There exist absolute constants $c, c' > 0$ such that the following holds. For any finite field $\mathbb{F}_p$ of prime order, and any $n > c' \log^4 p$,*

$$(\mathcal{S}^2_{n,p} \Rightarrow \mathcal{AFF}_p, 2^{-cn^{1/4}}) .$$

We will prove the following theorem which immediately implies Theorem 4.1. To see this, consider the encoding function that takes as input an element of $x \in \mathbb{F}_p$ and chooses uniformly random $L, R \in \mathbb{F}_p^n$ conditioned on $\langle L, R\rangle = x$, and the decoding function $\mathsf{Dec}(\ell, r)$ is defined as $\mathsf{Dec}(\ell, r) := \langle \ell, r\rangle$.

Theorem 4.2. *There exist absolute constants $c, c' > 0$ such that the following holds. For any finite field $\mathbb{F}_p$ of prime order, and any $n > c' \log^4 p$, let L, R be random variables uniform and independent in $\mathbb{F}_p^n$, and $f, g : \mathbb{F}_p^n \to \mathbb{F}_p^n$ be arbitrary functions. Then,*

$$\Delta(\langle L, R\rangle, \langle f(L), g(R)\rangle \; ; \; U_{\mathbb{F}_p}, D(U_{\mathbb{F}_p}) ,$$

where $U_{\mathbb{F}_p}$ is uniform and independent of L, R, and D is a distribution over $\mathcal{AFF}_p$.

We need the following result that can be seen as a generalization of the linearity test from [Sam07] and that is discussed and proved in [ADL14].

Theorem 4.3. *Let p be a prime, and n be a positive integer. For any $\varepsilon = \varepsilon(n, p) > 0$, $\gamma_1 = \gamma_1(n, p) \leq 1$, $\gamma_2 = \gamma_2(n, p) \geq 1$, the following is true. For any function $f : \mathbb{F}^n \to \mathbb{F}^n$, let $\mathcal{A} \subseteq \{(x, f(x)) : x \in \mathbb{F}^n\} \subseteq \mathbb{F}^{2n}$. If $|\mathcal{A}| \geq \gamma_1 \cdot |\mathbb{F}^n|$ and there exists some set $\mathcal{B}$ such that $|\mathcal{B}| \leq \gamma_2 \cdot p^n$, and*

$$\Pr_{a,a' \in \mathcal{A}}[a - a' \in \mathcal{B}] \geq \varepsilon,$$

then there exists a linear map $M : \mathbb{F}^n \to \mathbb{F}^n$ such that

$$\Pr_{(x,f(x)) \in \mathcal{A}}[f(x) = Mx] \geq p^{-O(\log^6(\frac{\gamma_2}{\gamma_1 \varepsilon}))} .$$

4.1 Proof sketch of Theorem 4.2

The following lemma shows that for any large enough subdomain of $\mathbb{F}^n \times \mathbb{F}^n$ for which $\langle L, R\rangle, \langle f(L), g(R)\rangle$ is not close to the desired distribution for some D, there exists a large enough subdomain on which f is linear.

Lemma 4.1. *Let p be a prime, n a positive integer, and $0 < t < n$. Let $\mathcal{L} \subseteq \mathbb{F}_p^n$ such that $|\mathcal{L}| \geq p^{n-t}$. Let L, R be independent random variables uniformly distributed in $\mathcal{L}$ and $\mathbb{F}_p^n$, respectively. Then, either there exists a distribution G over $\mathcal{AFF}_p$ such that*

$$\Delta(\langle L, R\rangle, \langle f(L), g(R)\rangle \; ; \; U_{\mathbb{F}_p}, D(U_{\mathbb{F}_p}) \leq p^{-t} ,$$

or there exists a subset $|\mathcal{L}' \subseteq \mathcal{L}|$, a linear map $M \in (\mathbb{F}_p^n)^{\mathbb{F}_p^n}$, and a constant C such that $|\mathcal{L}'| \geq |\mathcal{L}| \cdot p^{-Ct^4 \log^4 p}$, and $f(x) = Mx$ for all $x \in \mathcal{L}'$.

Proof. We assume that

$$\Delta(\langle L, R\rangle, \langle f(L), g(R)\rangle \; ; \; U_{\mathbb{F}_p}, D(U_{\mathbb{F}_p}) > p^{-t} \; ,$$

as otherwise the result trivially holds. Then, by Lemma 2.7 there exist $a \in \mathbb{F}$ such that $\Delta(\langle L, R\rangle + a\langle f(L), g(R)\rangle \; ; \; U_{\mathbb{F}_p}) \geq p^{-t-2}$. Define functions $F, G : \mathbb{F}^n \to \mathbb{F}^{2n}$ as follows

$$F(x) = (x, f(x)), \quad G(y) = (y, ag(y)).$$

We have that $\Delta(\langle F(L), G(R)\rangle \; ; \; U_{\mathbb{F}_p}) \geq p^{-t-2}$. Applying Lemma 2.8, we get that for (L', R') i.i.d to (L, R) we have

$$\Pr[\langle F(L'), G(R')\rangle = \langle F(L), G(R)\rangle] \geq \frac{1}{p} + \frac{1}{p^{2t+5}}.$$

Applying Lemma 2.9 with $X = F(L), Y = G(R), X' = F(L'), Y' = G(R')$ we get that

$$\Pr[\langle F(L) - F(L'), G(R)\rangle = 0] \geq \frac{1}{p} + \frac{1}{p^{2t+5}}.$$

Define

$$\mathcal{B} := \left\{ \alpha \in \mathbb{F}_p^{2n} : \Pr[\langle \alpha, G(R)\rangle = 0] \geq \frac{1}{p} + \frac{1}{p^{2t+6}} \right\}.$$

Let $B \in \mathcal{B}$ be uniform. Then $\Delta(\langle B, G(R)\rangle, U_{\mathbb{F}_p}) \geq \frac{1}{p^{2t+6}}$. Also, since R is uniform in $\mathbb{F}_p^n$, $G(R)$ has min-entropy $n \log p$. Hence, by Lemma 2.2, we have $\mathbf{H}_\infty(B) \leq (n+4t+13) \cdot \log p$, which implies $|\mathcal{B}| \leq p^{n+4t+13}$. Furthermore, we have that

$$\Pr[\langle F(L') - F(L''), G(R')\rangle = 0] \leq \Pr[F(L') - F(L'') \in \mathcal{B}] + \frac{1}{p} + \frac{1}{p^{2t+6}}.$$

So we must have that

$$\Pr[F(L') - F(L'') \in \mathcal{B}] \geq \frac{1}{p^{2t+5}} - \frac{1}{p^{2t+6}} \geq \frac{1}{p^{2t+6}}.$$

Thus, using Theorem 4.3, we get that there exists a linear map $M : \mathbb{F}^n \to \mathbb{F}^n$ for which

$$\Pr_{x \in \mathbb{F}^n}[Mx = f(x)] \geq p^{-O(t^4 \log^4 p)} \; .$$

□

We now show that if f is linear, then we get the desired distribution.

Lemma 4.2. *Let p be a prime, n a positive integer, and $0 < s < n$. Let $\mathcal{L}' \subseteq \mathbb{F}_p^n$ such that $|\mathcal{L}'| \geq p^{n-s}$, and $f(x) = Mx$ for all $x \in \mathbb{F}_p^n$, where M is a linear map in $(\mathbb{F}_p^n)^{\mathbb{F}_p^n}$. Let L, R be independent random variables uniformly distributed in $\mathcal{L}'$ and $\mathbb{F}_p^n$, respectively. Then, there exists a distribution G over $\mathcal{AFF}_p$ such that*

$$\Delta(\langle L, R\rangle, \langle f(L), g(R)\rangle \; ; \; U_{\mathbb{F}_p}, G(U_{\mathbb{F}_p}) \leq p^{-s} \; .$$

Proof. We will prove that the given statistical distance is small for almost all fixing of $R \in \mathbb{F}_p^n$, and hence conclude the desired result.

Let $\mathcal{S}$ be the set of all $x \in \mathbb{F}_p^n$ such that $\Delta(\langle L, s\rangle, U_{\mathbb{F}_p}) > p^{-3-s}$. By Lemma 2.2, $|\mathcal{S}| \leq p^{3s+8}$.

Note that

$$\langle f(L), g(r)\rangle = \langle ML, g(r)\rangle = \langle L, M^T g(r)\rangle \; .$$

So, without loss of generality, we assume M to be the identity function, and replace g by $M^T g$. Assume $(\langle L, r\rangle, \langle f(L), g(r)\rangle)$ is not p^{-s-1}-close to $U_{\mathbb{F}_p}, G(U_{\mathbb{F}_p})$ for any G distributed over $\mathcal{AFF}_p$. This means

that for any $a \in \mathbb{F}_p$, $(\langle L, r\rangle, \langle f(L), g(r)\rangle)$ is not p^{-s-1}-close to $U_{\mathbb{F}_p}, aU_{\mathbb{F}_p} + B$, for some random variable B independent of $U_{\mathbb{F}_p}$. By Lemma 2.7, for every $a \in \mathbb{F}_p$, there exists $b \in \mathbb{F}_p$ such that

$$\Delta(\langle L, r + b(g(r) - ar)\rangle \; ; \; U_{\mathbb{F}_p}) \geq p^{-3-s} \; . \tag{2}$$

We will show that this implies that $r \in \mathbb{F}_p S + \mathbb{F}_p S$.

$$r \in \{\alpha_1 x_1 + \alpha_2 x_2 \mid \alpha_1, \alpha_2 \in \mathbb{F}_p, x_1, x_2 \in \mathcal{S}\} \; .$$

By equation 2 with $a = 0$, there is some $b = b^*$ such that $r + b^* g(r) = x_1 \in \mathcal{S}$. We assume without loss of generality that $b^* \neq 0$ since if $b^* = 0$, then $r \in \mathcal{S}$, and the desired statement is true.

Letting $a = -1/b^*$, we have that there exists b such that $r(1 + b/b^*) + bg(r) = x_2 \in \mathcal{S}$.

Combining, we get that $r = (bx_1 - bx_2)/b^*$, thereby proving that $r \in \mathbb{F}_p S + \mathbb{F}_p S$. Thus, by Lemma 2.6, the desired statistical distance is at most

$$\frac{p^{6s+20}}{p^n} \cdot 1 + p^{-1-s} \leq p^{-s} \; .$$

□

Finishing the proof sketch. By Lemma 4.1, whenever $\langle L, R\rangle, \langle f(L), g(R)\rangle$ are not close to $U, D(U)$ for some affine function D, we can always find a large subset of the domain on which f is linear, and thus, using Lemma 4.2, we get that on this subset, $\langle L, R\rangle, \langle f(L), g(R)\rangle$ is close to $U, D(U)$ for some D. We thus continue to find a large subset of the domain on which $\langle L, R\rangle, \langle f(L), g(R)\rangle$ is close to $U, D(U)$ until we are left with a very small fraction of the entire domain. The result then follows from Lemma 2.6.

4.2 Non-malleable codes against affine tampering in $\mathbb{F}_p$

To complete the construction of non-malleable codes in the split-state model, we need non-malleable codes against affine tampering in $\mathbb{F}_p$. The following was shown in [Agg15].

Theorem 4.4. *For any integer $k > 0$, and $p > 2^{4k}$,*

$$(\mathcal{AFF}_p \Rightarrow \mathsf{NM}_k, 2^{-\Omega(k)}) \; .$$

The construction for this is quite simple. A so called affine-evasive subset S of $\mathbb{F}_p$ of size significantly larger than 2^k was constructed with the property that for any fixed $(a, b) \in \mathbb{F}_p \times \mathbb{F}_p \setminus \{1, 0\}$, we have that $|aS + b \cap S| \ll |S|$. The set S is then partitioned into $K = 2^k$ subsets $S_1, \ldots, S_K$, and the encoding of the i-th message is a uniformly random element of S_i, and all elements in $\mathbb{F}_p$ not in S decode to a special symbol $\perp$.

5 The non-malleable codes via two-source non-malleable extractors

Towards the goal of constructing non-malleable codes, Cheraghchi and Guruswami [CG14b] introduced non-malleable extractors as a stronger primitive that immediately yields efficient non-malleable codes as long as the preimage of the extractor is efficiently samplable. Informally, a non-malleable two-source extractor nmExt guarantees that for any independent random sources X, Y, and any functions f, g with at least one of them having no fixed points, $\mathsf{nmExt}(X, Y)$ is indistinguishable from uniform even given $\mathsf{nmExt}(f(X), g(Y))$.[8] It is easy to see that a non-malleable two-source extractor gives non-malleability for

[8] We say that the extractor is a strong non-malleable two-source extractor if for any independent random sources X, Y, and any functions f, g with at least one of them having no fixed points, $\mathsf{nmExt}(X, Y)$ is indistinguishable from uniform even given $\mathsf{nmExt}(f(X), g(Y))$ and Y.

a uniformly random message (average-case security) while a non-malleable code achieves non-malleability for every message (worst-case security). A non-malleable two-source extractor can be transformed into a non-malleable code $(\mathsf{Enc}, \mathsf{Dec})$ by setting $\mathsf{Enc}(m) := \mathsf{nmExt}^{-1}(m)$, and $\mathsf{Dec}(x, y) := \mathsf{nmExt}(x, y)$.

NME to NMC: Limitations. We note that the transformation from $\mathsf{nm2Ext}$ to NMCs requires arguing worst-case security from average-case security, which incurs a factor $2^{|\mathsf{message\ size}|}$ penalty in the security parameter. Most results on building 2-split-state NMCs have focused on improving the rate of non-malleable two-source extractors and relied on this *lossy* transformation to build NMCs.

Since their conception, the non-malleable two-source extractors went a long way and found independent applications, from the network extraction [GSZ21], to variants of privacy amplification [AOR+22]. More importantly, we know numerous connections and reductions between two-source extractors, seeded non-malleable extractors and two-source non-malleable extractors (see [CGL16, Li17, BCD+18, AOR+22]). This gives us hope that further progress in the constructions of these objects might give us an explicit two-source extractor with a negligible error and a low entropy requirements for both sources.

6 Rate amplification techniques

Another useful technique towards improving the rate of NMC constructions is *rate amplification* or *bootstrapping*. It's a recurring theme in cryptography to combine a scheme with a very strong security but bad efficiency with a scheme with bad security but a great efficiency in such a way that resulting scheme inherits the best of both worlds: good security and efficiency.

In the context of non-malleable codes it was first used by [AGM+15]. Authors achieved a rate 1 non-malleable code against bitwise tampering and permutations by combining the rate 0 scheme (from [AGM+14]) with an error correcting secret sharing scheme (that has no non-malleability guarantee). In the context of 2 split state tampering it was used by [KOS17, KOS18, AO20, ASK+22].

The abstract idea is to use an efficient code to encode the message, while the bad rate code will encode tags and checks independent of the message's size. What is left is to argue that those tags will guarantee the security of the construction.

In the remainder of this section we will dive deeper into the construction of [KOS17, KOS18, ASK+22]. The latter paper achieves current state of the art rate of $\frac{1}{3}$, but it strongly builds on the construction of the former paper, thus we can't discuss one without the other.

6.1 Technical Overview of [KOS18]

This paper does not build non-malleable code but it forms a crucial building block for the construction of [ASK+22].

Informal Theorem 6.1 (Main Result of [KOS18]). *There exists an efficient, information theoretically secure non-malleable randomness encoder with rate arbitrarily close to $\frac{1}{2}$, and the negligible error.*

Kanukurthi, Obbattu and Sekar [KOS18] introduced the notion of *non-malleable randomness encoders* (NMRE). Similar to a 2-split-state NMC, a 2-split-state NMRE consists of two independently tamperable states L and R. Contrary to an NMC, where the encoder encodes arbitrary messages, an NMRE's encoder outputs L and R such that they decode to a random string, and herein lies all the difference: as we have already discussed in the context of non-malleable extractors, it might not be possible to efficiently find the preimage of the specific message, or the security parameter might be too small to allow for fixing the specific message in a blackbox way.

While the problem of building high-rate NMCs has eluded researchers for over a decade, we know how to build NMREs with rate $\frac{1}{2}$ (see [KOS18]). At the same time, we emphasize that obtaining a high-rate NMC (instead of an NMRE) is critical for many applications (such as non-malleable commitments.)

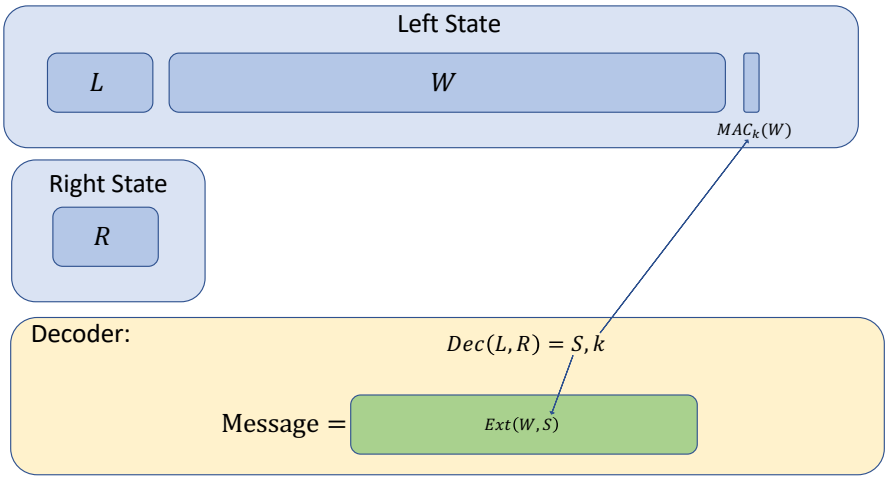

Figure 1: Construction of the non-malleable randomness encoder by [KOS18].

Informally, the NMRE of [KOS18] (see Figure 1.) picks the source w and seed s to a strong seeded extractor (Ext) as well as a key k to a message authentication code (MAC). The code consists of two states, left: $\ell \| w \| \sigma_w$, and right: r, where ℓ, r are an encoding of s, k with any low-rate augmented non-malleable code, and σ_w is a tag evaluated on w with k as the key. The codeword, if valid, decodes to $\mathsf{Ext}(w; s)$.

To denote the tampering of a variable x we will use the $\underline{x}$ notation. The security proof can be split into an analysis of the following three cases:

- $\mathcal{P}_{id} = \{(\ell, w, \sigma_w, r) : (w, \sigma_w) = (\underline{w}, \underline{\sigma_w}) \wedge \mathsf{NMDec}(\ell, r) = \mathsf{NMDec}(\underline{\ell}, \underline{r})\}$. This partition corresponds to the adversary not tampering with the codeword. In this case, the codeword will decode to the same message.

- $\mathcal{P}_{tag} = \{(\ell, w, \sigma_w, r) : (w, \sigma_w) \neq (\underline{w}, \underline{\sigma_w}) \wedge \mathsf{NMDec}(\ell, r) = \mathsf{NMDec}(\underline{\ell}, \underline{r})\}$. This partition corresponds to the case where $s = \underline{s}$ and $k = \underline{k}$. Since the MAC's key remains secure and hidden from the adversary, the codeword will decode to $\perp$ with high probability via the security of the message authentication codes.

- $\mathcal{P}_{rest} = \{(\ell, w, \sigma_w, r) : \mathsf{NMDec}(\ell, r) \neq \mathsf{NMDec}(\underline{\ell}, \underline{r})\}$. Finally, this partition represents the case when adversary did apply non-trivial tampering to ℓ, r. By the properties of non-malleable code, if the codeword falls into this partition (and the likelihood of falling into this partition is not too small) $\underline{S}$ is independent of S (even given L).

  Now we will proceed with the following trick: we will reveal $\underline{S}$ and L, since $\underline{W}$ is now a function of W only[9] we get that $\mathbf{H}_\infty(W|\mathsf{Ext}(\underline{W}, \underline{S}), \underline{S}, L) \geq |W| - |\mathsf{Ext}| - |\underline{S}|$, where $|\underline{S}|$ penalty comes since $\underline{L}$ might have depended on W and thus $\underline{S}$ might depend on W.

  This is a spot where we need augmented property as S remains uniform and independent of W, $\underline{S}$ and L. Thus, as long as $\mathbf{H}_\infty(W|\mathsf{Ext}(\underline{W}, \underline{S}), \underline{S}, L) > |\mathsf{Ext}|$, we will obtain that $\mathsf{Ext}(W, S)$ is uniform given $\mathsf{Ext}(\underline{W}, \underline{S})$. This means that the original message remains uniform given the message after tampering. The only thing to ensure is that $|W| - |\mathsf{Ext}| - |\underline{S}| > |\mathsf{Ext}|$. Since the size of S is small, we roughly get that $|W| > 2|\mathsf{Ext}|$ which leads to the rate $\frac{1}{2}$.

In order to extend this construction to encode an arbitrary message m, one option would be to reverse sample w and s such that $\mathsf{Ext}(w; s) = m$. Unfortunately, this won't work because, on the one hand, we

[9] We can ignore tag σ_w as a tiny leakage, alternatively the tag can be moved inside the non-malleable encoding.

require the seed s to be short (as it is encoded using a poor-rate NMC) and, on the other hand, given a source w, there will be at most $2^{|s|}$ possible messages that could have been encoded. Thus adversary tampering with w will likely be able to distinguish between two messages of his choice (since only for one of them there will exist s_i such that $\mathsf{Ext}(w, s_i) = m_i$). In other words, to obtain any meaningful security, s needs to be as long as the message. However, if s is long, the above approach will not yield an improvement in the rate.

6.2 Technical Overview of [ASK+22]

Theorem 6.1 (Main Result of [ASK+22]). *There exists an efficient, information-theoretically secure ε-right-augmented[10] non-malleable code in the 2-split-state model with rate 1/3. Authors give two instantiations of the scheme: the first gives a strikingly simple construction and achieves an error of $2^{-\Omega(\kappa^{1/5}/\operatorname{polylog}(\kappa))}$; the second instantiation loses out on the simplicity but achieves an error of $\varepsilon = 2^{-\Omega(\frac{\kappa}{\log^3 \kappa})}$, where κ is the size of the message.*

As we discussed earlier, fixing a specific message in the scheme of [KOS18] is not possible. The idea is to add extra information to the right state that will allow for fixing a specific message. The construction described in Figure 2 goes as follows: as before we will pick random w, s then we will fix $c = \mathsf{Ext}(w, s) \oplus m$, and after that we will pick two random keys k_c, k_w and encode using a non malleable code: $\mathsf{Enc}(s, k_c, k_w) = \ell, r$. Finally, we calculate σ_w a MAC of w under key k_w and σ_c a tag of c under key k_c. The encoding is left state: $(\ell||w||\sigma_w)$ and right state: $(r||c||\sigma_c)$.

As a side note, we mention that the encoding scheme is identical to that due to Kanukurthi, Obbattu and Sekar [KOS17]. While [KOS17] gave a four-state construction, [ASK+22] merged states to obtain a two-state construction.
We now offer an overview of the proof.

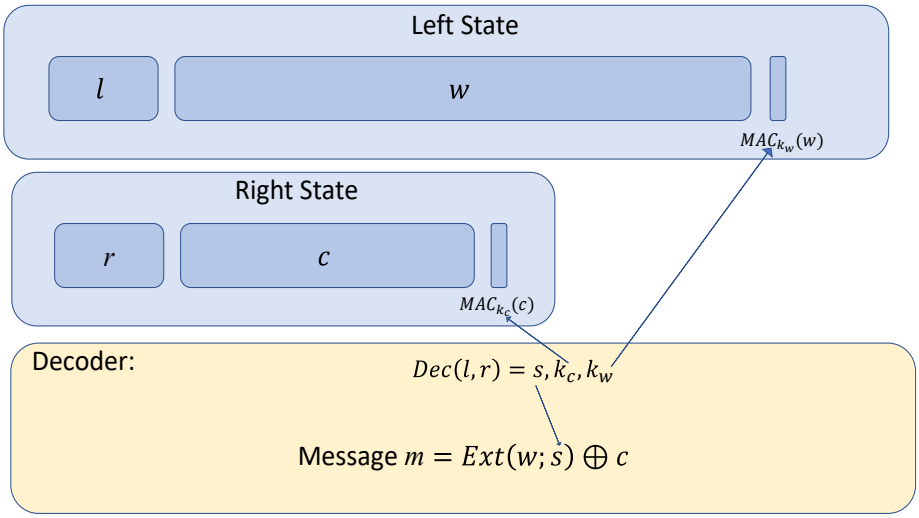

Figure 2: Overview of the construction from [ASK+22]. Blocks ℓ, r come from augmented non-malleable code. The encoder proceeds in steps: first, we randomly sample s, k_w, k_c, w (all independently of the message we are encoding), then we encode s, k_w, k_c using NMC into ℓ, r. We then set $c = \mathsf{Ext}(w; s) \oplus m$, and evaluate σ_c as a MAC tag of c on key k_c, and σ_w as MAC tag of w on key k_w.

This construction uses the following building blocks: a message authentication code, a strong seeded extractor, and a low-rate non-malleable code which we shall use to encode the keys of the message

[10] Right-augmented property guarantees that the right state of the NMC is simulatable independent of the message, along with the tampered message.

authentication code and the seed for the seeded extractor. Also, for a variable X, $\underline{X}$ will denote its tampering. We proceed with a slightly simplified sketch of the proof.

Proof Overview The proof proceeds by partitioning the codeword space. We describe the partitions below:

- $\mathcal{P}_1 = \{(\ell, w, \sigma_w, r, c, \sigma_c) : (w, \sigma_w, c, \sigma_c) = (\underline{w}, \underline{\sigma_w}, \underline{c}, \underline{\sigma_c}) \wedge \mathsf{NMDec}(\ell, r) = \mathsf{NMDec}(\underline{\ell}, \underline{r})\}$. This partition captures the scenario when, even after tampering, the inner codeword (ℓ, r) decodes to the same message, and W, σ_w, C, σ_c remain unchanged. In this case, the final codeword must decode to the same message.
- $\mathcal{P}_2 = \{(\ell, w, \sigma_w, r, c, \sigma_c) : (w, \sigma_w, c, \sigma_c) \neq (\underline{w}, \underline{\sigma_w}, \underline{c}, \underline{\sigma_c}) \wedge \mathsf{NMDec}(\ell, r) = \mathsf{NMDec}(\underline{\ell}, \underline{r})\}$. $\mathcal{P}_2$ captures the scenario when, the decoding of the inner code remains unchanged after tampering, while one of the pairs (W, σ_w) or (C, σ_c) are changed. If this event occurs then, using the security of MACs, the tampering is detected with overwhelming probability.
- $\mathcal{P}_3 = \{(\ell, w, \sigma_w, r, c, \sigma_c) : \mathsf{NMDec}(\ell, r) \neq \mathsf{NMDec}(\underline{\ell}, \underline{r})\}$. $\mathcal{P}_3$ captures the scenario that the inner code is non-trivially tampered and does not decode to $\mathsf{NMDec}(L, R)$. Authors show that the tampered codeword is independent of the original message m. This is the most interesting case.

In order to prove non-malleability, we need to demonstrate the existence of a simulator whose outputs is indistinguishable from the output of the tampering experiment. The simulator doesn't use the message; however, it outputs a special symbol same to indicate that the tampered message is unchanged. The simulator's output is run through a special wrapper function (typically called "Copy" function) that, in this case, outputs the original message.

The simulator generates the codeword $((L, W, \sigma_w), (R, \widetilde{C}, \widetilde{\sigma}_c))$ of a random message. If this simulated codeword is in $\mathcal{P}_1$, it outputs same. Recall that the wrapper function will then output the original message. If the simulated codeword is in $\mathcal{P}_2$, the simulator outputs $\perp$, else the simulator outputs $\mathsf{Dec}\Big((L, W, \sigma_w), (R, \widetilde{C}, \widetilde{\sigma}_c)\Big)$. (Note that the code is right-augmented i.e., it satisfies a stronger notion of security where the right state of the codeword can be revealed without breaking non-malleability.)

To prove non-malleability, we need to show that this behaviour of the simulator is indistinguishable from that of the tampering experiment. To do this, we first need to argue that the probability of a codeword being in any given partition is independent of the message. Authors do it by showing how to determine a partition given small leakages from left and right state and then arguing that those small leakages can't leak the encoded message, and thus the probability of falling into each partition can not depend on the encoded message[11]. Next, authors show that the output of the tampering experiment is, in each case, indistinguishable from the simulator's output.

For the case where the codeword is in partition $\mathcal{P}_1$, it is clear that the simulator output is identical to that of the tampering experiment. We, therefore, focus on the other two cases.

6.2.1 Codeword is in $\mathcal{P}_2$ i.e., $\mathsf{NMDec}(\underline{L}, \underline{R}) = \mathsf{NMDec}(L, R)$.

Intuitively, we would like to argue that the tag keys K_w, K_c will remain securely hidden from the adversary, and if he decides to tamper with W or C he will not be able to fake tags σ_w, σ_c. Thus either the whole codeword remains untampered (in which case, we are in $\mathcal{P}_1$) or the new codeword will not be valid.

The standard approach would be to argue that if $\Pr(\mathsf{NMDec}(\underline{L}, \underline{R}) = \mathsf{NMDec}(L, R))$ is not too small then

$$\Pr(\text{tampered codeword is valid } \wedge (\underline{W}, \underline{C}) \neq (W, C) \mid \mathsf{NMDec}(\underline{L}, \underline{R}) = \mathsf{NMDec}(L, R))$$

[11] This proof relies on the secret sharing property of the non-malleable code as well as the security of the strong randomness extractor.

is negligible. However we have to be delicate here. For example if adversary wants to tamper with W he has access to L and knows that $\mathsf{NMDec}(\underline{L}, \underline{R}) = \mathsf{NMDec}(L, R)$. This reveals some information about R and thus adversary potentially might get hold of some partial information about the encoded data (and K_w, K_c in particular). This is why it is actually easier to directly argue that

$$\Pr(\text{tampered codeword is valid } \wedge (\underline{W}, \underline{C}) \neq (W, C) \wedge \mathsf{NMDec}(\underline{L}, \underline{R}) = \mathsf{NMDec}(L, R)) \tag{3}$$

is negligible. Notice that the codeword will not be valid in only one of three cases: if $\mathsf{NMDec}(\underline{L}, \underline{R}) = \perp$ or if one of the MACs on W or C does not verify correctly. Since $\mathsf{NMDec}(\underline{L}, \underline{R}) = \mathsf{NMDec}(L, R)$ we know that the only options left are the failures to verify MACs. Moreover we know that $(\underline{K_w}, \underline{K_c}) = (K_w, K_c)$, thus Inequality 3 can be rewritten:

$$\Pr(\mathsf{Vrfy}_{K_w}(W, \sigma_w) = \mathsf{Vrfy}_{K_c}(C, \sigma_c) = 1 \wedge (\underline{W}, \underline{C}) \neq (W, C) \wedge \mathsf{NMDec}(\underline{L}, \underline{R}) = \mathsf{NMDec}(L, R)) \tag{4}$$

is negligible. Now we can upper-bound the term in the Inequality 4 by the following

$$\Pr(\mathsf{Vrfy}_{K_w}(W, \sigma_w) = \mathsf{Vrfy}_{K_c}(C, \sigma_c) = 1 \wedge (\underline{W}, \underline{C}) \neq (W, C)).$$

Which by the union bound can be upper-bounded with

$$\Pr(\mathsf{Vrfy}_{K_w}(W, \sigma_w) = 1 \wedge \underline{W} \neq W) + \Pr(\mathsf{Vrfy}_{K_c}(C, \sigma_c) = 1 \wedge \underline{C} \neq C).$$

Finally, we can argue that each of the elements of the sum is negligible. Notice that when tampering with W adversary has access to L but that can not reveal any information about K_w since every non-malleable code is a secret sharing scheme. The rest follows from the security of MACs.

6.2.2 Codeword is in $\mathcal{P}_3$ i.e., $\mathsf{NMDec}(\underline{L}, \underline{R}) \neq \mathsf{NMDec}(L, R)$.

In this case, we will follow the adventures of the seed S; the MACs and keys do not play any role here. In fact, for the purposes of this proof sketch, we will ignore the MAC keys and tags. We will also assume that this case (i.e., codeword $\in \mathcal{P}_3$) occurs with substantial probability (else we don't have to worry about it). In such a case, we will argue that the final message is independent of the original message.

We start with replacing C (see figure 2) with $\widetilde{C}$ where $\widetilde{C}$ is completely uniform and independent of the message (eventually we would need to replace $\widetilde{C}$ back with $C = \mathsf{Ext}(W, S) \oplus m$).

After technical transformations authors obtain that:

$$\mathsf{Ext}(W; S) \approx U | S, L, \widetilde{C}, \underline{L}, \mathsf{Ext}(\underline{W}; \underline{S}), \underline{S} \tag{5}$$

The intuition behind the equation above is very similar to the case of $\mathcal{P}_{rest}$ in the section 6.1, the proof is more involved than the one in [KOS18], but we omit the technical details behind the equation 5. Also, note that in the equation above, there is no dependence on m on either side as $\widetilde{C}$ is independent of m. Ultimately, we would like to say that the output of the tampering experiment is indistinguishable from the simulated output. Authors accomplish this in three steps:

1. Adding R. In equation 5, the only information correlated to W and R is $\underline{S}$. Since $\mathsf{Ext}(W; S) \approx \mathcal{U}$ even given $\underline{S}$, we can safely add R to Equation 5.

$$\mathsf{Ext}(W; S) \approx U | S, L, \widetilde{C}, \underline{L}, \mathsf{Ext}(\underline{W}; \underline{S}), \underline{S}, R, g_1(R, \widetilde{C}, \widetilde{\sigma}_c) \ .$$

From here, we would ideally like to drop $\widetilde{C}$ and somehow bring back the dependence on m via C. For now, we drop $\widetilde{C}$

$$\mathsf{Ext}(W; S) \approx U | S, L, \underline{L}, \mathsf{Ext}(\underline{W}; \underline{S}), \underline{S}, R, \underline{R} \ . \tag{6}$$

The way we'll bring C is to condition $\widetilde{C}$ on being a "cipher of m". For that, we first need to prove that $\widetilde{C}$ is independent of W given appropriate auxiliary information.

2. Capturing $\widetilde{C}$'s correlation with W. In this step, authors prove that $\widetilde{C}$ is independent of W given $S, L, \underline{L}, \mathsf{Ext}(\underline{W}; \underline{S}), \underline{S}, R, \underline{R}$. We first observe that $\widetilde{C}$ is independent of W given (L, R, S). Now we would like to add the other random variables in the auxiliary information. Authors use a Lemma due to Dziembowski and Pietrzak which states that independence in the presence of additional auxiliary information is indeed possible, provided it satisfies a few properties:

- The auxiliary information may be computed in multiple steps.
- Computation in all of the steps can use (L, R, S) and the part of the auxiliary information generated in previous steps.
- Computation in a given step can either depend on $\widetilde{C}$ or W but not both.

By computing auxiliary information in the order $\underline{L}$ followed by $\underline{R}$ followed by $\mathsf{Ext}(\underline{W}; \underline{S})$, one can easily prove that $\widetilde{C}$ is independent of W given $S, L, \underline{L}, \mathsf{Ext}(\underline{W}; \underline{S}), \underline{S}, R, \underline{R}$.

3. Conditioning $\widetilde{C}$ appropriately Since W is independent of $\widetilde{C}$ given appropriate auxilliary information, in Equation 6, we can condition $\widetilde{C}$ to either be $m \oplus \mathsf{Ext}(W, S)$ or $m \oplus U$. (Note that the former is identical to C.) By doing so, Equation 6 will lead to the following $C, S, L, \underline{L}, \mathsf{Ext}(W; S), \underline{S}, R, \underline{R} \approx \mathcal{U}, S, L, \underline{L}, \mathsf{Ext}(W; S), \underline{S}, R, \underline{R}$, where $\underline{R}, \underline{S}$ are appropriately computed.
The desired result follows by observing that the tampered codeword is a function of

$$\underline{L}, \underline{R}, \mathsf{Ext}(\underline{W}; \underline{S}), C, R \; .$$

Putting it Together. So far, we've described the simulator and sketched the proof for showing that the simulated output is indistinguishable from the tampered output in each of the cases. To complete the proof, we need to combine all three cases and, in particular, the probability that the codewords (tampered vs simulated) lie in each of the partitions needs to be analysed.

To do this, authors follow a standard argument: they consider a "skewed" codeword which, like the tampered codeword, encodes the real message. However the probability with which the skewed codeword lie in various partitions are the same as for the simulated codewords (in other words "skewed" codeword behaves like original codeword on each partition, but partitions are "assembled" with slightly modified probabilities). Authors complete the proof by showing that the probability that the tampered codeword lies in a partition is independent of the message and then combine all three cases using the skewed codeword as a intermediate hybrid.

This allows authors to finish the argument about tampered message not revealing the original message.

Candidate Instatiation. While one can turn any augmented non-malleable code (or randomness encoder) into a good rate non-malleable code, a very simple result can be obtained using [ADL14]. To encode a message m all we will need is *an affine evasive function* h. It is a function $h : \mathbb{F}_p \to \mathcal{M} \cup \perp$ such that $\Pr(h(aU + b) \neq \perp \mid h(U) = m)$ is negligible for all a, b, m, and $U|h(U) = m$ should be efficiently samplable, the construction of the said function can be found in [ADL14, Agg15]. The encoding procedure is described in Figure 3.

Short and Simple: The Encoding Procedure:

1. Sample s, k_w, k_c, w uniformly at random.
2. Sample x uniformly random, such that $h(x) = s, k_w, k_c$.
3. Sample $\ell, r \in \mathbb{F}_p^n$ uniformly random, such that $\langle \ell, r \rangle = x$.
4. Evaluate $c = \mathsf{Ext}_2(w; s) \oplus m$.
5. Calculate MACs $\sigma_c = \mathsf{Tag}_{k_c}(c)$ and $\sigma_w = \mathsf{Tag}'_{k_w}(w)$.

The final output is: on the left: ℓ, w, σ_w, and on the right: r, c, σ_c.

Figure 3: Simple non-malleable code with a great rate. Here h is an affine evasive function. The decoding procedure is analogous: the decoder inverts Steps 3 and 2, obtains keys k_w, k_c, verifies MACs from the Step 5 and proceeds to obtain the message via the Step 4. If in Step 2 the function h outputs $\perp$, then the decoder aborts and outputs $\perp$.

7 Application: Non-Malleable Codes for Computable Tampering

Split-state tampering functions, even when allowed leakage between the states, are subject to strong independence constraints. In this section, we will look at tampering families without any such constraints but instead having limited *computational complexity*. In fact, we will show, in some sense, how to reduce computational constraints to independence by showing how to construct non-malleable codes for a variety natural computational tampering classes from split state non-malleable codes. We will consider the following tampering classes:

- Decision tree tampering (Section 7.1 [BGW19]): each tampered output symbol is a function of a small polynomial number of (adaptively chosen) queries to codeword symbols.
- Small-depth circuit tampering (Section 7.2 [CL17, BDSG+18, BGW19]): the tampered codeword is produced by a boolean circuit of polynomial size and nearly logarithmic depth.
- (Bounded) Polynomial-size circuit tampering (Section 7.3 [BDL21]): the tampered codeword is produced by circuit of bounded polynomial size, n^d for some constant d where n is the codeword length.

On computational complexity and non-malleable codes. We begin with some remarks connecting non-malleable codes with more conventional computational complexity. First, we note that non-malleable codes for circuit classes require circuit lower bounds.

Proposition 7.1 (Informal). *For most natural tampering classes, $\mathcal{C}$, an explicit non-malleable code resilient to tampering by class $\mathcal{C}$ implies a circuit lower bound for that class: an explicit function that is hard for $\mathcal{C}$ to compute.*

In particular, if $(\mathsf{Enc}, \mathsf{Dec})$ is a non-malleable code resilient to $\mathcal{C}$ tampering, then Dec cannot be computed by $\mathcal{C}$. Suppose not, then consider the tampering function that computes Dec and outputs a fixed encoding 0 if the first bit of the message is 1 and outputs a fixed encoding of 1 otherwise. Moreover,

it is not difficult to observe that Enc gives rise to (efficiently samplable) input distributions against which Dec is *hard-on-average* for $\mathcal{C}$ to compute.

Given our difficulties in proving circuit lower bounds, one interpretation of this observation is that we can only expect to construct *unconditionally secure* non-malleable codes against very limited circuit classes. Or in other words, non-malleable codes for expressive circuit classes, such as polynomial size circuits, require computational assumptions.

Given that (strong) circuit lower bounds are necessary for non-malleable codes, one might wonder if they are sufficient. In general, this is not true.

Theorem 7.1 (Informal [BDKM20]). *Explicit hard functions for a class $\mathcal{C}$, do not imply non-malleable codes for $\mathcal{C}$.*

Consider the class of tampering functions, $\mathsf{Local}^{n-1} = \{f\}$, such that each output bit is an arbitrary function of all but 1 of the input bits, i.e. for each $j \in [n]$ the function computing the jth tampered bit, f_j can be written as $f_j(c_1, c_2, \ldots, c_{i_j-1}, c_{i_j+1}, \ldots, c_n)$ for some $i_j \in [n]$. It is easy to observe that such functions cannot compute Parity, i.e $\oplus_i c_i$.[12] In fact, functions in Local^{n-1} have *no* advantage over random guessing computing Parity of uniformly random inputs.

One might hope to use the fact that Parity is hard for this class, Local^{n-1}, directly by encoding a single bit b as uniformly bits $c_1, \ldots, c_n$ such that $\oplus_i c_i = b$. However, note that this code, while providing some form of leakage-resilience, is trivially malleable by the class: consider the function that flips the first bit.

This straw man argument intuitively leads us to believe that non-malleability requires much more than (average-case) circuit lower bounds. [BDKM20] justified this intuition, proving that non-malleable codes for Local^{n-1} tampering *do not exist.*

Key idea: communication bottlenecks. We saw that the straw man approach of encoding directly using a hard function for a computational tampering class will not succeed. Instead, we show how to leverage split-state non-malleable codes to construct non-malleable codes against computational tampering classes. The high level intuition for all of these constructions is to induce and exploit *communication bottlenecks* in the tampering computation.

What do we mean by communication bottlenecks? Imagine that the (random) inputs to a computation can be partitioned into two subsets X and Y such that two parties, Alice (holding X) and Bob (holding Y), can simulate the computation by communicating at most t bits. Why is this helpful? This class of computation (independent tampering on X and Y conditioned on small communication between X and Y) is precisely corresponds to the tampering class handled by (adaptive) leakage-leakage resilient split-state non-malleable codes (See extensions to Definition 3.1 and Remark 3.1). For clarity we define this tampering class as two-party t-communication tampering.[13]

Definition 7.1. *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^n$ be a function and $f_A : \{0,1\}^n \to \{0,1\}^n$,$f_B : \{0,1\}^n \to \{0,1\}^n$ such that $f(x,y) = f_A(x,y), f_B(x,y)$.*

We say that f is a two-party $t(n)$-communication tampering function *if there is a two-party protocol Π_f where two parties Alice and Bob communicate at most $t(n)$ bits such that for any $x, y \in \{0,1\}^n$, if Alice is given x and Bob is given y, Alice outputs $f_A(x,y)$ and Bob outputs $f_B(x,y)$.*

We denote the class of two-party $t(n)$-communication tampering functions as $t(n)-\mathsf{SS}$. Moreover, we say a non-malleable code for this tampering class is augmented *if the left half of the codeword, communication transcript, and outcome of the tampering experiment can be jointly simulated.*

[12] There is a syntactic problem here in that the output length of Parity does not match that of the tampering class, but consider instead a function whose first bit of output is the Parity of its inputs.

[13] This class is also referred to as "leaky split state tampering" in the literature.

Our goal, in this section, is to construct coding schemes, $\mathsf{Enc}, \mathsf{Dec}$, that induce communication bottlenecks when composed with any tampering function in the target class, i.e. for any tampering f, the function $\mathsf{Dec}(f(\mathsf{Enc}(X, Y))$ can be simulated by a two party protocol with at most t bits of communication (existing leakage-resilient split-state codes can handle t that is a constant fraction of $|X|$ and $|Y|$). More precisely, we want to construct a non-malleable reduction (Definition 2.3), $\mathsf{Enc}, \mathsf{Dec}$, from the computational tampering class, $\mathcal{C}$, to two-party t-communication tampering, i.e. for every tampering function $f \in \mathcal{C}$ there exists some distribution D_f over two-party t-communication tampering protocols such that

$$\mathsf{Dec}(f(\mathsf{Enc}(X, Y)) \approx_\varepsilon D_f(X, Y).$$

In Section 7.1, we will see how [BGW19] constructs such a non-malleable reduction for the class of decision tree tampering functions: where each tampered output bit is produced by a bounded number of queries to the input bits. ([BGW19]'s construction extends an earlier of [BDSKM16] for local tampering functions, which corresponds to the case where the queries are static: chosen independently of the input.)

Section 7.2 will not construct a communication bottlenecking non-malleable reduction directly, but implicitly. In particular, this Section will present [BDSG$^+$18]'s non-malleable reduction from small-depth circuit tampering to decision tree tampering. This reduction, as well as an earlier (but inefficient) construction of [CL17], critically use a technique from the circuit lower bound literature: random restrictions.

Finally, in Section 7.3, we will see how assumptions in the derandomization literature can be used to induce communication bottlenecks in polynomial size circuit tampering. In particular, how [BDL21] use hardness against *nondeterministic* circuits to construct non-malleable codes for polynomial size circuit tampering from augmented leakage-resilient split-state non-malleable codes. The code presented here has only inverse polynomial security. Other constructions for this class are known that do not rely on split-state non-malleable codes. Unfortunately, while these constructions are beautiful and achieve negligible security error, they are not fully explicit: relying either on an untamperable common random strings (CRS model) [CG14a, FMVW14], or poorly understood heuristic cryptographic assumptions [BDK$^+$19, DKP21].[14]

Challenges (comparison with pseudorandomness). The idea of communication bottlenecks has a fruitful history in pseudorandomness [Nis92, INW94, NZ96, IMZ19], but our setting presents unique challenges that make it difficult to extend results directly.

Firstly, non-malleable *codes* are required to meaningfully encode (and decode) information. (In contrast, pseudorandomness is only required to "fool" the computation.) While it is often intuitive how to tweak a pseudorandom generator to encode information, we must also simulate decoding of whatever the computation outputs with low communication, which can be delicate as the adversarial tampering could try to force decoding to behave badly.

Secondly, and perhaps more importantly, non-malleable codes must handle adversarial computations that take n bits of input and output n bits. (Compare with pseudorandomness, where it only necessary (and possible) to consider adversarial computations with short output.) For example, while it is straightforward to fool a single decision-tree (using bounded-inependence), n decision trees can copy X to the Y portion which cannot be simulated with low communication.

On the upside, here the adversarial computation doesn't have the last word: the (standard) non-malleability experiment only outputs after decoding. Additionally, non-malleable codes are not concerned with pseudorandomness, so there is no need to stringently account for the randomness consumed by the encoding.

Despite these differences, some of the constructions here will draw on techniques from the pseudorandomness literature, particularly those of Section 7.2 and Section 7.3.

[14] The latter constructions from cryptographic assumptions only achieve computational security: no *efficient* distinguisher (polynomial size circuit) can distinguish the real and simulated experiments.

7.1 Decision Tree Tampering [BGW19]

As mentioned above, decision trees of depth d capture tampering where each output bit is set arbitrarily after adaptively reading d locations of the input, where the choice of which input location to read next at any point in time can depend on the values of all the previous locations read.

Definition 7.2 (Decision Trees). *A decision tree with n input bits is a binary tree whose internal nodes have labels from $x_1, \ldots, x_n$ and whose leaves have labels from $\{0,1\}$. If a node has label x_i then the test performed at that node is to examine the i-th bit of the input. If the result is 0, one descends into the left subtree , whereas if the result is 1, one descends into the right subtree. The label of the leaf so reached is the output value on that particular input. The* depth *of a decision tree is the number of edges in a longest path from the root to a leaf. Let $\mathsf{DT}(t)$ denote decision trees with depth at most t.*

[BGW19] construct non-malleable codes resilient to tampering by decision-trees of depth $n^{1/4-o(1)}$.

Theorem 7.2 ([BGW19]). *For any $t = O(n^{1/4}/\log^{3/2} n)$, there is an explicit and efficient non-malleable code that is unconditionally secure against depth-t decision trees with codeword length $n = O(kt^2 \log^4 n/\log\log n)$ and error $\exp(-\Omega(n/t^4 \log^5 n))$ for a k-bit message.*

Technical overview. This theorem follows by constructing a non-malleable reduction (Def. 2.3) from decision-tree tampering to two-party bounded communication tampering. Theorem 7.2 follows from composing this reduction with a leakage-resilient split-state non-malleable code (i.e. a non-malleable code for two-party bounded communication tampering).

Lemma 7.1 (NMR from [BGW19]). *For any constant $\alpha \in (0,1)$ and $t = O(n^{1/4}/\log^{3/2} n)$, there is a $(\mathsf{DT}(t) \Rightarrow t(n)\text{–}\mathsf{SS}, \varepsilon)$-non-malleable reduction with rate $\Omega(1/t^2 \log^3 n)$ where $\varepsilon \leq \exp(-\Omega(n/t^4 \log^5 n))$.*

We will outline [BGW19]'s reduction for decision tree tampering. Their reduction builds on a reduction of [BDSKM16] for local tampering (where the bounded number queries to codeword are made non-adaptively). In fact, the two reductions are quite similar (though not identical), however the analysis differs substantially.

The key idea of this construction is to exploit size differences. The encoder and decoder will work independently on the left and right pieces of the message, so we will in turn think of having left and right encoders, decoders, codewords, and tampering functions (corresponding to the respective outputs).

First, suppose that the right piece of the message (corresponding to the right split-state codeword) is much longer than that of the left. Then, suppose both the right and left encoders and decoders are simply the identity function. Then, all the left tampering functions together will make a number of queries to the right codeword that is below the leakage threshold.

However, because the right is much longer than the left, the above analysis won't help in simulating tampering on the right with low leakage from the left. Instead, [BGW19] modify the left encoder/decoder to make it much longer than the right, but while retaining the property that the left can be decoded from just a few decision trees. To do so, sample a random small set, whose size is that of the message, in a much larger array. Then, plant the message in these locations and zero everything else out. Then, bit-wise secret share a description of the small set (i.e., its seed) such that the secrecy threshold is relatively large. To decode, simply extract the seed and output what is in the corresponding locations of the array.

Now, note that decoding the left still only requires at most relatively few queries to the right: decision tree depth times both encoded seed length plus message length. But we can't make the encoded seed too long or we will be dead again. Instead, [BGW19] critically use the fact that tampering is by a *forest* of decision trees. In particular, for any small set of tampering functions on the right, the seed remains uniformly chosen regardless of what queries the set makes, so we expect only a small fraction of any queries made to the array to actually hit the message locations. Strong concentrations bounds guarantee that this is more or less what actually happens. Then, simply union bound over all such subsets to

guarantee that collectively the right tampering function makes few queries to the left with overwhelming probability.

Finally, apply the same style of encoding used on the left to the right side to fix the syntactic mismatch and reduce to the case where the right and left messages are the same size.

7.2 Small-Depth Circuit Tampering [BDSG+18]

Small-depth circuit tampering captures the case where each tampered output bit is produced by a size $S(n) = n^{\omega(1)}$ circuit of depth $d(n) = o(\log n / \log\log n)$ over the standard basis with arbitrary fan-in, we denote this class $\mathsf{AC}_d(S(n))$. (This includes the case where each output is produced by a constant depth polynomial size circuit, AC^0.)

Theorem 7.3 ([BDSG+18, BGW19]). *For $d \leq c_1 \log n / \log\log n$, there exists an explicit, efficient, information theoretic non-malleable code for d-depth circuits (of unbounded fan-in) of size $\exp(n^{c_2/d})$ with error $\exp(-n^{\Omega(1/d)})$ and encoding length $n = k^{1+c}$, where $c, c_1, c_2 \in (0,1)$ are constants.*

For the special case of AC^0-tampering, there exist efficient non-malleable codes for $O(1)$-depth polynomial size circuits circuits with negligible error and encoding length $n = k^{1+o(1)}$.

[BDSG+18] showed how to use a tool from circuit lower bounds and derandomization, pseudorandom switching lemmas, to construct a non-malleable reduction from small-depth circuit tampering to decision tree tampering (which, as we have seen, can be reduced to split-state tampering). Prior to this construction, Chattopadhyay and Li had constructed an (invertable) *seedless non-malleable extractor* for small-depth circuit tampering [CL17]. However, unlike the construction here, the error of their extractor yields inefficient non-malleable codes (k length messages encode into codewords of length $n = 2^{\Omega(\sqrt{k})}$). We state [BDSG+18]'s main technical lemma, a non-malleable reduction from small-depth circuit tampering to small-depth decision tree tampering, before sketching their non-malleable reduction and its analysis here.

Lemma 7.2 ([BDSG+18]). *For $S, d, n, t \in \mathbb{N}, p, \delta \in (0,1)$, there exists $\sigma = \mathsf{poly}(t, \log(2^t S), \log(1/\delta), \log(1/p))$ and $m = O(\sigma \log n)$ such that, for any $2m \leq k \leq n(p/4)^d$,*

$$(\mathsf{AC}_d(S) \implies \mathsf{DT}(dmt), \varepsilon)$$

where

$$\varepsilon = nS\left(2^{2t+1}(5pt)^t + \delta\right) + \exp(-\tfrac{\sigma}{2\log(1/p)}).$$

Let us start by considering the simpler case of reducing w-DNFs (each clause contains at most w literals) to low-depth decision tree tampering. The reduction for general small-depth circuits will follow from a recursive composition of this reduction.

A non-malleable reduction $(\mathcal{E}, \mathcal{D})$ reducing DNF-tampering to small-depth decision tree tampering needs to satisfy two conditions (i) $\Pr[\mathcal{D}(\mathcal{E}(x)) = x] = 1$ for any x and, (ii) $\mathcal{D} \circ f \circ \mathcal{E}$ is a distribution over small-depth decision trees for any width-w DNF f. A classic result from circuit complexity, the switching lemma [FSS84, Ajt89, Yao85, Hås86], states DNFs become small-depth decision trees under random restrictions ("killing" input variables by independently fixing them to a random value with some probability). Thus a natural choice of $\mathcal{E}$ for satisfying (ii) is to simply sample from the generating distribution of restrictions and embed the message in the surviving variable locations (fixing the rest according to restriction). However, although $f \circ \mathcal{E}$ becomes a decision tree, it is not at all clear how to decode and fails even (i). To satisfy (i), a naive idea is to simply append the "survivor" location information to the encoding. However, this is now far from a random restriction (which requires among other things that the surviving variables are chosen independently of the random values used to fix the killed variables)is no longer guaranteed to "switch" the DNFs to decision trees with overwhelming probability.

To circumvent those limitations, we consider *pseudorandom switching lemmas*, usually arising in the context of derandomization [AW85, AAI+01, IMP12, GMR13, TX13, GW14], to relax the stringent properties of the distribution of random restrictions needed for classical switching lemmas. In particular, we invoke a pseudorandom switching lemma from Trevisan and Xue [TX13], which reduces DNFs to decision trees while only requiring that randomness specifying survivors and fixed values be σ-wise independent. This allows us to avoid problems with independence arising in the naive solution above. Now, we can append a σ-wise independent encoding of the (short) random seed that specifies the surviving variables. This gives us a generating distribution of random restrictions such that (a) DNFs are switched to decision trees, and (b) the seed can be decoded and used to extract the input locations.

At this point, we can satisfy (i) easily: $\mathcal{D}$ decodes the seed (whose encoding is always in, say, the first m coordinates), then uses the seed to specify the surviving variable locations and extract the original message. In addition to correctness, $f \circ \mathcal{E}$ becomes a distribution over local functions where the distribution only depends on f (not the message). However, composing $\mathcal{D}$ with $f \circ \mathcal{E}$ induces dependence on underlying message: tampered encoding of the seed, may depend on the message in the survivor locations. The encoded seed is comparatively small and thus (assuming the restricted DNF collapses to a low-depth decision tree) requires a comparatively small number of bits to be leaked from the message in order to simulate the tampering of the encoded seed. Given a well simulated seed we can accurately specify the decision trees that will tamper the input (the restricted DNFs whose output locations coincide with the survivors specified by the tampered seed). This intermediate leaky decision tampering class, which can be described via the following adversarial game: (1) the adversary commits to N decision trees, (2) the adversary can select m of the decision trees to get leakage from, (3) the adversary then selects the actual tampering function to apply from the remaining local functions. However, provided the seed length, m, is short enough, this just amounts to querying a slightly higher depth decision tree.

To deal with depth d circuits, we can recursively apply this restriction-embedding scheme d times. Each recursive application allows us to trade a layer of gates for another (adaptive) round of m bits of leakage in the leaky decision tree game. One can think of the recursively composed simulator as applying the composed random restrictions to collapse the circuit to decision trees and then, working inwardly, sampling all the seeds and the corresponding survivor locations until the final survivor locations can be used to specify ultimate decision tree tampering.

7.3 (Bounded) Polynomial Size Circuit Tampering [BDL21]

In this subsection, we show how to construct non-malleable codes for tampering by n^c-size circuits, where c is some constant. As mentioned at the outset, non-malleable codes for circuit tampering imply circuit lower bounds. Given that explicit lower bounds against superlinear size circuits are well-beyond our current techniques in complexity, assumptions are needed for such non-malleable codes. [BDL21] showed how to use hardness assumptions against *nondeterministic* circuits to construct such codes from split-state non-malleable codes. We begin by presenting the hardness assumption before giving a brief overview of [BDL21]'s construction.

Definition 7.3 (Nondeterministic circuit). *A* nondeterministic circuit C *is a circuit with "non-deterministic" inputs, in addition to the usual inputs. We say* C *evaluates to* 1 *on* x *if and only if there exists an assignment,* w*, to the non-deterministic input wires such that the circuit, evaluated deterministically on input* (x, w) *outputs* 1*.*

Assumption 1 (E requires exponential size nondeterministic circuits). *There is a language* $L \in \mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ *and a constant* γ *such that for sufficiently large* n*, non-deterministic circuits of size* $2^{\gamma n}$ *fail to decide* L *on inputs of length* n*.*

Informally, the above assumption says that non-uniformity and non-determinism do not always imply significant speed-ups of uniform deterministic computations.

Theorem 7.4 ([BDL21]). *If* E *requires exponential size non-deterministic circuits, then for every constant* c*, and for sufficiently large* k*, there is an explicit, efficient,* n^{-c}*-secure non-malleable code for* k*-bit messages, with codeword length* $n = \mathsf{poly}(k)$*, resilient to tampering by* n^c*-size circuits.*

[BDL21] construct their codes by "fooling" non-malleable codes for *split-state tampering* with special properties: augmented, leakage-resilient, and admitting a special form of encoding (given half a codeword, can efficiently sample the other half to encode any message).

Split-state tampering functions may manipulate the left and right halves of a codeword arbitrarily, but independently (i.e. functions such that $(c_L, c_R) \mapsto (f_L(c_L), f_R(c_R))$ for some f_L, f_R). Leakage-resilient split-state tampering allows each tampered codeword half to depend on bounded leakage from the opposite codeword half. In addition to split-state NMC, [BDL21] also use a pseudorandom generator (PRG) for nondeterministic circuits, where $c' > c$ is a constant. In particular, they require that the PRG, G, is secure even when given the seed (seed extending), i.e. no nondeterministic circuit of bounded polynomial size can distinguish $G(s)$ from a uniformly random string *and* s is a prefix of $G(s)$. The existence of such PRGs follows from Assumption 1 [KvMS12, IW97, KvM02, SU05, SU06, AASY16].

Given a (leakage-resilient) split-state non-malleable code, with necessary properties and a seed-extending pseudorandom PRG for nondeterministic circuits, G, we encode a message x by sampling the following:

$$(s, c_R) \text{ such that } (G(s), c_R) \text{is a split-state encoding of } x.$$

The proof proceeds by contradiction starting with the assumption that the construction is not non-malleable. The analysis follows by giving a nondeterministic reduction that uses the (assumed) malleablity of the construction to violate the PRG security.

1. Assume towards contradiction that (s, c_R) is *malleable* and fix the corresponding poly-size tampering function g which is *not* split-state and violates non-malleability.

2. Transform g into a split-state tampering function f_L, f_R on (c_L, c_R), where (1) f_L is *unbounded*, relies on $|s|$ bits of leakage from c_R and returns some c'_L, (2) f_R is efficient, relies on $|s|$ bits of leakage from c_L and returns c'_R. Crucially, split-state tampering function (f_L, f_R) is guaranteed to break non-malleability when $c_L = (s||y) = G(s)$.

3. Since (c_L, c_R) is a leakage-resilient split-state non-malleable code where c_L is uniform random, then when c_L is random (as opposed to in the construction where codewords are sampled as $(G(s), c_R)$), every tampering funcion (f'_L, f_R) *fails* to break non-malleability, even when f'_L is unbounded and chooses its output c'_L in the "optimal" way.

4. Construct an Arthur-Merlin protocol (with bounded poly-size Arthur), that distinguishes between input c_L being random or pseudorandom. Such a protocol can then be transformed into a non-deterministic polynomial bounded circuit (this follows from classical results: $\mathsf{IP}[O(1)] \subseteq \mathsf{AM} \subseteq \mathsf{NP/poly}$ [GS86, Bab85, BM88, AASY16]).

5. Intuitively, Arthur can efficiently compute all the values needed to simulate the tampering experiment except for c'_L, which is obtained from Merlin. Specifically, on input c_L, Arthur samples c_R, and computes $c'_R = f_R(c_R)$, as well as the leakage on c_R. Arthur sends c_L and the leakage on c_R to Merlin who responds with c'_L. If c_L is pseudorandom, then an honest Merlin will return $c'_L = f_L(c_L)$, and, with Merlin's help, Arthur can check that non-malleability is violated with this c'_L. If c_L is random, then despite any response $c'_L = f'_L(c_L)$ from Merlin, non-malleability will *not* be violated, and a dishonest Merlin cannot convince Arthur otherwise.

8 Application to Non-malleable Commitments

In this section, we discuss one of the most important applications of non-malleable codes in the split-state model. In [ASK+22], the authors construct a 1/3-rate NMC (which we described in Section 6.2), and then use the textbook non-malleable commitment scheme with computational binding and statistical hiding from [GPR16]. This construction achieves a communication cost of approximately 41 times the length of the message being committed. We begin by defining non-malleable commitments, introduced by Dolev, Dwork and Naor [DDN91], that give computational binding and statistical hiding property.

Definition 8.1. *[GPR16] A non-malleable commitment scheme, $\langle \mathcal{C}, \mathcal{R} \rangle$ is a two-phase, two-party protocol between a committer $\mathcal{C}$ and a receiver $\mathcal{R}$. In the commit phase, $\mathcal{C}$ uses secret m and interacts with $\mathcal{R}$ who uses no input. Let $z = \mathsf{Com}(m; r)$ denote $\mathcal{R}$'s view after the commit phase. Let $(w, m) = \mathsf{Decom}(z, m, r)$ denote $\mathcal{R}$'s view after the decommit phase, which $\mathcal{R}$ either accepts or rejects. We say that $\langle \mathcal{C}, \mathcal{R} \rangle$ is a computationally binding and ε-statistically hiding non-malleable commitment scheme if the following properties hold:*

1. ***Correctness:*** *If the parties follow the protocol, then $\mathcal{R}(z, w, m) = 1$, i.e., the receiver accepts.*
2. ***Binding:*** *For any PPT adversarial receiver $\mathcal{R}^*$, that outputs $(w', m'), (w, m), z$, with $m' \neq m$, the probability that $\mathcal{R}(z, w, m) = 1 = \mathcal{R}(z, w', m')$ is negligible.*
3. ***Hiding:*** *For all distinct message pairs m, m', $\{\mathsf{Com}(m; r)\}_r \approx_\varepsilon \{\mathsf{Com}(m'; r')\}_{r'}$.*
4. ***Non-malleability:*** *For avoiding trivial man-in-the-middle attack of copying the identity of the committer, we consider the committer and receiver to additionally have an identity $\mathsf{Id} \in \{0,1\}^\lambda$ as common input (λ is the computational security parameter). To define non-malleability, we consider the real/ideal paradigm. In the real interaction, there is a man-in-the-middle adversary M interacting with a committer, $\mathcal{C}$, in the left session and a receiver $\mathcal{R}$, in the right. All the quantities associated with the right interaction are denoted by the "tilde'd" versions of their left counterparts (e.g., $\mathcal{C}$ commits to m in the left interaction while M commits to $\tilde{m}$ in the right). Let MIM_m denote the random variable describing $(\mathsf{VIEW}, \tilde{m})$, consisting of M's view in the experiment and the value M commits to in the right interaction, given that $\mathcal{C}$ committed to m on the left. The ideal interaction is the same, except that $\mathcal{C}$ commits to an arbitrary message, say 0, on the left. Let MIM_0 denote the corresponding random variable for 0. M is forced to use an identity $\tilde{\mathsf{Id}}$ on the right, which is distinct from Id used on the left. MIM_m and MIM_0 output a special symbol $\perp_{\mathsf{Id}}$ when M has used the same identity on the right as received on the left.*
   Non-malleability guarantees that for every PPT man-in-the-middle M, and for all messages m, we have $\{\mathsf{MIM}_m(y)\}_{y \in \{0,1\}^*} \approx_c \{\mathsf{MIM}_0(y)\}_{y \in \{0,1\}^*}$, where y is the auxiliary input received by M.

The round complexity of a commitment scheme denotes the number of rounds of interaction between the committer and receiver. The communication complexity of a commitment scheme denotes the total size of the transcript of the interaction between the committer and the receiver.

The non-malleable commitment scheme from [GPR16] uses a 2-split-state augmented non-malleable code tolerating leakage as an underlying building block. By Theorems 3.2 and 3.4, the leakage-resilience requirement can be removed. So, if one instantiates this scheme with the 1/3-rate augmented non-malleable code, one gets a non-malleable commitment scheme with a communication complexity of $41 \cdot |\mathsf{message\ length}|$. We begin by looking at the building blocks used.

8.1 Building Blocks

The construction from [GPR16] requires two building blocks, which were instantiated in [ASK+22] as follows.

- A non-interactive computationally binding and statistically hiding commitment, $(\mathsf{Com}, \mathsf{Decom})$, with message space $\{0,1\}^{2\beta_1}$, which is a non-interactive two phase protocol as in Definition 8.1 satisfying correctness, computational binding and statistical hiding. There is a hashing based statistically hiding commitment of [HM96], which has commitment size of $\approx 9 \cdot (\mathsf{message\ length})$.
- A leakage resilient and augmented non-malleable code, $(\mathsf{Enc}, \mathsf{Dec})$, with message space $\{0,1\}^{\alpha}$ and codeword space $\{0,1\}^{\beta_1} \times \{0,1\}^{\beta_2}$.

8.2 Construction

We now describe the construction of non-malleable commitments from [GPR16], using the building blocks from Section 8.1. For multiplication and addition operations in the construction below, we assume a natural correspondence between the binary β_1-bit strings and the field $GF(2^{\beta_1})$.

- **Setup:** Let $\mathsf{Id} \in \{0,1\}^{\lambda}$ be $\mathcal{C}$'s identity, also given as input to $\mathcal{R}$. λ is the computational security parameter.
- **Inputs:** $\mathcal{C}$ has input message $m \in \{0,1\}^{\alpha}$ to be committed to. Id is a common input of both $\mathcal{C}$ and $\mathcal{R}$.
- **Commit Phase:**
  1. $\mathcal{C} \to \mathcal{R}$: Let $(L, R) \leftarrow \mathsf{Enc}(m \| \mathsf{Id})$. Pick random $r \leftarrow \{0,1\}^{\beta_1}$ and send $\mathsf{Com}(L \| r)$ to $\mathcal{R}$.
  2. $\mathcal{R} \to \mathcal{C}$: Send random $a \leftarrow \{0,1\}^{\beta_1} \backslash \{0^{|\beta_1|}\}$.
  3. $\mathcal{C} \to \mathcal{R}$: Send $b = ra + L$ and R.
- **Decommit Phase:** $\mathcal{C}$ opens the commitment in Step 1. Let $L' \| r'$ be the decommited value.
- **Receiver's Output:** If L' and r' do not satisfy $r'a + L' = b$, then output $\perp_{inc}$. Else, compute $m' \| \mathsf{Id}' = \mathsf{Dec}(L', R)$, and output $\perp_{\mathsf{Id}}$ if $\mathsf{Id}' = \mathsf{Id}$. Else output m'.

Figure 4: Non-malleable Commitment Scheme $\langle \mathcal{C}, \mathcal{R} \rangle$

In [GPR16], the additional property needed from the underlying NMC is called conditional augmented property [GPR16, Definition 10], which guarantees that if the left state L is first picked at random from the space of left state of valid codewords (whose decode is $\neq \perp$) and then the right state is picked, conditioned on the message and the left state, the augmented non-malleability (with right augmentedness) is still guaranteed. One can observe that the proof of [GPR16, Claim 2], showing that a non-malleable code is conditional augmented, only requires leakage resilience from the left state of the NMC. Hence, the main theorem of [GPR16, Theorem 1], with instantiations from Section 8.1, can be stated as follows.

Theorem 8.1. *[GPR16, ASK+22] If* $(\mathsf{Com}, \mathsf{Decom})$ *is a non-interactive computationally binding and statistically hiding commitment scheme, and* $(\mathsf{Enc}, \mathsf{Dec})$ *is a leakage resilient augmented non-malleable code, then the protocol* $\langle \mathcal{C}, \mathcal{R} \rangle$ *in Figure 4 is a non-malleable commitment scheme against synchronizing adversary with computational binding and statistical hiding.*

Further, using the hashing based non-interactive commitment scheme [HM96] and the non-malleable code from [ASK+22], the communication cost of the above scheme is 41α*, where* α *is the message length.*

9 The New Frontier, Open Questions.

In this section we will describe few interesting unresolved questions that are related to this survey:

Exponential error. To date, none of the explicit split-state non-malleable codes achieve genuinely exponential error, $\varepsilon = 2^{-\Omega(n)}$. In particular, if one desires error 2^{-k}, no construction with a codeword of length $O(k)$ is currently known. This is of particular importance in applications where the security parameter may be larger than the message length.

Surpassing Bourgain's extractor. As we already mentioned in Section 5, further improvements to the constructions of non-malleable extractors would imply an explicit construction of a two-source extractor with a negligible error and low sources entropy requirements. For more reading on the problem we refer to [COA21]. Finding such non-malleable extractors is an interesting open question.

Rate above $\frac{1}{3}$. The techniques mentioned in Section 6 do not allow us to build a non-malleable code with rate better than 1/3. Each of the states has to be at least as long as the message (because of the secret sharing property discussed in Section 3), and the seeded extractor trick requires the length of one of the states to be double the message length. Thus going below 1/3 requires a new approach, and is left as a (perhaps challenging) open question.

Is the construction from [ADL14] way stronger than we can prove? We discuss this construction in Section 4. The construction requires very large size of the vectors ($\Omega(n^4)$ coordinates, each of size $\Omega(n)$ for an n-bit message). We hypothesise, however, that [ADL14] should remain secure even for a constant number of coordinates. Even if the above is not true, finding an explicit attack would greatly expand our understanding of tamper resistance of the inner product. This might even have interesting consequences in additive combinatorics.

Eight is a crowd.. As we discuss in Remark 3.11, the question about the minimum number of states necessary to build a continuous non-malleable code remains unanswered. We know it's at least 3 and at most 8. We hypothesise that extending the techniques from [ADN+19b] to existential results in non-malleable extractors will yield an existential result in 6 state model. Closing the gap between 3 and 8 would be very interesting, especially if the answer is not 3!

References

[AAG+16] Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In *Theory of Cryptography Conference*, pages 393–417. Springer, 2016.

[AAI+01] Manindra Agrawal, Eric Allender, Russell Impagliazzo, Toniann Pitassi, and Steven Rudich. Reducing the complexity of reductions. *Computational Complexity*, 10(2):117–138, 2001.

[AASY16] Benny Applebaum, Sergei Artemenko, Ronen Shaltiel, and Guang Yang. Incompressible functions, relative-error extractors, and the power of nondeterministic reductions. *Comput. Complex.*, 25(2):349–418, 2016.

[AB16] Divesh Aggarwal and Jop Briët. Revisiting the sanders-bogolyubov-ruzsa theorem in f p n and its application to non-malleable codes. In *Information Theory (ISIT), 2016 IEEE International Symposium on*, pages 1322–1326. Ieee, 2016.

[ADKO15a] Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, pages 459–468, 2015.

[ADKO15b] Divesh Aggarwal, Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Leakage-resilient non-malleable codes. In *Theory of Cryptography Conference*, pages 398–426. Springer, 2015.

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

[ADN+19a] Divesh Aggarwal, Ivan Damgård, Jesper Buus Nielsen, Maciej Obremski, Erick Purwanto, João Ribeiro, and Mark Simkin. Stronger leakage-resilient and non-malleable secret sharing schemes for general access structures. In *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part II*, pages 510–539, 2019.

[ADN+19b] Divesh Aggarwal, Nico Döttling, Jesper Buus Nielsen, Maciej Obremski, and Erick Purwanto. Continuous non-malleable codes in the 8-split-state model. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 531–561. Springer, 2019.

[Agg15] Divesh Aggarwal. Affine-evasive sets modulo a prime. *Information Processing Letters*, 115(2):382–385, 2015.

[AGM+14] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes resistant to permutations and perturbations. *IACR Cryptology ePrint Archive*, 2014:841, 2014.

[AGM+15] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, pages 375–397, 2015.

[Ajt89] Miklós Ajtai. First-order definability on finite structures. *Ann. Pure Appl. Logic*, 45(3):211–225, 1989.

[AKO17] Divesh Aggarwal, Tomasz Kazana, and Maciej Obremski. Inception makes non-malleable codes stronger. In *Theory of Cryptography Conference*, pages 319–343. Springer, 2017.

[AO20] Divesh Aggarwal and Maciej Obremski. A constant rate non-malleable code in the split-state model. In *61st IEEE Annual Symposium on Foundations of Computer Science, FOCS 2020, Durham, NC, USA, November 16-19, 2020*, pages 1285–1294. IEEE, 2020.

[AOR+22] Divesh Aggarwal, Maciej Obremski, João Ribeiro, Mark Simkin, and Luisa Siniscalchi. Privacy amplification with tamperable memory via non-malleable two-source extractors. Unpublished Manuscript, 2022.

[ASK+22] Divesh Aggarwal, Sruthi Sekar, Bhavana Kanukurthi, Maciej Obremski, and Sai Lakshmi Bhavana Obbattu. Rate one-third non-malleable codes. ACM Symposium on the Theory of Computing (STOC), 2022. To Appear.

[AW85] Miklós Ajtai and Avi Wigderson. Deterministic simulation of probabilistic constant depth circuits (preliminary version). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 11–19, 1985.

[Bab85] László Babai. Trading group theory for randomness. In Robert Sedgewick, editor, *Proceedings of the 17th Annual ACM Symposium on Theory of Computing, May 6-8, 1985, Providence, Rhode Island, USA*, pages 421–429. ACM, 1985.

[BCD+18] Avraham Ben-Aroya, Eshan Chattopadhyay, Dean Doron, Xin Li, and Amnon Ta-Shma. A new approach for constructing low-error, two-source extractors. In *Proceedings of the 33rd Computational Complexity Conference*, CCC '18, pages 3:1–3:19, Germany, 2018. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik.

[BDK+19] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, Huijia Lin, and Tal Malkin. Non-malleable codes against bounded polynomial time tampering. In Yuval Ishai and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, volume 11476 of *Lecture Notes in Computer Science*, pages 501–530. Springer, 2019.

[BDKM20] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Limits to non-malleability. In Thomas Vidick, editor, *11th Innovations in Theoretical Computer Science Conference, ITCS 2020, January 12-14, 2020, Seattle, Washington, USA*, volume 151 of *LIPIcs*, pages 80:1–80:32. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020.

[BDL21] Marshall Ball, Dana Dachman-Soled, and Julian Loss. (nondeterministic) hardness vs. non-malleability. Unpublished Manuscript, 2021.

[BDSG+18] Marshall Ball, Dana Dachman-Soled, Siyao Guo, Tal Malkin, and Li-Yang Tan. Non-malleable codes for small-depth circuits. In *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 826–837. IEEE, 2018.

[BDSKM16] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes for bounded depth, bounded fan-in circuits. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 881–908. Springer, 2016.

[BDSKM18] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes from average-case hardness: Decision trees, and streaming space-bounded tampering. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 618–650. Springer, 2018.

[BFM+22] Gianluca Brian, Sebastian Faust, Elena Micheli, , and Daniele Venturi. Continuously non-malleable codes against persistent decision-tree tampering. Unpublished Manuscript, 2022.

[BFO+20] Gianluca Brian, Antonio Faonio, Maciej Obremski, Mark Simkin, and Daniele Venturi. Non-malleable secret sharing against bounded joint-tampering attacks in the plain model. Eurocrypt, 2020.

[BGW19] Marshall Ball, Siyao Guo, and Daniel Wichs. Non-malleable codes for decision trees. CRYPTO, 2019.

[Bih93] Eli Biham. New types of cryptanalytic attacks using related keys (extended abstract). In Tor Helleseth, editor, *Advances in Cryptology - EUROCRYPT '93, Workshop on the Theory and Application of of Cryptographic Techniques, Lofthus, Norway, May 23-27, 1993, Proceedings*, volume 765 of *Lecture Notes in Computer Science*, pages 398–409. Springer, 1993.

[BM88] László Babai and Shlomo Moran. Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes. *J. Comput. Syst. Sci.*, 36(2):254–276, 1988.

[BS19] Saikrishna Badrinarayanan and Akshayaram Srinivasan. Revisiting non-malleable secret sharing. In *Advances in Cryptology - EUROCRYPT 2019 - 38th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Darmstadt, Germany, May 19-23, 2019, Proceedings, Part I*, pages 593–622, 2019.

[CDTV16] Sandro Coretti, Yevgeniy Dodis, Björn Tackmann, and Daniele Venturi. Non-malleable encryption: Simpler, shorter, stronger. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography*, pages 306–335, Berlin, Heidelberg, 2016. Springer Berlin Heidelberg.

[CG14a] Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. In *ITCS*, 2014.

[CG14b] Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In *TCC*, 2014.

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

[CGM+16] Nishanth Chandran, Vipul Goyal, Pratyay Mukherjee, Omkant Pandey, and Jalaj Upadhyay. Block-wise non-malleable codes. In *ICALP*, volume 55 of *LIPIcs*, pages 31:1–31:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[CKOS19] Eshan Chattopadhyay, Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Privacy amplification from non-malleable codes. In Feng Hao, Sushmita Ruj, and Sourav Sen Gupta, editors, *Progress in Cryptology - INDOCRYPT 2019 - 20th International Conference on Cryptology in India, Hyderabad, India, December 15-18, 2019, Proceedings*, volume 11898 of *Lecture Notes in Computer Science*, pages 318–337. Springer, 2019.

[CKR16] Nishanth Chandran, Bhavana Kanukurthi, and Srinivasan Raghuraman. Information-theoretic local non-malleable codes and their applications. In *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, pages 367–392, 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, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1171–1184. ACM, 2017.

[CMTV15] Sandro Coretti, Ueli Maurer, Björn Tackmann, and Daniele Venturi. From single-bit to multi-bit public-key encryption via non-malleable codes. *Thoery of Cryptography- TCC*, 2015.

[COA21] Eldon Chung, Maciej Obremski, and Divesh Aggarwal. Extractors: Low entropy requirements colliding with non-malleability. Cryptology ePrint Archive, Report 2021/1480, 2021. `https://ia.cr/2021/1480`.

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

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

[DDN00] D. Dolev, C. Dwork, and M. Naor. Nonmalleable cryptography. *SIAM*, 30:391–437, 2000.

[DDV10] Francesco Davì, Stefan Dziembowski, and Daniele Venturi. Leakage-resilient storage. In Juan A. Garay and Roberto De Prisco, editors, *SCN*, volume 6280 of *Lecture Notes in Computer Science*, pages 121–137. Springer, 2010.

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

[DKO+18] Ivan Damgård, Tomasz Kazana, Maciej Obremski, Varun Raj, and Luisa Siniscalchi. Continuous nmc secure against permutations and overwrites, with applications to cca secure commitments. TCC, 2018.

[DKP21] Dana Dachman-Soled, Ilan Komargodski, and Rafael Pass. Non-malleable codes for bounded parallel-time tampering. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part III*, volume 12827 of *Lecture Notes in Computer Science*, pages 535–565. Springer, 2021.

[DKS19] Dana Dachman-Soled, Mukul Kulkarni, and Aria Shahverdi. Tight upper and lower bounds for leakage-resilient, locally decodable and updatable non-malleable codes. *Inf. Comput.*, 268, 2019.

[DLSZ20] Dana Dachman-Soled, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Locally decodable and updatable non-malleable codes and their applications. *J. Cryptol.*, 33(1):319–355, 2020.

[DPW10] Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In *ICS*, pages 434–452. Tsinghua University Press, 2010. Journal version appeared in Siam Journal of Computing, 2018.

[DW09] Yevgeniy Dodis and Daniel Wichs. Non-malleable extractors and symmetric key cryptography from weak secrets. In Michael Mitzenmacher, editor, *Proceedings of the 41st Annual ACM Symposium on Theory of Computing*, pages 601–610, Bethesda, MD, USA, 2009. ACM.

[FHMV17] Sebastian Faust, Kristina Hostáková, Pratyay Mukherjee, and Daniele Venturi. Non-malleable codes for space-bounded tampering. In *Annual International Cryptology Conference*, pages 95–126. Springer, 2017.

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

[FMVW14] S. Faust, P. Mukherjee, D. Venturi, and D. Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In *Eurocrypt*. Springer, 2014. To appear.

[FSS84] Merrick L. Furst, James B. Saxe, and Michael Sipser. Parity, circuits, and the polynomial-time hierarchy. *Mathematical Systems Theory*, 17(1):13–27, 1984.

[GK18] Vipul Goyal and Ashutosh Kumar. Non-malleable secret sharing. In Ilias Diakonikolas, David Kempe, and Monika Henzinger, editors, *Proceedings of the 50th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2018, Los Angeles, CA, USA, June 25-29, 2018*, pages 685–698. ACM, 2018.

[GLM+03] Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic Tamper-Proof (ATP) security: Theoretical foundations for security against hardware tampering. In Moni Naor, editor, *First Theory of Cryptography Conference — TCC 2004*, volume 2951 of *LNCS*, pages 258–277. Springer-Verlag, February 19–21 2003.

[GMR13] Parikshit Gopalan, Raghu Meka, and Omer Reingold. DNF sparsification and a faster deterministic counting algorithm. *Computational Complexity*, 22(2):275–310, 2013.

[GMW18] Divya Gupta, Hemanta K Maji, and Mingyuan Wang. Constant-rate non-malleable codes in the split-state model. Technical report, Technical Report Report 2017/1048, Cryptology ePrint Archive, 2018.

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

[GS86] Shafi Goldwasser and Michael Sipser. Private coins versus public coins in interactive proof systems. In Juris Hartmanis, editor, *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 59–68. ACM, 1986.

[GSZ21] Vipul Goyal, Akshayaram Srinivasan, and Chenzhi Zhu. Multi-source non-malleable extractors and applications. In *Annual International Conference on the Theory and Applications of Cryptographic Techniques*, pages 468–497. Springer, 2021.

[GW14] Oded Goldreich and Avi Wigderson. On derandomizing algorithms that err extremely rarely. In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 109–118, 2014.

[Hås86] Johan Håstad. Almost optimal lower bounds for small depth circuits. In *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*, pages 6–20, 1986.

[HM96] Shai Halevi and Silvio Micali. Practical and provably-secure commitment schemes from collision-free hashing. In Neal Koblitz, editor, *Advances in Cryptology—CRYPTO '96*, volume 1109 of *LNCS*, pages 201–215. Springer-Verlag, 18–22 August 1996.

[IKSS21] Yuval Ishai, Dakshita Khurana, Amit Sahai, and Akshayaram Srinivasan. On the round complexity of black-box secure MPC. In Tal Malkin and Chris Peikert, editors, *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021, Virtual Event, August 16-20, 2021, Proceedings, Part II*, volume 12826 of *Lecture Notes in Computer Science*, pages 214–243. Springer, 2021.

[IMP12] Russell Impagliazzo, William Matthews, and Ramamohan Paturi. A satisfiability algorithm for AC0. In *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, pages 961–972, 2012.

[IMZ19] Russell Impagliazzo, Raghu Meka, and David Zuckerman. Pseudorandomness from shrinkage. *J. ACM*, 66(2):11:1–11:16, 2019.

[INW94] Russell Impagliazzo, Noam Nisan, and Avi Wigderson. Pseudorandomness for network algorithms. In Frank Thomson Leighton and Michael T. Goodrich, editors, *Proceedings of the Twenty-Sixth Annual ACM Symposium on Theory of Computing, 23-25 May 1994, Montréal, Québec, Canada*, pages 356–364. ACM, 1994.

[IPSW06] Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *Advances in Cryptology—EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 308–327. Springer-Verlag, 2006.

[IW97] Russell Impagliazzo and Avi Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In Frank Thomson Leighton and Peter W. Shor, editors,

Proceedings of the Twenty-Ninth Annual ACM Symposium on the Theory of Computing, El Paso, Texas, USA, May 4-6, 1997, pages 220–229. ACM, 1997.

[JW15] Zahra Jafargholi and Daniel Wichs. Tamper detection and continuous non-malleable codes. In *Theory of Cryptography Conference*, pages 451–480. Springer, 2015.

[KOS17] Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Four-state non-malleable codes with explicit constant rate. In *Theory of Cryptography Conference*, pages 344–375. Springer, 2017.

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

[KSW96] John Kelsey, Bruce Schneier, and David A. Wagner. Key-schedule cryptanalysis of idea, g-des, gost, safer, and triple-des. In Neal Koblitz, editor, *Advances in Cryptology - CRYPTO '96, 16th Annual International Cryptology Conference, Santa Barbara, California, USA, August 18-22, 1996, Proceedings*, volume 1109 of *Lecture Notes in Computer Science*, pages 237–251. Springer, 1996.

[KvM02] Adam R. Klivans and Dieter van Melkebeek. Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses. *SIAM J. Comput.*, 31(5):1501–1526, 2002.

[KvMS12] Jeff Kinne, Dieter van Melkebeek, and Ronen Shaltiel. Pseudorandom generators, typically-correct derandomization, and circuit lower bounds. *Comput. Complex.*, 21(1):3–61, 2012.

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

[Li19a] Xin Li. Non-malleable extractors and non-malleable codes: Partially optimal constructions. In *Computational Complexity Conference, CCC 2019, New Brunswick, June 18-20, 2019*, 2019.

[Li19b] Xin Li. Non-malleable extractors and non-malleable codes: Partially optimal constructions. *CCC'19*, 2019.

[LL12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In *Advances in Cryptology–CRYPTO 2012*, pages 517–532. Springer, 2012.

[Nis92] Noam Nisan. Pseudorandom generators for space-bounded computation. *Comb.*, 12(4):449–461, 1992.

[NZ96] Noam Nisan and David Zuckerman. Randomness is linear in space. *Journal of Computer and System Sciences*, 52(1):43–53, 1996.

[Sam07] Alex Samorodnitsky. Low-degree tests at large distances. In *ACM symposium on Theory of computing*, pages 506–515. ACM, 2007.

[SU05] Ronen Shaltiel and Christopher Umans. Simple extractors for all min-entropies and a new pseudorandom generator. *J. ACM*, 52(2):172–216, 2005.

[SU06] Ronen Shaltiel and Christopher Umans. Pseudorandomness for approximate counting and sampling. *Comput. Complex.*, 15(4):298–341, 2006.

[SV19] Akshayaram Srinivasan and Prashant Nalini Vasudevan. Leakage resilient secret sharing and applications. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology – CRYPTO 2019*, pages 480–509, Cham, 2019. Springer International Publishing.

[TX13] Luca Trevisan and Tongke Xue. A derandomized switching lemma and an improved derandomization of AC0. In *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, pages 242–247. IEEE Computer Society, 2013.

[Yao85] Andrew Chi-Chih Yao. Separating the polynomial-time hierarchy by oracles (preliminary version). In *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, pages 1–10, 1985.