Non-Malleable Codes for Bounded Depth, Bounded Fan-in Circuits

Marshall Ball[1], Dana Dachman-Soled[2], Mukul Kulkarni[2], and Tal Malkin[1]

[1] Columbia University
{marshall,tal}@cs.columbia.edu
[2] University of Maryland
danadach@ece.umd.edu, mukul@umd.edu

Abstract. We show how to construct efficient, unconditionally secure non-malleable codes for bounded output locality. In particular, our scheme is resilient against functions such that any output bit is dependent on at most n^δ bits, where n is the total number of bits in a codeword and $0 \le \delta < 1$ a constant. Notably, this tampering class includes NC^0 .

1 Introduction

Non-malleable codes were first introduced by Dziembowski, Pietrzak and Wichs [DPW10] as an extension of error-correcting codes. Whereas error-correcting codes provide the guarantee that (if not too many errors occur) the receiver can recover the original message from a corrupted codeword, non-malleable codes are essentially concerned with security. In other words, correct decoding of corrupted codewords is not guaranteed (nor required), but it is instead guaranteed that adversarial corruptions cannot influence the output of the decoding in a way that depends on the original message: the decoding is either correct or *independent* of the original message

The main application of non-malleable codes is in the setting of tamper-resilient computation. Indeed, as suggested in the initial work of Dziembowski et al. [DPW10], non-malleable codes can be used to encode a secret state in the memory of a device such that a tampering adversary interacting with the device does not learn anything more than the input-output behavior. Unfortunately, it is impossible to construct non-malleable codes secure against arbitrary tampering, since the adversary can always apply the tampering function that decodes the entire codeword to recover the message m and then re-encodes a related message m'. Thus, non-malleable codes are typically constructed against limited classes of tampering functions $\mathcal{F}$. Indeed, given this perspective, error correcting codes can be viewed as a special case of non-malleable codes, where the class of tampering functions, $\mathcal{F}$, consists of functions which can only modify some fraction of the input symbols. Since non-malleable codes have a weaker guarantee than error correcting codes, there is potential to achieve non-malleable codes against much broader classes of tampering functions $\mathcal{F}$.

Indeed, there has been a large body of work on constructing non-malleable codes against classes of tampering functions which can potentially change every bit of the codeword, and for which no error correcting is possible. In particular, much attention has been given in the literature to the bit-wise tampering class (cf. [DPW10,CKM11,CCFP11]), where each bit of the codeword can be tampered individually, and its generalization, the split state tampering class (cf. [LL12,DKO13,ADL14,CG14a,CG14b,ADKO15,AAG+16]), where the codeword is split into two or more parts, each of which can be tampered individually (and independently of other blocks). One goal in this line of papers is to bring down the number of states, preferably to just two split states. Another goal is to increase the *rate* of the code, defined as the ratio k/n where k is the length of the original message and n is the length of the codeword outputted by the encoding algorithm. A constant-rate code is preferred, with the best possible rate being $1 - o(1)$.

However, beyond some non-explicit (randomized) or inefficient constructions for more general classes (cf. [DPW10,CG14a,FMVW14]), almost all known results are only for function classes that are split state or "compartmentalized". There are a few exceptions, providing explicit and efficient non-malleable codes against

non-compartmentalized classes of functions, including Chabanne et al. [CCP12]—who address certain types of linear tampering functions—and Agrawal et al. [AGM$^+$15a,AGM$^+$15b]—who address the class of functions that can permute the bits of a codeword, flip individual bits, or set them to 0 or 1.

Other than the above results, achieving (explicit and efficient) non-malleable codes against natural tampering classes that are not split-state is a fundamental open question in this area, and is the focus of our paper.

1.1 Our Results

In this work, we devise explicit, efficient, and unconditionally secure non-malleable codes against a powerful tampering class which includes all bounded-depth circuits with bounded fan-in and unbounded fan-out. Specifically, we consider the class Local$^{\ell_o}$, consisting of all functions $f : \{0,1\}^n \to \{0,1\}^n$ that can be computed with output locality $\ell_o(n)$, where each output bit depends on at most $\ell_o(n)$ input bits. Note that this class includes all fan-in-b circuits of depth at most $\log_b \ell_o$.

The class of bounded depth circuits is natural both as a complexity class and in the context of practical tampering attacks, where it seems to realistically capture the capabilities of a tampering adversary who has *limited time to tamper with memory* before the memory gets overwritten and/or refreshed. Moreover, the class of bounded output locality functions is a natural class in its own right, and is in fact much broader, including arbitrarily complex functions (even those outside of P), as long as the output locality constraint is maintained; we do not impose any constraints on the number or type of gates in the circuit. Finally, as we discuss below, our constructions actually hold for an even broader class, that also includes all split state functions, and beyond. We prove the following.

> **Main Theorem (informal)**: *For any $\ell_o = o(\frac{n}{\log n})$, there is an explicit, unconditionally secure non-malleable code for* Local$^{\ell_o}$, *which encodes a $2k$ bit string into a string of length $n = \Theta(k\ell_o)$ bits. The encoding and decoding run in time polynomial in n, namely* poly(k, ℓ_o).

This construction can be instantiated for any $\ell_o = o(n/\log n)$, and the resulting code has rate $\Theta(1/\ell_o)$. In general, since the the output length is $n = \Theta(k\ell_o)$ bits, this may result in super-polynomial length encoding. However, using sublinear locality n^δ yields an efficient code. We highlight this, as well as the special cases of *constant depth* circuits (a subset of Local$^{O(1)}$), in the following.

> **Corollaries**: *There are efficient, explicit, and unconditionally secure non-malleable codes for the following classes:*
> - Local$^{n^\delta}$ *for any constant $0 \le \delta < 1$, with inverse-polynomial rate.*
> - NC^0 *with rate $\Theta(1/\ell_o)$ for any $\ell_o = \omega(1)$.*
> - NC^0_c *for any constant c, with constant rate.*

The first corollary follows by instantiating the main theorem with $\ell_o = n^\delta$, the second by using any ℓ_o that is super constant (e.g., $\log^*(n)$), and the third by using $\ell_o = 2^c$ (a constant).

While our result for NC^0 correspond to constant depth circuits, the first corollary above implies as a special case that the code is also non-malleable against any $\delta \log n$ depth NC circuit, for any constant $0 \le \delta < 1$. Note that, since separations between P and NC^1 are not known, constructing (unconditional) non-malleable codes against NC^1 is unlikely, since an attacker in P can always decode and re-encode a related message, thus immediately breaking non-malleability.

Intermediate Results for (Input-and-Output) Local Functions. To prove our results, we use the concept of *non-malleable reduction*, introduced by Aggarwal et al. [ADKO15]. Informally, a class of functions $\mathcal{F}$ reduces to the class $\mathcal{G}$, if there is an encoding and decoding algorithms satisfying the following: applying the encoding to the input, then applying any $f \in \mathcal{F}$, and then applying the decoding, can be simulated by directly applying some function $g \in \mathcal{G}$ to the input. [ADKO15] prove that in this case a non-malleable code

for $\mathcal{G}$ can be used to construct one for $\mathcal{F}$, and further prove a composition theorem, providing an elegant and powerful way to construct non-malleable codes.

Following this technique, we start by proving two separate results, and compose them (together with known results for the class of split state functions), to obtain a restricted variant of the main theorem above. We then use the same ideas to show a single construction allowing for a better combined analysis that achieves the full main theorem (via reduction to the class of split state functions). We believe our techniques are more comprehensible presented in this modular fashion, and the intermediate results are of independent interest.

First, we consider the class $\mathrm{Local}_{\ell_i}^{\ell_o}$ of local functions, with output locality ℓ_o as well as input locality ℓ_i (namely each input bit influences at most ℓ_i output bits). This class includes bounded-depth circuits with bounded fan-in and bounded fan-out. Our first intermediate result shows that the class $\mathrm{Local}_{\tilde{O}(\sqrt{n})}^{\tilde{O}(\sqrt{n})}$ (and in fact a larger, leaky version of it) can be non-malleably reduced to the class of split state functions. Plugging in known results for non-malleable split state codes, we obtain a non-malleable code for this class. Our second result shows a non-malleable reduction of the class $\mathrm{Local}^{\tilde{O}(\sqrt{n})}$ to the above class (thus giving a non-malleable code for functions with output locality $\tilde{O}(\sqrt{n})$). Finally, we combine the encoding schemes presented previously to a single encoding scheme (via a reduction to split state class), and improve the analysis to show resilience against $o(n/\log n)$ output locality.

We remark that our first technical result for (input and output) local functions is of independent interest, and although as stated it is strictly weaker than our output-local results, the construction can have advantages in terms of complexity and concrete parameters, and has stronger resilience to leakage and to tampering functions that are both local and split-state, as we discuss next. We believe that both $\mathrm{Local}_{\ell_i}^{\ell_o}$ and Local^{ℓ_o} are interesting classes, capturing natural types of tampering adversaries.

Extended Classes: Combining with Split State and Beyond. Our results are in fact broader than presented so far. First, every one of our results works not only for the class of functions claimed, but also for any split state function. This is because for all of our schemes, encoding is applied independently on each half of the input, and thus can handle a split-state tampering function trivially.

Furthermore, our intermediate result for (input-output) local functions can handle any function that applies arbitrary changes *within* each part and has bounded input and output locality *between* the two parts (this class is much broader than all functions that are either split state or local). More precisely, we can handle functions where any bit on the left affects at most $\tilde{O}(\sqrt{n})$ bits on the right (and vice-versa), and any bit on the left is affected by at most $\tilde{O}(\sqrt{n})$ bits on the right (and vice-versa).

Finally, our constructions can also handle some leakage to the tampering function, capturing an adversary that first leaks some bits, and can then select a tampering function. For our input-output local tampering result, the leakage can be a constant fraction of the bits, while for our output-local tampering result, the leakage is more limited.

Relation of Our Class to Previous Work. As mentioned above, almost all previous results presenting explicit and efficient non-malleable codes, do so for a split state tampering class (with two or more states). These classes are a special case of ours, as we explained, which is not surprising given that we use results for split state functions as a starting point to prove our result. As for the exceptions that go beyond split state, we note that the class of functions that permute the bits or apply bitwise manipulations, introduced by [AGM+15a], is also a special case of our class, as it is a subset of Local^1 (in fact, even a subset of Local_1^1). The restricted linear tampering class considered by [CCP12], on the other hand, seems incomparable to our class of output-local functions.

Thus, in terms of the tampering class captured, our results simultaneously encompass (and significantly extend) almost all previously known explicit, efficient constructions of non-malleable codes (we are aware of only one exception). This is not the case in terms of the *rate*, where several previous works focus on optimizing the rate for smaller classes of functions (e.g., [CG14b] achieve rate $1 - o(1)$ non-malleable codes for bit-wise tampering functions, and [AAG+16] do so for split-state functions under computational assumptions), while we only achieve a constant rate for these classes.

We also mention that the original work of Dziembowski et al. [DPW10] already considered the question of constructing non-malleable codes against the class $\text{Local}^{\delta \cdot n}$, where n is the length of the codeword and $\delta < 1$ is a constant. We emphasize, however, that in [DPW10] (and an improvement in [CG14a]), they showed a construction of non-malleable codes against $\text{Local}^{\delta \cdot n}$ in a *non-standard, random oracle model* where the encoding and decoding functions make queries to a random oracle, but the adversarial tampering function *does not* query the random oracle. Our work shows that it is possible to construct non-malleable codes for $\text{Local}^{\delta \cdot n}$ for $\delta = o(1/\log n)$ in the standard model, with *no* random oracle.

On Randomized Decoding. Our constructions require the decoding function of the non-malleable code to be randomized. We note that, unlike the case of error correcting codes and encryption schemes, deterministic decoding for non-malleable codes *does not* seem to be without loss of generality, even in the case where the encoding scheme enjoys perfect correctness. To see why, note that while perfect correctness guarantees that all possible coins of the decoding algorithm produce the same output on a *valid codeword*, correctness provides no guarantees in the case when the codeword is corrupted and so it is not possible to derandomize by simply fixing an arbitrary sequence of coins for the decoder. Moreover, since the decoder holds no secret key in the non-malleable codes setting, it is also not possible to derandomize the decoding process by including the key of a pseudorandom function in the secret key. Since the standard definition of non-malleable codes already allows for randomized encoding, and since the standard definition of non-malleable codes only guarantees security in the one-time setting—wherein each time the codeword is decoded it must be refreshed by re-encoding the original message—we believe that allowing randomized decoding is the natural and "correct" definition for non-malleable codes (although the original definition required deterministic decoding).

Interestingly, we can combine our technical building blocks into a construction of non-malleable codes against Local^{ℓ_o} for any $\ell_o \leq n^{1/4}$, using *deterministic* decoding (the construction is described in Appendix A). Unfortunately, when compared to our construction utilizing *randomized* decoding, this construction has a lower rate of $O(1/\ell_o^2)$ (instead of $O(1/\ell_o)$), and due to that also lower output locality that can be supported ($\text{Local}^{n^{1/4}}$ instead of any $\text{Local}^{n^{\delta}}$ efficient or $\text{Local}^{o(n/\log n)}$ inefficient).

We therefore leave as an interesting open question to resolve whether ramdomized decoding is *necessary* for achieving security against certain tampering classes, $\mathcal{F}$, or whether there is a generic way to derandomize decoding algorithms for non-malleable codes.

1.2 Technical Overview

We give a high level technical overview of our constructions. We use as an underlying tool a so called "reconstructable probabilistic encoding scheme", a code that can correct a constant fraction of errors (denoted c^{err}), and enjoys some additional secret-sharing like properties: given a (smaller) constant fraction c^{sec} of the codeword gives no information on the encoded message, and can in fact be completed to a (correctly distributed) encoding of any message. This or similar tools were used in previous works either implicitly or explicitly, e.g., the construction based on Reed Solomon codes and Shamir secret sharing with Berlekamp-Welch correction, as used already in [BOGW88] is a RPE (any small enough subset of shares is distributed uniformly at random, and any such collection of shares can be extended to be the sharing of any message of our choice). Other RPE schemes with possibly improved parameters can be constructed from, e.g., [DGR00,CDMW08,CDTV16,CDMW15].

Handling Local Functions. Local functions are functions that have both small input and small output locality (i.e. each input bit affects a small number of output bits and each output bit depends on a small number of input bits). Our goal is to show a *non-malleable reduction* from a class of local functions with appropriate parameters, to the class of split-state functions. Loosely speaking, a non-malleable reduction from a class $\mathcal{F}$ to a class $\mathcal{G}$, is a pair (E, D) of encoding/decoding functions along with a *reduction* that transforms every $f \in \mathcal{F}$ into a distribution G_f over functions $g \in \mathcal{G}$, such that for every x, the distributions $\mathsf{D}(f(\mathsf{E}(x)))$ and $G_f(x)$ are statistically close. In the case of reductions to split-state, we let $x = (\mathsf{L}, \mathsf{R})$ where

$\mathsf{L}, \mathsf{R} \in \{0,1\}^k$. We want to construct (E, D) such that, informally, given any local f, the effect of applying f to the encoding $\mathsf{E}(x)$ and then decoding $\mathsf{D}(f(\mathsf{E}(x)))$, can be simulated by applying some split state function $g = (g_L, g_R)$ directly to $x = (\mathsf{L}, \mathsf{R})$.

We will use an encoding that works on each half of the input separately, and outputs $E(\mathsf{L}, \mathsf{R}) = (E^L(\mathsf{L}), E^R(\mathsf{R})) = (s^{\mathsf{L}}, s^{\mathsf{R}})$, where $|s^{\mathsf{L}}| = n_L, |s^{\mathsf{R}}| = n_R$ (we will refer to these as "left" and "right" sides, though as we will see they will not be of equal lengths, and we will have appropriately designed decoding algorithms for each part separately). Now for any given local f, consider $f(s^{\mathsf{L}}, s^{\mathsf{R}}) = (f^L(s^{\mathsf{L}}, s^{\mathsf{R}}), f^R(s^{\mathsf{L}}, s^{\mathsf{R}}))$. Clearly, if f^L only depended on s^{L} and f^R only depended on s^{R}, we would be done (as this would naturally correspond to a distribution of split state functions on the original $x = (\mathsf{L}, \mathsf{R})$). However, this is generally not the case, and we need to take care of "cross-effects" of s^{R} on f^L and s^{L} on f^R.

Let's start with f^L, and notice that if its output locality is at most ℓ_o, then at most $n_L \ell_o$ bits from s^{R} could possibly influence the output of f^L. Thus, we will use E^R that is an RPE with $n_L \ell_o \leq c^{\mathsf{sec}} n_R$. This means that we can just fix the relevant $n_L \ell_o$ bits from $s^{\mathsf{R}} = E^R(\mathsf{R})$ randomly (and independently of R), and now f^L will only depend on s^{L}, while s^{R} can still be completed to a correctly distributed encoding of R. Note that this requires making the right side larger than the left side ($n_R \geq \frac{n_L \ell_o}{c^{\mathsf{sec}}}$).

Now let's consider f^R. Clearly we cannot directly do the same thing we did for f^L, since that technique required n_R to be much longer than n_L, while applying it here would require the opposite. Instead, we will take advantage of the smaller size on the left, and its limited input locality. Specifically, if the input locality of f^L is ℓ_i, then at most $n_L \ell_i$ bits on the right side can be influenced by s^{L}.

A first (failed) attempt would be to just make sure that the encoding on the right can correct up to $n_L \ell_i$ errors, and hope that we can therefore set s^{L} arbitrarily when computing f^R and the resulting encoding would still be decoded to the same initial value R. While this argument works if the only changes made to s^{R} (a valid codeword) are caused by the "crossing bits" from s^{L}, it fails to take into account that f^R can in fact apply other changes inside s^{R}, and so it could be that s^{R} is malformed in such a way that applying f^R will cause it to decode differently in a way that crucially depends on s^{L}. The issue here seems to be that there is an exact threshold for when the decoding algorithm succeeds or not, and thus the function can be designed so that f^R is just over or under the threshold depending on the left side.

To overcome this problem, we use randomized decoding and a "consistency check" technique introduced in [CDMW08], and a forthcoming version by the same authors [CDMW15], in a different context. Roughly speaking, we make the right side encoding redundant, so that *any* large enough subset of bits is enough to recover R. An RPE has this property due its error correction capabilities. The decoding algorithm will decode via the first such subset, but will check a *random* subset of bits were consistent with a particular corrected codeword. This will yield similar behavior, regardless of which subset is used to decode. This construction has various subtleties, but they are all inherited from previous work, so we do not explain them here. The main point is that, like in [CDMW08,CDMW15], while the real decoding algorithm uses the *first* subset large enough, it can be simulated by using *any* other large enough subset.

Now, using the fact that "large enough" is not too large, and that at most $n_L \ell_i$ bits on the right side can be influenced by s^{L}, we can show that with high probability, there is a large enough "clean" subset of s^{R} that has *no* influence from s^{L}. The real decoding algorithm could be simulated by a decoding that uses this clean subset, which in turn means that the output of the decoding on $f^R(s^{\mathsf{L}}, s^{\mathsf{R}})$ is in fact independent of s^{L}, as needed.

Putting the above together provides us the first result, namely a non-malleable reduction from local to split state functions. We note that the proof above in fact works for a more general class of functions (a fact we will use in our second construction). In particular, the first part requires a limit on the output locality of f^L, and the second part requires a limit on the output locality of f^R and the input locality of f^L, where all of these only refer to "cross-over" influences (within each part separately f can be arbitrary). Moreover, due to our use of encoding, security is maintained even with leakage, as long as the leakage is a constant fraction of bits on the left and a constant fraction on the right, independently. Similarly, security is maintained even when a constant fraction of bits on the left do not adhere to the input locality bound.

Removing Input Locality. We next present a non-malleable reduction from output local functions (which have no restriction on input locality) to local functions. Now let f be an output local tampering function. Since the input and output to f are the same size, note that the *average* input locality of f can be bounded by its output locality, ℓ_o. Our local construction above requires low input locality for the left side, but also requires the left side to be much shorter than the right side. Unfortunately, what this means is that the input locality of *all* bits on the left side of the local encoding described above can be far higher than average. So, in order to bound the average input locality of the left side, we must increase the length of the left side, but this destroys our analysis from the first construction.

In order to achieve the best of both worlds, our idea is to construct a non-malleable reduction which increases the size of the left side of the underlying local encoding by adding dummy inputs. The "relevant" inputs, which correspond to bits of the left side of the underlying local encoding, are placed randomly throughout the left side of the new encoding. The idea is that since the adversary does not know which bit positions on the left side are "relevant," it cannot succeed in causing too many "relevant" positions to have input locality that is too much above average.

But now, in order to decode properly, the decoding algorithm must be able to recover these "relevant" locations, without sharing a secret state with the encoding algorithm (which is disallowed in the standard non-malleable codes setting). In order to do this, the first idea is to encode the relevant positions on the left side of the new encoding in an additional string, which is then appended to the left side during the new encoding procedure. Unfortunately, it is not clear how to make this work: Since this additional string is long, it can depend on a large number of input bits from both the left and right sides; on the other hand, in order to obtain a reduction from output local to local functions, the reduction must be able to recover this (possibly tampered) additional string so that it "knows" which output bits of $\widetilde{X}^{\mathsf{L}}$ are relevant.

The solution is to use a PRG with a short seed. The seed of the PRG is now the additional string that is appended to the left side and the output of the PRG yields an indicator string which specifies the "relevant" locations for decoding. Note that now since the PRG seed of length r is short, we can, using the leakage resilient properties of the underlying local code, leak *all* $r \cdot \ell_o \leq c^{\mathsf{sec}} \cdot n_L \leq c^{\mathsf{sec}} \cdot n_R$ number of bits affecting these output locations from both the left and right sides.

Moreover, because the tampering attacker is very limited, in the sense that it must choose the tampering function before learning any information about the output of the PRG, we are able to show that Nisan's PRG (see Definition 12), an *unconditional* PRG is sufficient for our construction. Thus, our construction does not rely on any computational assumption.

Improving the parameters. Ultimately the technique sketched above and presented in the body of the paper imposes two restrictions on output locality (modulo smaller terms): (1) $n_L \ell_o \leq n_R$ (2) $\ell_o \approx \ell_i \leq n_L$. Together these restrictions imply tolerance against output locality of approximately $\sqrt{n}$. The first restriction follows from the asymmetric encoding to handle bits on the left dependent on the right. The second restriction results from handling bits on the left of affecting the right side's consistency check.

To bypass this $\sqrt{n}$ barrier, we consider the two encoding schemes as a single scheme. Then in analysis, we can use the pseudorandom hiding of the left side encoding to relax the second bound. Namely, with high probability only a small portion of the left side RPE affects the consistency check, even if the consistency check and/or output locality is large with respect to n_L. This simple change in analysis gives resilience against $o(n/\log n)$ output locality.

1.3 Other Related Work

The concept of non-malleability was introduced by Dolev, Dwork and Naor [DDN00] and has been applied widely in cryptography since. Although it was defined in the computational setting, most recent work on non-malleability has been in the information-theoretic setting. The study of non-malleable codes was inspired by error-correcting codes and early works on tamper resilience [ISW03], [GLM+04], [IPSW06].

Dziembowski, Pietrzak and Wichs [DPW10] motivated and formalized the notion of non-malleable codes. They showed the existence of such codes for the class of all bit-wise independent functions (which can

be viewed as split state with n parts). Later work on split state classes improved this by reducing the number of states, increasing the rate, or adding desirable features to the scheme. For example, [DKO13] presented an information theoretic non-malleable code for 1-bit messages against 2 state split state functions, followed by [ADL14], who gave an information-theoretic construction for k-bit messages using results from additive combinatorics. A constant rate construction for a constant (> 2) number of states was provided in [CZ14,ADL14]. This line of research culminated with the result of [ADKO15], who used their reduction-based framework to achieve constant rate codes for two state split-state functions (using several intermediate constructions against various classes of functions). [AAG$^+$16] improve this to (optimal) rate $1 - o(1)$ non-malleable codes for two states, in the computational setting.

Beyond the above and other results constructing explicit efficient codes, there are several inefficient, existential or randomized constructions for much more general classes of functions (sometimes presented as efficient constructions in a random-oracle model). In particular, Dziembowski et al. [DPW10] gave an existential proof for the existence non-malleable codes secure against any 'small-enough' tampering family $(< 2^{2^n})$. [CG14a,FMVW14] give randomized construction of non-malleable codes against bounded poly-size circuits (where the bound on the circuit size is selected prior to the code).

Several other variants and enhanced models were considered. For example, [CKM11], in the context of designing UC secure protocols via tamperable hardware tokens, consider a variant of non-malleable codes which has *deterministic* encryption and decryption. It is interesting to note the contrast between their restriction to deterministic encoding (and decoding) and our relaxation to randomized decoding (and encoding). They provide efficient constructions for bit and block-wise functions, as well as more general but inefficient constructions. [LL12], in the context of securing cryptographic protocols against continual split-state leakage and tampering, provide a (computational) non-malleable code for split state functions, in the CRS model. This was one of the first works using the split state model for tampering. [DLSZ15,CKR16] consider a variant of non-malleable codes that is also locally decodable and updatable. [FMNV14] allow continual tampering, and [ADKO14] allow for bounded leakage model. As discussed previously, [CCP12] considers a subclass of linear tampering functions. We guide the interested reader to [KKS11] and [LL12] for illustrative discussion of various models for tamper and leakage resilience.

2 Preliminaries

2.1 Notation

Firstly, we present some standard notations that will be used in what follows. For any positive integer n, $[n] := \{1, \ldots, n\}$. If $x = (x_1, \ldots, x_n) \in \Sigma^n$ (for some set Σ), then $x_{i:j} := (x_i, x_{i+1}, \ldots, x_{j-1}, x_j)$ for $i \leq j$. If Σ is a set, then $\Sigma^\Sigma := \{f : \Sigma \to \Sigma\}$, the set of all functions from Σ to Σ. We say two vectors $x, y \in \Sigma^n$ are ε-*far* if they disagree on at least $\varepsilon \cdot n$ indices, $|\{i : x_i \neq y_i\}| \geq \varepsilon n$. Conversely, we say two vectors $x, y \in \Sigma^n$ are $(1 - \varepsilon)$-*close* if they agree on at least $(1 - \varepsilon) \cdot n$ indices, $|\{i : x_i = y_i\}| \geq (1 - \varepsilon)n$. Alternatively, for $x, y \in \mathrm{GF}(2)^n$ define their distance to be $d(x, y) := \frac{\|x+y\|_0}{n}$. (I.e. x and y are ε-far if $d(x, y) \geq \varepsilon$.) We take the statistical distance between two distributions, A and B, over a domain X to be $\Delta(A, B) := 1/2 \sum_{x \in X} |A(x) - B(x)|$. We say A and B are statistically indistinguishable, $A \overset{s}{\approx} B$, if $\Delta(A, B)$ is negligible, in some parameter appropriate to the domain.

2.2 Non-Malleable Codes and Reductions

Definition 1 (Coding Scheme). *[DPW10] A* Coding scheme, (E, D), *consists of a randomized encoding function* $\mathsf{E} : \{0,1\}^k \mapsto \{0,1\}^n$ *and a randomized decoding function* $\mathsf{D} : \{0,1\}^n \mapsto \{0,1\}^k \cup \{\bot\}$ *such that* $\forall x \in \{0,1\}^k, \Pr[\mathsf{D}(\mathsf{E}(x)) = x] = 1$ *(over randomness of* E *and* D*)*.

We note that this definition differs from the original one given in [DPW10], in that we allow the decoding to be randomized, while they required deterministic decoding. While this technically weakens our definition (and a code with deterministic decoding would be preferable), we feel that allowing randomized decoding fits

the spirit and motivation of non-malleable codes, and possibly is "the right" definition (which was simply not used before because it was not needed by previous constructions). More importantly, it may allow for a wider classes of functions.

This difference (allowing randomized decoding) also applies to the rest of the section, but all the previous results (in particular, Theorem 1) go through in exactly the same way, as long as we have independent randomness in all encoding and decoding.

Originally, non-malleable codes were defined in the following manner:

Definition 2 (Non-Malleable Code). *[ADKO15] Let $\mathcal{F}$ denote a family of tampering functions. Let $\mathsf{E} : B \to A$, $\mathsf{D} : A \to B$ be a coding scheme. For all $f \in \mathcal{F}$ and all $x \in B$ define:*

$$Tamper_x^f := \{c \leftarrow \mathsf{E}(x); \tilde{c} \leftarrow f(c); \tilde{x} \leftarrow D(\tilde{c}); \ output: \ \tilde{x}\}.$$

Then, (E, D) is an ε-non-malleable code with respect to $\mathcal{F}$, if there exists a distribution D_f over $\{0,1\}^k \cup \{\perp, same\}$ such that $\forall x \in B$, the statistical distance between

$$Sim_x^f := \{\tilde{x} \leftarrow D_f; \ output: \ x \ if \ \tilde{x} = same \ \& \ \tilde{x}, \ otherwise\},$$

and $Tamper_x^f$ is at most ε.

The above of definition has its origins in [DPW10]. Dziembowski, Pietrzak, and Wichs required the simulator to be efficient. Aggarwal et al. demonstrated that the above relaxation is, in fact, equivalent for deterministic decoding. Allowing decoding to be randomized does not affect their proof. For this reason, we will not concern ourselves with the efficiency of a simulator (or, equivalently, sampling relevant distributions) for the remainder of this paper.

Aggarwal et al. provide a simpler alternative to the above simulation-based definition, which they prove equivalent.[ADKO15] Their definition is based on the notion of non-malleable reduction, which we will use.

Definition 3 (Non-Malleable Reduction). *[ADKO15] Let $\mathcal{F} \subset A^A$ and $\mathcal{G} \subset B^B$ be some classes of functions. We say $\mathcal{F}$ reduces to $\mathcal{G}$, $(\mathcal{F} \Rightarrow \mathcal{G}, \varepsilon)$, if there exists an efficient (randomized) encoding function $\mathsf{E} : B \to A$, and an efficient (randomized) decoding function $\mathsf{D} : A \to B$, such that*

(a) $\forall x \in B, \Pr[\mathsf{D}(\mathsf{E}(x)) = x] = 1$ (over the randomness of E, D).
(b) $\forall f \in \mathcal{F}, \exists G : \ \forall x \in B, \ \Delta(\mathsf{D}(f(\mathsf{E}(x))); G(x)) \leq \varepsilon$, where G is a distribution over $\mathcal{G}$ and $G(x)$ denotes the distribution $g(x)$, where $g \leftarrow G$.

If the above holds, then (E, D) is an $(\mathcal{F}, \mathcal{G}, \varepsilon)$-non-malleable reduction.

Definition 4 (Non-Malleable Code). *[ADKO15] Let NM_k denote the set of trivial manipulation functions on k-bit strings, consisting of the identity function $id(x) = x$ and all constant functions $f_c(x) = c$, where $c \in \{0,1\}^k$.*

A coding scheme (E, D) defines an $(\mathcal{F}, k, \varepsilon)$-non-malleable code, if it defines an $(\mathcal{F}, \mathrm{NM}_k, \varepsilon)$-non-malleable reduction.

Aggarwal et al. also prove the following useful theorem for composing non-malleable reductions.

Theorem 1 (Composition). *[ADKO15] 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 note that the proof given in [ADKO15] goes through unchanged with randomized decoding.

2.3 Tampering Families

Definition 5 (Split-State Model). *[DPW10] The* split-state model, SS_k, *denotes the set of all functions:*

$$\{f = (f_1, f_2) : \ f(x) = (f_1(x_{1:k}) \in \{0,1\}^k, f_2(x_{k+1:2k}) \in \{0,1\}^k) \ for \ x \in \{0,1\}^{2k}\}.$$

Theorem 2 (Split-State Non-malleable Codes with Constant Rate). *[ADKO15] There exists an efficient, explicit* $(\mathrm{SS}_{O(k)}, k, 2^{-\Omega(k)})$ *non-malleable code,* $(\mathsf{E}_{\mathrm{SS}}, \mathsf{D}_{\mathrm{SS}})$.

We next define a class of local functions, where the number of input bits that can affect any output bit (input locality) and the number of output bits that depend on an input bit (output locality) are restricted. Loosely speaking, an input bit x_i affects the output bit y_j if for any boolean circuit computing f, there is a path in the underlying DAG from x_i to y_j. The formal definitions are below, and our notation follows that of [App14]

Definition 6. *We say that a bit x_i affects the boolean function f,*
if $\exists \{x_1, x_2, \cdots x_{i-1}, x_{i+1}, \cdots x_n\} \in \{0,1\}^{n-1}$ *such that,*
$f(x_1, x_2, \cdots x_{i-1}, 0, x_{i+1}, \cdots x_n) \neq f(x_1, x_2, \cdots x_{i-1}, 1, x_{i+1}, \cdots x_n)$.

Given a function $f = (f_1, \ldots, f_n)$ *(where each f_j is a boolean function), we say that input bit x_i affects output bit y_j, or that output bit y_j depends on input bit x_i, if x_i affects f_j.*

Definition 7 (Output Locality). *A function $f : \{0,1\}^n \to \{0,1\}^n$ is said to have* output locality m *if every output bit f_i is dependent on at most m input bits.*

Definition 8 (Input Locality). *A function $f : \{0,1\}^n \to \{0,1\}^n$ is said to have* input locality ℓ *if every input bit f_i is affects at most ℓ output bits.*

Definition 9 (Local Functions). *[App14] A function $f : \{0,1\}^n \to \{0,1\}^n$ is said to be (m, ℓ)-local, $f \in \mathrm{Local}_\ell^m$, if it has input locality ℓ and output locality m. We denote the class Local_n^m (namely no restriction on the input locality) by Local^m.*

The above notions can be generalized to function ensembles $\{f_n : \{0,1\}^n \to \{0,1\}^n\}_{n \in \mathbb{Z}}$ with the following corresponding locality bound generalizations: $\ell(n), m(n)$.

Recall that NC^0 is the class of functions where each output bit can be computed by a boolean circuit with constant depth and fan-in 2 (namely in constant parallel time). It is easy to see that $\mathsf{NC}^0 \subseteq \mathrm{Local}^{O(1)}$.

2.4 Reconstructable Probabilistic Encoding Scheme

Reconstructable Probabilistic Encoding (RPE) schemes were defined by Choi et al. (in an in-submission journal version of [CDMW08], as well as in [CDMW15]), extending a definition given by Decatur, Goldreich and Ron [DGR00]. Informally, this is an error correcting code, which has an additional secrecy property and reconstruction property. The secrecy property allows a portion of the output to be revealed without leaking any information about the encoded message. The reconstruction property allows, given a message and a partial codeword for it, to reconstruct a complete consistent codeword. Thus, this is a combination of error correcting code and secret sharing, similar to what has been used in the literature already starting with Ben-Or, Goldwasser, and Wigderson [BOGW88].

Definition 10 (Binary Reconstructable Probabilistic Encoding).
[CDMW08,CDMW15] We say a triple $(\mathsf{E}, \mathsf{D}, \mathsf{Rec})$ is a binary reconstructable probabilistic encoding scheme *with parameters $(k, n, c^{\mathsf{err}}, c^{\mathsf{sec}})$, where $k, n \in \mathbb{N}$, $0 < c^{\mathsf{err}}, c^{\mathsf{sec}} < 1$, if it satisfies the following properties:*

1. ***Error correction.*** $\mathsf{E} : \{0,1\}^k \to \{0,1\}^n$ *is an efficient probabilistic procedure, which maps a message $m \in \{0,1\}^k$ to a distribution over $\{0,1\}^n$. If we let $\mathcal{W}$ denote the support of E, any two strings in $\mathcal{W}$ are $2c^{\mathsf{err}}$-far. Moreover, D is an efficient procedure that given any $w' \in \{0,1\}^n$ that is $(1 - \epsilon)$-close to some string w in $\mathcal{W}$ for any $\epsilon \leq c^{\mathsf{err}}$, outputs w along with a consistent m.*

2. **Secrecy of partial views.** *For all $m \in \{0,1\}^k$ and all sets $S \subset [n]$ of size $\leq \lfloor c^{\mathsf{sec}} \cdot n \rfloor$, the projection of $\mathsf{E}(m)$ onto the coordinates in S, as denoted by $\mathsf{E}(m)|_S$, is identically distributed to the uniform distribution over $\{0,1\}^{\lfloor c^{\mathsf{sec}} n \rfloor}$.*

3. **Reconstruction from partial views.** *Rec is an efficient procedure that given any set $S \subset [n]$ of size $\leq \lfloor c^{\mathsf{sec}} \cdot n \rfloor$, any $I \in \{0,1\}^n$, and any $m \in \{0,1\}^k$, samples from the distribution $\mathsf{E}(m)$ with the constraint $\forall i \in S, \mathsf{E}(m)_i = I_i$.*

Choi et al. show that a construction of Decatur, Goldreich, and Ron [DGR00] meets the above requirements.

Lemma 1. *[CDMW08,CDMW15] For any $k \in \mathbb{N}$, there exists constants $0 < c^{\mathsf{rate}}, c^{\mathsf{err}}, c^{\mathsf{sec}} < 1$ such that there is a binary RPE scheme with parameters $(k, c^{\mathsf{rate}}k, c^{\mathsf{err}}, c^{\mathsf{sec}})$.*

Remark 1. To achieve longer encoding lengths ck, with the same c^{err} and c^{sec} parameters, one can simply pad the message to an appropriate length.

Specifically, Decatur, Goldreich and Ron [DGR00] construct a probabilistic encoding scheme that possesses the first two properties listed above. Moreover, since the construction they present, instantiates E with a linear error-correcting code, we have that property (3) holds. (Any linear error-correcting code has efficient reconstruction.)

These are the parameters we use here, but we believe it may be possible to achieve a better rate if we use parameters based on the recent result of Coretti et al. [CDTV15] (see also [CG14b]).

2.5 Boolean Function Restrictions

It will be convenient to use the following notation for Boolean function restrictions.

Definition 11 (Restriction). *For a vector $v \in \{0, 1, *\}^n$ and a Boolean function $f : \{0,1\}^n \to \{0,1\}^n$ the restriction of f to v, $\tilde{f}|_v$ is defined as $\tilde{f}|_v(x) = f(z)$ where,*

$$z_i = \begin{cases} x_i & v_i = * \\ v_i & v_i \in \{0,1\} \end{cases}$$

Let $f : D \to \{0,1\}^r$ be a function. Then, we denote by f_i the function which outputs the i-th output bit of f. Let $f : D \to \{0,1\}^r$ be a function and let $v \in \{0,1\}^r$ be a vector. Then, we denote by f_v the function which outputs all f_i such that $v_i = 1$.

2.6 Pseudorandom Generators of Space-Bounded Computation

Definition 12. *[Nis92] A generator $\mathsf{prg} : \{0,1\}^m \to (\{0,1\}^n)^k$ is a* pseudorandom generator *for $\mathrm{space}(w)$ and block size n with parameter ε if for every finite state machine, Q, of size 2^w over alphabet $\{0,1\}^n$ we have that*

$$|\Pr_y[Q \text{ accepts } y] - \Pr_x[Q \text{ accepts } \mathsf{prg}(x)]| \leq \varepsilon$$

where y is chosen uniformly at random in $(\{0,1\}^n)^k$ and x in $\{0,1\}^m$.

Theorem 3. *[Nis92] There exists a fixed constant $c > 0$ such that for any $w, k \leq cn$ there exists an (explicit) pseudorandom generator $\mathsf{prg} : \{0,1\}^{O(k)+n} \to \{0,1\}^{n2^k}$ for $\mathrm{space}(w)$ with parameter 2^{-cn}. Moreover, prg can be computed in polynomial time (in $n, 2^k$).*

3 Non-malleable Codes for $\mathrm{Local}_{\ell_o(n)}^{\ell_i(n)}$

Theorem 4. (E, D) *is a* $(\mathrm{Local}_{\ell_i(k)}^{\ell_o(k)} \Rightarrow \mathrm{SS}_k, \mathsf{negl}(k))$-*non-malleable reduction given the following parameters for* $\mathrm{Local}_{\ell_i(k)}^{\ell_o(k)}$ *(where* $c^{\mathsf{rate}}, c^{\mathsf{err}}, c^{\mathsf{sec}}$ *are taken from lemma 1):*

- $\ell_o \le \frac{c^{\mathsf{rate}} c^{\mathsf{sec}} k}{\log^2(k)}$.
- $\ell_i \le 12 \ell_o / c^{\mathsf{sec}}$.
- $n := c^{\mathsf{rate}} \frac{k^2}{\log^2(k)} + c^{\mathsf{rate}} k = O\left(\frac{k^2}{\log^2(k)}\right)$.

Putting together Theorem 4 with Theorems 1 and 2, we obtain the following.

Corollary 1. $(\mathsf{E} \circ \mathsf{E}_{\mathsf{SS}}, \mathsf{D}_{\mathsf{SS}} \circ \mathsf{D})$ *is a* $(\mathrm{Local}_\ell^\ell, k, \mathsf{negl}(k))$-*non-malleable code with rate* $\Theta(1/\ell)$, *where* $\ell = \tilde{O}(\sqrt{n})$.

Remark 2. The reduction presented below is, in fact, a $(\mathrm{XLocal}_\ell^\ell \Rightarrow \mathrm{SS}_k, \mathsf{negl}(k))$-non-malleable reduction, where $\ell = \tilde{O}(\sqrt{n})$ and $\mathrm{XLocal}_\ell^\ell$ is the following class of functions $f : \{0,1\}^{n_L+n_R} \to \{0,1\}^{n_L+n_R}$:

- For $i = 1, \ldots, n_L$, there are at most ℓ indices $j \in \{n_L + 1, \ldots, n_L + n_R\}$ such that the i-th input bit affects f_j. And, for $i = n_L + 1, \ldots, n_L + n_R$, there are at most ℓ indices $j \in \{1, \ldots, n_L\}$ such that the i-th input bit affects f_j.
- For $i = 1, \ldots, n_L$, there are at most ℓ indices $j \in \{n_L + 1, \ldots, n_L + n_R\}$ such that the f_i-th is affected by the j-th input bit. And, for $i = n_L + 1, \ldots, n_L + n_R$, there are at most ℓ indices $j \in \{1, \ldots, n_L\}$ such that the f_i-th is affected by the j-th input bit.

In other words, the reduction holds for a generalized variant of split state tampering where we only restrict locality with respect to the opposite side, and allow arbitrary locality *within* each side. n_L and n_R are the lengths of the left and right side codewords, respectively.

We construct an encoding scheme (E, D) summarized in Figure 1 and parametrized below. We then show that the pair (E, D) is an $(\mathrm{Local}_{\ell_i(k)}^{\ell_o(k)}, \mathrm{SS}_k, \mathsf{negl}(k))$-non-malleable reduction. This immediately implies that given a non-malleable encoding scheme $(\mathsf{E}^{\mathsf{ss}}, \mathsf{D}^{\mathsf{ss}})$ for class SS_k (where SS is the class of split-state functions), the encoding scheme scheme $\Pi = (\mathsf{E}^{\mathsf{bd}}, \mathsf{D}^{\mathsf{bd}})$, where $\mathsf{E}^{\mathsf{bd}}(m) := \mathsf{E}(\mathsf{E}^{\mathsf{ss}}(m))$ and $\mathsf{D}^{\mathsf{bd}}(s) := \mathsf{D}^{\mathsf{ss}}(\mathsf{D}(s))$ yields a non-malleable code against $\mathrm{Local}_{\ell_i(k)}^{\ell_o(k)}$.

We parametrize our construction for $\mathrm{Local}_{\ell_i(k)}^{\ell_o(k)} \Rightarrow \mathrm{SS}_k$ with the following:

- $(\mathsf{E}_L, \mathsf{D}_L)$ parametrized by $(k, n_L, c_L^{\mathsf{err}}, c_L^{\mathsf{sec}}) := (k, c^{\mathsf{rate}} k, c^{\mathsf{err}}, c^{\mathsf{sec}})$ where $c^{\mathsf{err}}, c^{\mathsf{sec}}, c^{\mathsf{rate}}$ are taken from lemma 1.
- $n_{\mathsf{check}} := \log^2(k)$.
- $\ell_{\mathsf{sec}} := \sqrt{\frac{c n_L}{n_{\mathsf{check}}}} = \Theta(\frac{\sqrt{k}}{\log(k)})$.
- $(\mathsf{E}_R, \mathsf{D}_R)$ parametrized by $(k, n_R, c_R^{\mathsf{err}}, c_R^{\mathsf{sec}}) := (k, \frac{\ell_o c^{\mathsf{rate}} k}{c^{\mathsf{sec}}}, c^{\mathsf{err}}, c^{\mathsf{sec}})$.
- $n := \ell_o c^{\mathsf{rate}} k + c^{\mathsf{rate}} k = O(\frac{k^2}{\log^2(k)})$.

Note that this setting of parameters is taken with our forthcoming reduction in mind. (See Corollary 2 and Theorem 5.) One may take any parametrization for which (a) such RPEs exist, (b) $(1 - c^{\mathsf{err}}/4)^{n_{\mathsf{check}}}$ is negligible in k, and (c) Observation 1 (below) is satisfied. For certain applications, parametrization other than ours may be advantageous.

Let $f(s^{\mathsf{L}}, s^{\mathsf{R}}) = (f^{\mathsf{L}}(s^{\mathsf{L}}, s^{\mathsf{R}}), f^{\mathsf{R}}(s^{\mathsf{L}}, s^{\mathsf{R}}))$, where $(s^{\mathsf{L}}, s^{\mathsf{R}}) \in \{0,1\}^{n_L} \times \{0,1\}^{n_R}$ and $f^{\mathsf{L}}(s^{\mathsf{L}}, s^{\mathsf{R}}) \in \{0,1\}^{n_L}$ and $f^{\mathsf{R}}(s^{\mathsf{L}}, s^{\mathsf{R}}) \in \{0,1\}^{n_R}$.

- Let $\mathcal{S}_{\mathsf{R} \to \mathsf{L}}$ denote the set of positions j such that input bit s_j^{R} affects the output of f^{L}.

Fig. 1. THE $(\mathsf{Local}_{\ell_i(k)}^{\ell_o(k)}, \mathsf{SS}, \mathsf{negl}(k))$-NON-MALLEABLE REDUCTION (E, D)

- Let $\mathcal{S}_{\mathsf{L} \to \mathsf{R}}$ denote the set of positions i such that input bit $\boldsymbol{s}_i^{\mathsf{L}}$ affects the output of f^{R}.
- For $J \subset [n_R]$, let $\mathcal{S}_{\mathsf{L} \to \mathsf{R}}^J$ denote the set of positions i such that input bit $\boldsymbol{s}_i^{\mathsf{L}}$ affects the output of f_j^{R} for some $j \in J$.
- For a set $R_{\mathsf{check}} \subseteq n_R$ of size n_{check}, let $\mathcal{S}_{\mathsf{check}}$ denote the set of positions i such that input bit $\boldsymbol{s}_i^{\mathsf{L}}$ affects the output of f_ℓ^{R} for some $\ell \in R_{\mathsf{check}}$.

The sets defined above are illustrated in Figure 2. We observe the following immediate facts about their sizes:

Observation 1 *For $f \in \mathsf{Local}_{\ell_i}^{\ell_o}$, we have the following:*

1. *There is some set $J^* \subset [n_R]$ such that $|J^*| = t$ and $|\mathcal{S}_{\mathsf{L} \to \mathsf{R}}^{J^*}| = 0$ (from now on, J^* denotes the lexicographically first such set).*
   (Since $|\mathcal{S}_{\mathsf{L} \to \mathsf{R}}| \leq \ell_i \cdot n_L \leq n_R - t$.)
2. *By choice of parameters $n_L, n_{\mathsf{check}}, c_L^{\mathsf{sec}}$, we have that $|\mathcal{S}_{\mathsf{check}}| \leq n_L \cdot c_L^{\mathsf{sec}}$.*
   (Since $\mathcal{S}_{\mathsf{check}} \leq \ell_o \cdot n_{\mathsf{check}}$.)
3. *By choice of parameters $n_L, n_R, c_R^{\mathsf{sec}}$, we have that $|\mathcal{S}_{\mathsf{R} \to \mathsf{L}}| \leq \ell_o \cdot n_L \leq n_R \cdot c_R^{\mathsf{sec}}$.*

Now, for every $f \in \mathsf{Local}_{\ell_i}^{\ell_o}$, we define the distribution G_f over SS_k. A draw from G_f is defined as follows:

- Choose a random subset $R_{\mathsf{check}} \subseteq [n_R]$ of size n_{check}.
- Choose vectors $I^{\mathsf{L}} \in \{0, 1\}^{n_L} \times \{*\}^{n_L}$, $I^{\mathsf{R}} \in \{*\}^{n_L} \times \{0, 1\}^{n_R}$ uniformly at random.
- Let J^* be the subset of $[n_R]$ as described in Observation 1.
- The split-state tampering function $g := (g_L, g_R) \in \mathsf{SS}_k$ has $I^{\mathsf{L}}, I^{\mathsf{R}}$ hardcoded into it and is specified as follows:
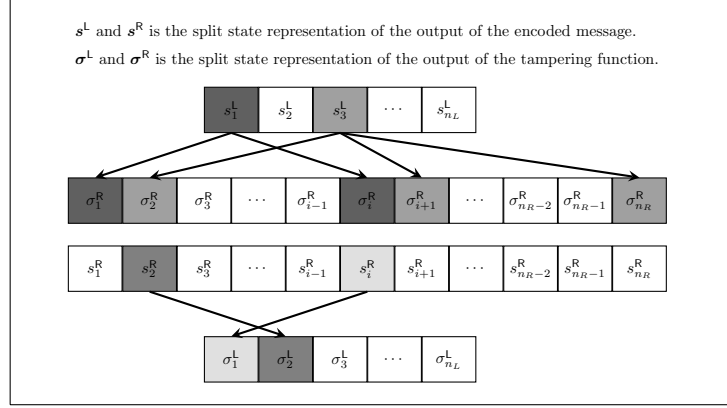 $g_L(\mathsf{L})$:

12

Fig. 2. The adversary chooses tampering function $f = (f^\mathsf{L}, f^\mathsf{R}) \in \mathsf{Local}_{\ell_i(k)}^{\ell_o(k)}$ which takes inputs $(\boldsymbol{s}^\mathsf{L}, \boldsymbol{s}^\mathsf{R})$ and produces outputs $(\boldsymbol{\sigma}^\mathsf{L}, \boldsymbol{\sigma}^\mathsf{R})$. The highlighted bits of $\boldsymbol{s}^\mathsf{L}$ and $\boldsymbol{s}^\mathsf{R}$ are the "bad" bits. E.g. note that bits s_2^R and s_i^R *affect* the output bits σ_2^L and σ_1^L respectively after f^L is applied to $(\boldsymbol{s}^\mathsf{L}, \boldsymbol{s}^\mathsf{R})$. Thus we add 2 and i to the set $\mathcal{S}_{\mathsf{R} \to \mathsf{L}}$. Similarly, the bits s_1^L and s_3^L *affect* the bits $\{\sigma_1^\mathsf{R}, \sigma_i^\mathsf{R}\}$ and the bits $\{\sigma_2^\mathsf{R}, \sigma_{i+1}^\mathsf{R}, \sigma_{n_R,}^\mathsf{R}\}$ respectively after the tampering function f^R is applied to $(\boldsymbol{s}^\mathsf{L}, \boldsymbol{s}^\mathsf{R})$. We therefore add 1 to the sets $\mathcal{S}_{\mathsf{L} \to \mathsf{R}}^1$ and $\mathcal{S}_{\mathsf{L} \to \mathsf{R}}^i$, while we add 3 to the sets $\mathcal{S}_{\mathsf{L} \to \mathsf{R}}^2, \mathcal{S}_{\mathsf{L} \to \mathsf{R}}^{i+1}$ and $\mathcal{S}_{\mathsf{L} \to \mathsf{R}}^{n_R}$. We also add both 1 and 3 to the set $\mathcal{S}_{\mathsf{L} \to \mathsf{R}}$.

1. (`apply tampering and plain decode on left`) Let $\boldsymbol{s}^\mathsf{L} := \mathsf{Rec}(\mathcal{S}_{\mathsf{check}}, I^\mathsf{L}, \mathsf{L})$. Let $(\sigma_1^\mathsf{L}, \ldots, \sigma_{n_L}^\mathsf{L}) := f^\mathsf{L}|_{I^\mathsf{R}}(\boldsymbol{s}^\mathsf{L})$. Compute $((w_1^\mathsf{L}, \ldots, w_{n_L}^\mathsf{L}), \widetilde{\mathsf{L}}) \leftarrow \mathsf{D}_L(\sigma_1^\mathsf{L}, \ldots, \sigma_{n_L}^\mathsf{L})$. If the decoding fails, set $\widetilde{\mathsf{L}} := \bot$.
2. (`output`) Output $\widetilde{\mathsf{L}}$.

$g_R(\mathsf{R})$:
1. (`apply tampering and decoding-check on right`) Let $\boldsymbol{s}^\mathsf{R} = (s_1^\mathsf{R}, \ldots, s_{n_R}^\mathsf{R}) := \mathsf{Rec}(\mathcal{S}_{\mathsf{R} \to \mathsf{L}}, I^\mathsf{R}, \mathsf{R})$. Let $(\sigma_1^\mathsf{R}, \ldots, \sigma_{n_R}^\mathsf{R}) := f^\mathsf{R}|_{I^\mathsf{L}}(\boldsymbol{s}^\mathsf{R})$. Define $\boldsymbol{\sigma}'^\mathsf{R} := \sigma_1'^\mathsf{R}, \ldots, \sigma_{n_R}'^\mathsf{R}$ as follows: Set $\sigma_\ell'^\mathsf{R} := \sigma_\ell^\mathsf{R}$ for $\ell \in [J^*]$. Set $\sigma_\ell'^\mathsf{R} := 0$ for $\ell \notin [J^*]$. Compute $((w_1^\mathsf{R}, \ldots, w_{n_R}^\mathsf{R}), \widetilde{\mathsf{R}}) \leftarrow \mathsf{D}_R(\sigma_1'^\mathsf{R}, \ldots, \sigma_t'^\mathsf{R})$. If the decoding fails or $(w_1^\mathsf{R}, \ldots, w_{n_R}^\mathsf{R})$ is not $c_R^{\mathsf{err}}/4$-close to $(\sigma_1^\mathsf{R}, \ldots, \sigma_{n_R}^\mathsf{R})$, then set $\widetilde{\mathsf{R}} := \bot$.
2. (`codeword-check on right`) For all $\ell \in R_{\mathsf{check}}$, check that $\sigma_\ell^\mathsf{R} = w_\ell^\mathsf{R}$. If the check fails, set $\widetilde{\mathsf{R}} := \bot$.
3. (`output`) Output $\widetilde{\mathsf{R}}$.
– Output $g = (g_L, g_R)$.

Whenever Rec is run above, we assume that enough positions are set by $\mathcal{S}$ such that there is only a single consistent codeword. If this is not the case, then additional positions are added to $\mathcal{S}$ from I^L, I^R, respectively.

By the definition of a non-malleable reduction (Definition 3), in order to complete the proof of Theorem 4, we must show that (E, D) have the following properties:

1. For all $x \in \{0,1\}^k$, we have $\mathsf{D}(\mathsf{E}(x)) = x$ with probability 1.
2. For all $f \in \mathsf{Local}_{\ell_i}^{\ell_o}$,
$$\Delta(\mathsf{D}(f(\mathsf{E}(x))); G_f(x)) \leq \mathsf{negl}(k),$$
where G_f is the distribution defined above.

Item (1) above is trivial and can be immediately verified. In the following, we prove Item (2) above by considering the following sequence of hybrid arguments for each function $f \in \mathsf{Local}_{\ell_i}^{\ell_o}$ (for the intermediate hybrids, we highlight the step in which they are different from the desired end distributions).

Hybrid H_0. This is the original distribution $\mathsf{D}(f(\mathsf{E}(x)))$

Hybrid H_1. H_1 corresponds to the distribution $\mathsf{D}'(f(\mathsf{E}(x)))$, where D' is defined as follows: $\mathsf{D}(\boldsymbol{\sigma} := (\boldsymbol{\sigma}^{\mathsf{L}}, \boldsymbol{\sigma}^{\mathsf{R}}))$:

1. (`plain decode on left`) Let $(\boldsymbol{\sigma}^{\mathsf{L}}, \boldsymbol{\sigma}^{\mathsf{R}}) := ([\sigma_i^{\mathsf{L}}]_{i \in [n_L]}, [\sigma_\ell^{\mathsf{R}}]_{\ell \in [n_R]})$. Compute $((w_1^{\mathsf{L}}, \ldots, w_{n_L}^{\mathsf{L}}), \mathsf{L}) \leftarrow \mathsf{D}_L(\sigma_1^{\mathsf{L}}, \ldots, \sigma_{n_L}^{\mathsf{L}})$. If the decoding fails, set $\mathsf{L} := \perp$.

2. (`decoding-check on right`) Define $\boldsymbol{\sigma}'^{\mathsf{R}} := \sigma_1'^{\mathsf{R}}, \ldots, \sigma_{n_R}'^{\mathsf{R}}$ as follows: Set $\sigma_\ell'^{\mathsf{R}} := \sigma_\ell^{\mathsf{R}}$ for $\ell \in J^*$ and $\sigma_\ell'^{\mathsf{R}} := 0$ for $\ell \notin J^*$, where $J^* \subseteq [n_R]$ is the lexicographically first set such that $|J^*| = t$ and $|\mathcal{S}_{\mathsf{L} \to \mathsf{R}}^{J^*}| = 0$. Compute $((w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}}), \mathsf{R}) \leftarrow \mathsf{D}_R(\sigma_1'^{\mathsf{R}}, \ldots, \sigma_{t_R}'^{\mathsf{R}})$. If the decoding fails or $(w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}})$ is not $c_R^{\mathsf{err}}/4$-close to $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{t_R}^{\mathsf{R}})$, set $\mathsf{R} := \perp$.

3. (`codeword-check on right`) For all $\ell \in R_{\mathsf{check}}$, check that $\sigma_\ell^{\mathsf{R}} = w_\ell^{\mathsf{R}}$. If the check fails, set $\mathsf{R} := \perp$.
4. (`output`) Output $x := (\mathsf{L}, \mathsf{R})$.

Note that the only difference between D and D' is that in `decoding-check on right`, $\boldsymbol{\sigma}^{\mathsf{R}}$ is decoded from J^*, instead of the first n_{check} positions.

Claim.
$$H_0 \overset{s}{\approx} H_1.$$

Proof. Let $\delta := \frac{c_R^{\mathsf{err}}}{4}$. Additionally, define

$$\rho(n_R, \delta, n_{\mathsf{check}}) := \frac{\binom{(1-\delta)n_R}{n_{\mathsf{check}}}}{\binom{n_R}{n_{\mathsf{check}}}}.$$

Notice that our parametrization of $n_{\mathsf{check}}, \delta$ yields $\rho(n_R, \delta, n_{\mathsf{check}}) = \mathsf{negl}(k)$.

$$\frac{\binom{(1-\delta)n_R}{n_{\mathsf{check}}}}{\binom{n_R}{n_{\mathsf{check}}}} = \frac{((1-\delta)n_R)! n_{\mathsf{check}}! (n_R - n_{\mathsf{check}})!}{n_{\mathsf{check}}! ((1-\delta)n_R - n_{\mathsf{check}})! n_R!} = \left(\frac{(1-\delta)n_R}{n_R}\right) \left(\frac{(1-\delta)n_R - 1}{n_R - 1}\right) \cdots \left(\frac{(1-\delta)n_R - n_{\mathsf{check}} + 1}{n_R - n_{\mathsf{check}} + 1}\right) \leq (1-\delta)^{n_{\mathsf{check}}},$$

where the last inequality follows due to the fact that for $i \in \{0, \ldots, n_{\mathsf{check}} - 1\}$, $\frac{(1-\delta)n_R - i}{n_R - i} \leq (1-\delta)$. Since $(1-\delta) < 1$ is a constant, we can set $n_{\mathsf{check}} = \omega(\log(k))$.

Note that correctness still holds for D' with probability 1.

We want to show that for every $\boldsymbol{\sigma} = (\boldsymbol{\sigma}^{\mathsf{L}}, \boldsymbol{\sigma}^{\mathsf{R}}) \leftarrow f(\mathsf{E}(x))$, $\mathsf{D}(\boldsymbol{\sigma}) = \mathsf{D}'(\boldsymbol{\sigma})$ with high probability, over the coins of D, D'.

Let $\mathsf{D} := (\mathsf{D}^{\mathsf{L}}, \mathsf{D}^{\mathsf{R}})$ (respectively, $\mathsf{D}' := (\mathsf{D}'^{\mathsf{L}}, \mathsf{D}'^{\mathsf{R}})$), where D^{R} (respectively, D'^{R}) correspond to the right output of the decoding algorithm. Notice that only decoding on the right changes. So, it suffices to show that for each $(\boldsymbol{\sigma}^{\mathsf{L}}, \boldsymbol{\sigma}^{\mathsf{R}})$ in the support of the distribution $f(\mathsf{E}(x))$,

$$\Pr[\mathsf{D}|_{\boldsymbol{\sigma}^{\mathsf{L}}}(\boldsymbol{\sigma}^{\mathsf{R}}) = \mathsf{D}'|_{\boldsymbol{\sigma}^{\mathsf{L}}}(\boldsymbol{\sigma}^{\mathsf{R}})] \geq 1 - \mathsf{negl}(n), \tag{3.1}$$

where the probabilities are taken over the coins of D, D'.

Let $\mathcal{W}$ denote the set of all valid codewords for the given reconstructable probabilistic encoding scheme with parameters $k, n_R, c_R^{\mathsf{err}}, c_R^{\mathsf{sec}}, \mathrm{GF}(2)$. For $x \in \mathrm{GF}(2)^{n_R}$, define its distance from $\mathcal{W}$ to be $d(x, \mathcal{W}) := \min_{w \in \mathcal{W}} d(x, w)$.

To analyze (3.1), we define the following set of instances (which intuitively corresponds to the set of instances on which both $\mathsf{D}|_{\boldsymbol{\sigma}^{\mathsf{L}}}$ and $\mathsf{D}'|_{\boldsymbol{\sigma}^{\mathsf{L}}}$ are likely to output $\perp$).

$$\Pi_\perp := \{\boldsymbol{\sigma}^{\mathsf{R}} \in \{0, 1\}^{n_R} \ d(\boldsymbol{\sigma}, \mathcal{W}) \geq \delta\}.$$

So, now consider the two cases:

14

- Suppose $\boldsymbol{\sigma}^{\mathsf{R}} \in \Pi_\perp$. Then, both $\mathsf{D}(\boldsymbol{\sigma}^{\mathsf{R}})$ and $\mathsf{D}'(\boldsymbol{\sigma}^{\mathsf{R}})$ will fail the codeword-check with probability $\geq 1 - \rho(n_R, \delta, n_{\mathsf{check}})$.
- Suppose $\boldsymbol{\sigma}^{\mathsf{R}} \notin \Pi_\perp$. Then, $\exists w \in \mathcal{W}$ such that $d(\boldsymbol{\sigma}^{\mathsf{R}}, w) \leq \delta$. Moreover, in *both* D and D' it must be the case that $\boldsymbol{\sigma}'^{\mathsf{R}}$ is $c^{\mathsf{err}}/2$-close to w. (Because $\delta + (n_R - t)/n_R \leq c^{\mathsf{err}}/2$). So both D and D' must decode to the *same* w. Fix a set of coins for D and D'. Therefore, when D and D' are run with the same coins, all comparisons made during the codeword-check are identical, and thus the probability (over the coins of D, D') that the codeword-check fails in D and D' is identical.

So for any $\boldsymbol{\sigma} = (\boldsymbol{\sigma}^{\mathsf{L}}, \boldsymbol{\sigma}^{\mathsf{R}})$, $\Delta(\{\mathsf{D}(\boldsymbol{\sigma})\}, \{\mathsf{D}'(\boldsymbol{\sigma})\}) = \Delta(\{\mathsf{D}^{\mathsf{R}}|_{\boldsymbol{\sigma}^{\mathsf{L}}}(\boldsymbol{\sigma}^{\mathsf{R}})\}, \{\mathsf{D}'^{\mathsf{R}}|_{\boldsymbol{\sigma}^{\mathsf{L}}}(\boldsymbol{\sigma}^{\mathsf{R}})\}) \leq \rho(n_R, \delta, n_{\mathsf{check}})$. Therefore, $\Delta(\{\mathsf{D}(f(\mathsf{E}(x)))\}, \{\mathsf{D}'(f(\mathsf{E}(x)))\}) \leq \rho(n_R, \delta, n_{\mathsf{check}})$.

Hybrid H_2. H_2 corresponds to the distribution $G'(x)$, where G'_f is a distribution over functions $g' = (g'_L, g'_R)$ defined as follows:

- Choose a random subset $R_{\mathsf{check}} \subseteq [n_R]$ of size n_{check}.

$\boxed{\text{– Choose vectors } I^{\mathsf{L}} \in \{0,1\}^{n_L}, I^{\mathsf{R}} \in \{0,1\}^{n_R} \text{ in the following way: } I^{\mathsf{L}} \leftarrow \mathsf{E}_L(\mathsf{L}), \, I^{\mathsf{R}} \leftarrow \mathsf{E}_R(\mathsf{R}).}$

- Let J^* be the subset of $[n_R]$ as described in Observation 1.
- The split-state tampering function $g := (g_L, g_R) \in \mathrm{SS}_k$ has $I^{\mathsf{L}}, I^{\mathsf{R}}$ hardcoded into it and is specified as follows:

  $g_L(\mathsf{L})$:
  1. (`apply tampering and plain decode on left`) Let $\boldsymbol{s}^{\mathsf{L}} := \mathsf{Rec}(\mathcal{S} := \mathcal{S}_{\mathsf{check}}, I^{\mathsf{L}}, \mathsf{L})$. Let $(\sigma_1^{\mathsf{L}}, \ldots, \sigma_{n_L}^{\mathsf{L}}) := f^{\mathsf{L}}|_{I^{\mathsf{R}}}(\boldsymbol{s}^{\mathsf{L}})$. Compute $((w_1^{\mathsf{L}}, \ldots, w_{n_L}^{\mathsf{L}}), \widetilde{\mathsf{L}}) \leftarrow \mathsf{D}_L(\sigma_1^{\mathsf{L}}, \ldots, \sigma_{n_L}^{\mathsf{L}})$. If the decoding fails, set $\widetilde{\mathsf{L}} := \perp$.
  2. (`output`) Output $\widetilde{\mathsf{L}}$.

  $g_R(\mathsf{R})$:
  1. (`apply tampering and decoding-check on right`) Let $\boldsymbol{s}^{\mathsf{R}} = (s_1^{\mathsf{R}}, \ldots, s_{n_R}^{\mathsf{R}}) := \mathsf{Rec}(\mathcal{S}_{\mathsf{R}\to\mathsf{L}}, I^{\mathsf{R}}, \mathsf{R})$. Let $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{n_R}^{\mathsf{R}}) := f^{\mathsf{R}}|_{I^{\mathsf{L}}}(\boldsymbol{s}^{\mathsf{R}})$. Define $\boldsymbol{\sigma}'^{\mathsf{R}} := \sigma_1'^{\mathsf{R}}, \ldots, \sigma_{n_R}'^{\mathsf{R}}$ as follows: Set $\sigma_\ell'^{\mathsf{R}} := \sigma_\ell^{\mathsf{R}}$ for $\ell \in [J^*]$. Set $\sigma_\ell'^{\mathsf{R}} := 0$ for $\ell \notin [J^*]$. Compute $((w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}}), \widetilde{\mathsf{R}}) \leftarrow \mathsf{D}_R(\sigma_1'^{\mathsf{R}}, \ldots, \sigma_t'^{\mathsf{R}})$. If the decoding fails or $(w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}})$ is not $c_R^{\mathsf{err}}/4$-close to $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{n_R}^{\mathsf{R}})$, then set $\widetilde{\mathsf{R}} := \perp$.
  2. (`codeword-check on right`) For all $\ell \in R_{\mathsf{check}}$, check that $\sigma_\ell^{\mathsf{R}} = w_\ell^{\mathsf{R}}$. If the check fails, set $\widetilde{\mathsf{R}} := \perp$.
  3. (`output`) Output $\widetilde{\mathsf{R}}$.
- Output $g = (g_L, g_R)$.

Note that the only difference between G_f and G'_f is that $I^{\mathsf{L}} \leftarrow \mathsf{E}_L(\mathsf{L}), I^{\mathsf{R}} \leftarrow \mathsf{E}_R(\mathsf{R})$ are chosen honestly, instead of being chosen uniformly at random. Furthermore, note that $g' = (g'_L, g'_R)$ are not split-state, since g'_L depends on I^{R} and g'_R depends on I^{L}.

Claim.
$$H_1 \equiv H_2.$$

The claim can be verified by inspection.

Hybrid H_3. Hybrid H_3 is simply the distribution $G_f(x)$, defined previously.

Claim.
$$H_2 \equiv H_3.$$

Note that the result of f^{R} only depends on the bits in J^* and R_{check}. Moreover, $f^{\mathsf{R}}_{\chi_{J^* \cup R_{\mathsf{check}}}}$ only depends on on $\boldsymbol{s}^{\mathsf{R}}, [s_i^{\mathsf{L}}]_{i \in \mathcal{S}_{\mathsf{check}}}$. Moreover, note that f^{L} depends only on $\boldsymbol{s}^{\mathsf{L}}, [s_i^{\mathsf{R}}]_{i \in \mathcal{S}_{\mathsf{R}\to\mathsf{L}}}$. Since by Observation 1, we have that $|\mathcal{S}_{\mathsf{check}}| \leq n_L \cdot c_L^{\mathsf{sec}}$ and $|\mathcal{S}_{\mathsf{R}\to\mathsf{L}}| \leq n_R \cdot c_R^{\mathsf{sec}}$, the claim follows from the secrecy property of the reconstructable probabilistic encoding scheme.

3.1 Extending to Leaky Local

The construction from Section 3 is actually secure against a slightly larger class of tampering functions beyond $\mathrm{Local}_{\ell_i}^{\ell_o}$ functions, which we call LL, or "Leaky Local." Notice that the parameters given above (as in observation 1) in fact yield:

1. $|\mathcal{S}_{\mathsf{L}\to\mathsf{R}}^{J^*}| + |\mathcal{S}_{\mathsf{check}}| = |\mathcal{S}_{\mathsf{check}}| \le n_L \cdot \frac{c_L^{\mathsf{sec}}}{3}$.
2. $|\mathcal{S}_{\mathsf{R}\to\mathsf{L}}^+| \le \ell_o \cdot n_L \le n_R \cdot \frac{2 c_R^{\mathsf{sec}}}{3}$.

It is not too hard to see that we can leak $1/3$ of the security threshold, on both the left and right, to a tampering adversary. Given this leakage, the adversary can then select a tampering function from the subset of Local^{ℓ_o} where all but a fraction of the first n_L bits have input locality ℓ_i. Note that the input locality restrictions are only needed on the left portions of codewords in the above proof. We formalize this new class of tampering functions as follows.

Definition 13. *Let* $\mathrm{LL} \subseteq \{\{0,1\}^{n_L} \times \{0,1\}^{n_R} \to \{0,1\}^{n_L} \times \{0,1\}^{n_R}\}$, *Leaky Local, be the set of functions* $\{\psi_{f,h_1,h_2}\}$, *parametrized by functions* (f, h_1, h_2), *where* $\psi_{f,h_1,h_2}(\boldsymbol{s}^{\mathsf{L}}, \boldsymbol{s}^{\mathsf{R}}) := C_{univ}(f(h_1(\boldsymbol{s}^{\mathsf{L}}), h_2(\boldsymbol{s}^{\mathsf{R}})), \boldsymbol{s}^{\mathsf{L}}, \boldsymbol{s}^{\mathsf{R}})$, *$f$ outputs a circuit C and C_{univ} is a universal circuit that computes the output of the circuit C on input* $(\boldsymbol{s}^{\mathsf{L}}, \boldsymbol{s}^{\mathsf{R}})$. *Moreover, we require that f, h_1, h_2 have the following form:*

- *On input $\boldsymbol{s}^{\mathsf{L}} \in \{0,1\}^{n_L}$, h_1 outputs a subset of $c_L^{\mathsf{err}}/3$ of its input bits.*
- *On input $\boldsymbol{s}^{\mathsf{R}} \in \{0,1\}^{n_R}$, h_2 outputs a subset of $c_R^{\mathsf{err}}/3$ of its input bits.*
- *On input $h_1(\boldsymbol{s}^{\mathsf{L}}), h_2(\boldsymbol{s}^{\mathsf{L}}) \in \{0,1\}^{c_L^{\mathsf{err}}/3} \times \{0,1\}^{c_R^{\mathsf{err}}/3}$, f outputs a circuit $C : \{0,1\}^{n_L} \times \{0,1\}^{n_R} \to \{0,1\}^{n_L} \times \{0,1\}^{n_R}$, where C has output-locality ℓ_o. Of the first n_L input bits, all but at most $c_L^{\mathsf{err}}/3$-fraction have input-locality at most ℓ_i.*

The following corollary can be easily verified.

Corollary 2. $(\mathsf{E} \circ \mathsf{E}_{\mathsf{SS}}, \mathsf{D}_{\mathsf{SS}} \circ \mathsf{D})$ *is an* $(\mathrm{LL}, \mathrm{SS}_k, \mathsf{negl}(k))$*-non-malleable reduction.*

4 Extending to $\mathrm{Local}^{m(n)}$

We now state our theorem for $\mathrm{Local}^{m(n)}$ tampering functions, or bounded fan-in bounded-depth circuits.

Theorem 5. $(\mathsf{E}', \mathsf{D}')$ *is a* $(\mathrm{Local}^{\ell_o'} \Rightarrow \mathrm{LL}, \mathsf{negl}(n))$*-non-malleable reduction given the following parameters for* $\mathrm{Local}^{\ell_o'}$:

- $\ell_o' := c^{\mathsf{sec}}/12 \cdot \ell_i$, *where ℓ_i is the input locality of LL,*
- $\mathsf{E}' : \{0,1\}^n \to \{0,1\}^N$, *where $N = n_{in} + 2n - n_L$, and $r = \log^4(k)$, where n is the output length of LL and n_L is the length of the left output of LL.*

We construct an encoding scheme $(\mathsf{E}', \mathsf{D}')$ summarized in Figure 3 and parametrized below. In brief, our encoding simply distributes the bits of the left input pseudorandomly in a string comparable in length to the right input. We then append a short description of where the encoding is hiding, a seed to pseudorandom generator.

We then show that the pair $(\mathsf{E}', \mathsf{D}')$ is an $(\mathrm{Local}^{\ell_o'}, \mathrm{LL}, \mathsf{negl}(n))$-non-malleable reduction. Combined with our previous construction, this immediately implies that given a non-malleable encoding scheme $(\mathsf{E}^{\mathsf{ss}}, \mathsf{D}^{\mathsf{ss}})$ for SS_k, the encoding scheme scheme $\widehat{\Pi} = (\widehat{\mathsf{E}^{\mathsf{bd}}}, \widehat{\mathsf{D}^{\mathsf{bd}}})$, where $\widehat{\mathsf{E}^{\mathsf{bd}}}(m) := \mathsf{E}'(\mathsf{E}(\mathsf{E}^{\mathsf{ss}}(m)))$ and $\widehat{\mathsf{D}^{\mathsf{bd}}}(\boldsymbol{s}) := \mathsf{D}^{\mathsf{ss}}(\mathsf{D}(\mathsf{D}'(\boldsymbol{s})))$ yields the following corollary, a non-malleable code against $\mathrm{Local}^{\ell_o'}$.

Corollary 3. $(\mathsf{E}', \mathsf{D}')$ *yields, with previous results, a* $(\mathrm{Local}^{\tilde{O}(\sqrt{n})}, k, \mathsf{negl}(k))$*-non-malleable reduction with sublinear rate, where $n = \Theta(\frac{k^2}{\log^2(k)})$.*

Let prg be a pseudorandom generator for space bounded computations (see Definition 12), with inputs of length r and outputs of length $\log(\tau) \cdot \tau$.

Let $G(\zeta)$ be defined as follows:

1. Compute $y := \mathsf{prg}(\zeta)$.
2. Divide pseudorandom tape y into blocks of bit strings $y_1, \ldots, y_\tau$. Let ϕ be the randomized function that chooses a bit $b \in \{0, 1\}$ with bias $p := 3n_L/2\tau$. For $i \in [\tau]$, let $\rho_i = \phi(y_i)$, where y_i is the explicit randomness of ϕ. Let $\rho = \rho_1, \ldots, \rho_\tau$. Let num denote the number of positions of ρ that are set to 1.
3. If $\mathsf{num} < n_L$, set $\rho := 1_L^n 0^{\tau - n_L}$.
4. Otherwise, flip all but the first n_L 1's in ρ to 0.
5. Output ρ.

Let $\mathsf{E}' : \{0, 1\}^n \to \{0, 1\}^N$ and $\mathsf{D}' : \{0, 1\}^N \to \{0, 1\}^n$.

$\mathsf{E}'(x^{\mathsf{L}} := x_1^{\mathsf{L}}, \ldots, x_{n_L}^{\mathsf{L}}, x^{\mathsf{R}})$:

1. Choose $\zeta \leftarrow \{0, 1\}^r$ uniformly at random. Choose $\zeta \leftarrow \{0, 1\}^r$ uniformly at random. Compute $\rho := G(\zeta)$.
2. For $j \in [\mathsf{num}]$, let pos_j denote the j-th position i such that $\rho_i = 1$.
3. Let $X^{\mathsf{L}} \in \{0, 1\}^\tau$ be defined in the following way: For $j \in [n_L]$, $X_{\mathsf{pos}_j}^{\mathsf{L}} := x_j^{\mathsf{L}}$. In all other locations, X_i^{L} is set uniformly at random.
4. Output the encoding $(\zeta, X^{\mathsf{L}}, x^{\mathsf{R}})$.

$\mathsf{D}'(Z := (\widetilde{\zeta}, \widetilde{X}^{\mathsf{L}}, \widetilde{x}^{\mathsf{R}}))$:

1. (Recover $\widetilde{\rho}$) Let $\widetilde{\rho} := G(\widetilde{\zeta})$. Let $\widetilde{\mathsf{num}} \geq n_L$ denote the number of ones in $\widetilde{\rho} := \widetilde{\rho}_1, \ldots, \widetilde{\rho}_\tau$.
2. (Recover x) For $j \in [\widetilde{\mathsf{num}}]$, let pos_j denote the j-th position i such that $\widetilde{\rho}_i = 1$.
3. Let $\widetilde{x}_j^{\mathsf{L}} \in \{0, 1\}^{n_L}$ be defined in the following way: For $j \in [\min(\widetilde{\mathsf{num}}, n_L)]$, $\widetilde{x}_j^{\mathsf{L}} := \widetilde{X}_{\mathsf{pos}_j}^{\mathsf{L}}$.
4. (output) Output $(\widetilde{x}^{\mathsf{L}}, \widetilde{x}^{\mathsf{R}})$.

Fig. 3. THE $(\mathsf{Local}^{\ell_o'}, \mathsf{LL}, \mathsf{negl}(n))$-NON-MALLEABLE REDUCTION $(\mathsf{E}', \mathsf{D}')$

Remark 3. As before, the encoding scheme presented below is independent on the left and right. Therefore, our reduction holds for not just for $\mathsf{Local}^{\ell_o'}$ but additionally any split-state function, independent on each side, trivially.

We parametrize our construction for $\mathsf{Local}^{\ell_o'} \Rightarrow \mathsf{LL}$ with the following:

– $r := \log^4(k)$
– $\tau := 2(n - n_L)$, where n is the length of the output of LL and n_L is the length of the left output of LL.

Now, for every $\mu \in \mathsf{Local}^{\ell_o'}$ where $\mu(\zeta, X^\mathsf{L}, x^\mathsf{R}) := (\mu^\zeta(\zeta, X^\mathsf{L}, x^\mathsf{R}), \mu^\mathsf{L}(\zeta, X^\mathsf{L}, x^\mathsf{R}), \mu^\mathsf{R}(\zeta, X^\mathsf{L}, x^\mathsf{R}))$ we define the distribution G_μ over LL. A draw from G_μ is defined as follows:

– Choose $\zeta \leftarrow \{0,1\}^r$ uniformly at random. Compute $y := \mathsf{prg}(\zeta)$, where $y = y_1, \ldots, y_\tau$. For $i \in [\tau]$, compute Compute $\rho_i := \phi(y_i)$.
– If ρ has less than n_L number of ones, then set h_1, h_2, f all to the constant function 0.
– Otherwise, choose vector $I^\mathsf{L} \in \{0,1\}^{\tau + n_R}$ such that $\forall i$ such that $1 \le i \le \tau$ if $\rho_i = 1$ then $I_i^\mathsf{L} = *$ and otherwise, I_i^L is chosen uniformly at random.
– The function h_1 is defined as follows: h_1 outputs the bits in input x^L that affect the output bits of μ^ζ (at most $r \cdot \ell_o' \le c_L^\mathsf{sec}/3 \cdot n_L$).
– The function h_2 is defined as follows: h_2 outputs the bits in x^R that affect the output bits of μ^ζ (at most $r \cdot \ell_o' \le c_R^\mathsf{sec}/3 \cdot n_R$).
– The function f is defined as follows:
  • f computes $\widetilde{\zeta}$, given ζ and the output of h_1, h_2.
  • f computes $\widetilde{y} := \mathsf{prg}(\widetilde{\zeta})$, where $\widetilde{y} = \widetilde{y}_1, \ldots, \widetilde{y}_\tau$.
  • For $i \in [\tau]$, f computes $\widetilde{\rho}_i := \phi(\widetilde{y}_i)$.
  • Let $\widetilde{\rho}^* \in \{0,1\}^\tau$ be defined as follows: For $i \in [\mathsf{pos}^*], \widetilde{\rho}^* = \widetilde{\rho}$; for $\mathsf{pos}^* < i \le \tau, \widetilde{\rho}^* = 0$, where pos^* is the index of the n_L-th one in $\widetilde{\rho}$ (and is set to τ if no such index exists).
  • Let $\mu^{\mathsf{L}, \zeta}$ (resp. $\mu^{\mathsf{R}, \zeta}$) correspond to the function $\mu^\mathsf{L}(\zeta, X^\mathsf{L}, x^\mathsf{R})$ (resp. $\mu^\mathsf{R}(\zeta, X^\mathsf{L}, x^\mathsf{R})$), which has ζ hardcoded in it.
  • Let C be the circuit corresponding to the following restriction: $((\mu^{\mathsf{L}, \zeta}|_{I^\mathsf{L}})_{\widetilde{\rho}^*}, \mu^{\mathsf{R}, \zeta}|_{I^\mathsf{L}})$.
  • If C is in LL, then f outputs C. Otherwise, f outputs the constant function 0.

By the definition of a non-malleable reduction (Definition 3), in order to complete the proof of Theorem 5, we must show that $(\mathsf{E}', \mathsf{D}')$ has the following properties:

1. For all $x \in \{0,1\}^n$, we have $\mathsf{D}'(\mathsf{E}'(x)) = x$ with probability 1.
2. For all $\mu \in \mathsf{Local}^{\ell_o'}$,
$$\Delta(\mathsf{D}'(\mu(\mathsf{E}'(x))); G_\mu(x)) \le \mathsf{negl}(n),$$
where G_μ is the distribution defined above.

Item (1) above is trivial and can be immediately verified.

In the following, we prove Item (2), above, by noting that the statistical distance $\Delta(\mathsf{D}'(\mu(\mathsf{E}'(x))); G_\mu(x))$ is upper bounded by the probability that either ρ does not contain at least n_L number of ones or C is not in LL.

We first argue that if ρ is chosen uniformly at random, then the probability that either of these events occurs is negligible and then show that the same must be true when ρ is chosen via a PRG with appropriate security guarantees.

Clearly, by multiplicative Chernoff bounds, if ρ is chosen uniformly at random, then the probability that ρ contains less than n_L ones is negligible. We now show that the probability that $C \notin \mathsf{LL}$ is negligible. If $C \notin \mathsf{LL}$, it means that more than $c_L^\mathsf{sec}/3$ number of positions i in X^L are such that (1) X_i^L has "high input locality" (i.e. input locality greater than $12/c_L^\mathsf{sec} \cdot \ell_o' = \ell_i$) (2) $\rho_i = 1$.

Since the adversary first specifies the tampering function μ, all positions in X^{L} with "high input locality" are determined. Note that, by choice of parameters (since $\tau \geq N/2$), there can be at most $c_L^{\mathsf{sec}} \cdot \tau/6$ number of positions in X^{L} with "high input locality". Since $p = 3n_L/2\tau$, we *expect* $c_L^{\mathsf{sec}} \cdot n_L/4$ number of positions i in X^{L} where (1) X_i^{L} has "high input locality" and (2) $\rho_i = 1$. Therefore, by multiplicative Chernoff bounds, the probability that more than $c_L^{\mathsf{sec}} \cdot n_L/3$ number of positions i in X^{L} are such that (1) X_i^{L} has "high input locality" and (2) $\rho_i = 1$ is negligible.

We now argue that these events must also occur with negligible probability when ρ is pseudorandom. Assume the contrary, then the following is a distinguisher T that can distinguish truly random strings y from strings $y := \mathsf{prg}(\zeta)$ with non-negligible probability.

T is a circuit that has a string $w \in \{0,1\}^\tau$ hardwired into it (non-uniform advice). w corresponds to the high input locality positions determined by the tampering function μ that was chosen by the adversary A. Intuitively, w is the string that causes A to succeed in breaking security of the non-malleable code with highest probability.

On input $y = y_1, \ldots, y_\tau$ (where either $y := \mathsf{prg}(\zeta)$ or y is chosen uniformly at random), $T(y)$ does the following:

1. Set $count_1 = 0$, $count_2 = 0$.
2. For $i = 1$ to τ:
   (a) Run $\phi(y_i)$ to obtain ρ_i.
   (b) If $\rho_i = 1$, set $count_2 := count_2 + 1$
   (c) If $\rho_i = 1$ and $w_i = 1$, set $count_1 := count_1 + 1$.
3. If $count_1 > c_L^{\mathsf{sec}} \cdot n_L/3$ or $count_2 < n_L$, output 0. Otherwise, output 1.

T can clearly be implemented by a read-once, Finite State Machine (FSM) with $2^{O(\log^2(\tau))}$ number of states. However, note that by Theorem 3, prg is a pseudorandom generator for space $\log^3(k)$ with parameter $2^{-\log^3(k)}$. Thus, existence of distinguisher T as above, leads to contradiction to the security of the Nisan PRG.

5   Achieving Resilience against $o(n/\log n)$ Output Locality

We now present the proof of our final main theorem. The encoding scheme we use is simply the composition of the two schemes presented previously with slightly different parameters The only substantial difference is in the analysis.

Theorem 6. $(\mathsf{E}', \mathsf{D}')$ *is a* $(\mathsf{Local}^{\ell_o} \Rightarrow SS, \mathsf{negl}(n))$*-non-malleable reduction given the following parameters for* Local^{ℓ_o}*:*

- $\ell_o = o(n/\log(n))$.
- $\mathsf{E}' : \{0,1\}^{2k} \to \{0,1\}^n$, *where* $n = O(\ell_o k)$.

Putting together Theorem 6 with Theorems 1 and 2, we obtain the following.

Corollary 4. $(\mathsf{E} \circ \mathsf{E}_{\mathsf{SS}}, \mathsf{D}_{\mathsf{SS}} \circ \mathsf{D})$ *is a* $(\mathsf{Local}^{\ell_o}, k, \mathsf{negl}(k))$*-non-malleable code with rate* $\Theta(1/\ell_o)$*, where* $\ell_o = o(n/\log n)$.

Remark 4. Note that $n = \Theta(\ell_o k)$. Thus, for resilience against $\ell_o = n^{1-\varepsilon}$ our codes is of polynomial length $n = k^{1/\varepsilon}$.

We construct an encoding scheme (E, D) summarized in Figure 1 and parametrized below. We then show that the pair (E, D) is a $(\mathsf{Local}^{\ell_o}, SS_k, \mathsf{negl}(k))$-non-malleable reduction.

We parameterize our construction for $\mathsf{Local}^{\ell_o} \Rightarrow SS_k$ with the following:

- $t := \lceil n_R(1 - c_R^{\mathsf{err}}/4) \rceil$.

- $c^{\mathsf{dec}} := 1 - \frac{t}{n_R}$.
- $\delta := \frac{c^{\mathsf{sec}}}{9}$.
- $(\mathsf{E}_L, \mathsf{D}_L)$ parametrized by $(k, n_L, c_L^{\mathsf{err}}, c_L^{\mathsf{sec}}) := (k, c^{\mathsf{rate}}k, c^{\mathsf{err}}, c^{\mathsf{sec}})$ where $c^{\mathsf{err}}, c^{\mathsf{sec}}, c^{\mathsf{rate}}$ are taken from lemma 1.
- $n_{\mathsf{check}} := k$.
- $n_R := \lceil \frac{2\ell_o c^{\mathsf{rate}}k}{\delta c^{\mathsf{dec}} c^{\mathsf{sec}}} \rceil$
- $(\mathsf{E}_R, \mathsf{D}_R)$ parametrized by $(k, n_R, c_R^{\mathsf{err}}, c_R^{\mathsf{sec}}) := (k, n_R, c^{\mathsf{err}}, c^{\mathsf{sec}})$.
- $r := k$
- $\tau := \lceil \frac{30\ell_o c^{\mathsf{rate}}k}{(c^{\mathsf{sec}})^2 c^{\mathsf{dec}}} + \frac{1}{\delta} \rceil$
- $n := r + \tau + n_R = O(\ell_o k)$.

Proof Overview. To prove the theorem, we analyze the composed encoding schemes as a single reduction. As mentioned in the introduction, the idea is to use the PRG to "free up" the restrictions relating the size of the left RPE (previously denoted by n_L) and ℓ_o that is an artifact of the piecewise analysis.

Recall that our encoding scheme is comprised of three blocks: (1) the PRG seed, (2) the "hidden" left side encoding, and (3) the right side encoding. First, (as in the previous section) we claim that a number of good things happen if the left side is "hidden" in a large block in a truly random way. Namely, we have that, with respect to the tampering function, only a small fraction of bits in the hidden left-side RPE is either (1) of high input locality, (2) effects bits in the right-side's consistency check or (3) effects the PRG seed used in decoding. (1) Implies that there exists a "safe" subset to simulate decoding from (as before), and (2) and (3) allow us to relax the bounds on locality. Next, we use a hybrid argument to essentially disconnect influence between the 3 blocks of our encoding (that is dependent on the underlying message, (L, R)).

Proof. We will consider the "Left" side of the encoding to be (ζ, X^{L}) and the "Right" side to be s^{R}.

Let $f(\zeta, X^{\mathsf{L}}, s^{\mathsf{R}}) = (f^{\mathsf{L}}(\zeta, X^{\mathsf{L}}, s^{\mathsf{R}}), f^{\mathsf{R}}(\zeta, X^{\mathsf{L}}, s^{\mathsf{R}}))$, where $(\zeta, X^{\mathsf{L}}, s^{\mathsf{R}}) \in \{0,1\}^{r+\tau}\{0,1\}^{n_R}$ and $f^{\mathsf{L}}(\zeta, X^{\mathsf{L}}, s^{\mathsf{R}}) \in \{0,1\}^{r+\tau}$ and $f^{\mathsf{R}}(\zeta, X^{\mathsf{L}}, s^{\mathsf{R}}) \in \{0,1\}^{n_R}$. Furthermore, let $f^{\mathsf{L}} = (f_\zeta, f_X)$ where $f_\zeta : \{0,1\}^{r+\tau+n_R} \to \{0,1\}^r$ and $f_X : \{0,1\}^{r+\tau+n_R} \to \{0,1\}^\tau$

- Let $U = \{i \in [\tau] : \rho_i = 1\}$ denote the (relative) locations of s^{L}.
- Let $\mathcal{S}_{\mathsf{R} \to \mathsf{L}}$ denote the set of positions j such that input bit s_j^{R} affects the output of f^{L}.
- Let $\mathcal{S}_{\mathsf{L} \to \mathsf{R}}$ denote the set of positions i such that input bit s_i^{L} affects the output of f^{R}.
- For $J \subset [n_R]$, let $\mathcal{S}_{\mathsf{L} \to \mathsf{R}}^J$ denote the set of positions i such that input bit s_i^{L} affects the output of f_j^{R} for some $j \in J$.
- For a set $R_{\mathsf{check}} \subseteq n_R$ of size n_{check}, let $\mathcal{S}_{\mathsf{check}}$ denote the set of positions i such that embedded input bit s_i^{L} affects the output of f_ℓ^{R} for some $\ell \in R_{\mathsf{check}}$.
- For a set $R_{\mathsf{check}} \subseteq n_R$ of size n_{check}, let $\mathcal{S}_{\mathsf{check}}^+$ denote the set of positions i such that input bit X_i^{L} affects the output of f_ℓ^{R} for some $\ell \in R_{\mathsf{check}}$.
- For a set $R_{\mathsf{check}} \subseteq n_R$ of size n_{check}, let $\mathcal{S}_{\mathsf{check}}$ denote the set of positions i such that embedded input bit s_i^{L} affects the output of f_ℓ^{R} for some $\ell \in R_{\mathsf{check}}$.
- Let $S_{\mathsf{in}}^+(\ell_i)$ denote the set of i such that X_i^{L} has input locality greater than ℓ_i.
- Let $S_{\mathsf{in}}(\ell_i)$ denote the set of i such that s_i^{L} has input locality greater than ℓ_i.
- Let $\mathcal{S}_{\mathsf{L} \to \zeta}^+$ denote the set of positions i such that input bit X_i^{L} affects the output of f_ζ.
- Let $\mathcal{S}_{\mathsf{L} \to \zeta}$ denote the set of positions i such that input bit s_i^{L} affects the output of f_ζ.
- Let $\mathcal{S}_{\mathsf{R} \to \zeta}$ denote the subset of $\mathcal{S}_{\mathsf{R} \to \mathsf{L}}$ that affects f_ζ.

We next define the following event Good_f.

Definition 14. *The event* Good_f *occurs if for tampering function* $f \in \mathsf{Local}^{\ell_o}$ *all of the following hold:*

1. ρ *contains at least* n_L *ones.*
2. $|\mathcal{S}_{\mathsf{check}} \cup S_{in}(1/\delta \cdot \ell_o) \cup \mathcal{S}_{\mathsf{L} \to \zeta}| \leq c^{\mathsf{sec}} \cdot n_L$.

Let $(\mathsf{E}_L, \mathsf{D}_L, \mathsf{Rec}_L)$ be a binary reconstructable probabilistic encoding scheme with parameters $(k, n_L, c_L^{\mathsf{err}}, c_L^{\mathsf{sec}})$ and let $(\mathsf{E}_R, \mathsf{D}_R, \mathsf{Rec}_R)$ be a binary reconstructable probabilistic encoding scheme with parameters $(k, n_R, c_R^{\mathsf{err}}, c_R^{\mathsf{sec}})$.

Also, let n_{check} be a parameter.

Let prg be a pseudorandom generator for space bounded computations (see Definition 12), with inputs of length r and outputs of length $\log(\tau) \cdot \tau$.

Let $\mathrm{G}(\zeta)$ be defined as follows:

1. Compute $y := \mathsf{prg}(\zeta)$.
2. Divide pseudorandom tape y into blocks of bit strings $y_1, \ldots, y_\tau$. Let ϕ be the randomized function that chooses a bit $b \in \{0, 1\}$ with bias $p := 3n_L/2\tau$. For $i \in [\tau]$, let $\rho_i = \phi(y_i)$, where y_i is the explicit randomness of ϕ. Let $\rho = \rho_1, \ldots, \rho_\tau$. Let num denote the number of positions of ρ that are set to 1.
3. If $\mathsf{num} < n_L$, set $\rho := 1_L^n 0^{\tau - n_L}$.
4. Otherwise, flip all but the first n_L 1's in ρ to 0.
5. Output ρ.

$\mathsf{E}(x := (\mathsf{L}, \mathsf{R}))$:

1. Compute $\boldsymbol{s}^{\mathsf{L}} = (s_1^{\mathsf{L}}, \ldots, s_{n_L}^{\mathsf{L}}) \leftarrow \mathsf{E}_L(\mathsf{L})$ and $\boldsymbol{s}^{\mathsf{R}} = (s_1^{\mathsf{R}}, \ldots, s_{n_R}^{\mathsf{R}}) \leftarrow \mathsf{E}_R(\mathsf{R})$.
2. Choose $\zeta \leftarrow \{0, 1\}^r$ uniformly at random. Compute $\rho := \mathrm{G}(\zeta)$.
3. Otherwise, for $j \in [n_L]$, let pos_j denote the j-th position i such that $\rho_i = 1$.
4. Let $X^{\mathsf{L}} \in \{0, 1\}^\tau$ be defined in the following way: For $j \in [n_L]$, $X_{\mathsf{pos}_j}^{\mathsf{L}} := s_j^{\mathsf{L}}$. In all other locations, X_i^{L} is set uniformly at random.
5. Output the encoding $(\zeta, X^{\mathsf{L}}, \boldsymbol{s}^{\mathsf{R}})$.

$\mathsf{D}(\boldsymbol{\sigma} := (\widetilde{\zeta}, \widetilde{X}^{\mathsf{L}}, \boldsymbol{\sigma}^{\mathsf{R}}))$:

1. (`Recover` $\widetilde{\rho}$) Let $\widetilde{y} := \mathsf{prg}(\widetilde{\zeta})$, where $\widetilde{y} = \widetilde{y}_1, \ldots, \widetilde{y}_\tau$. For $i \in [\tau]$, compute $\widetilde{\rho}_i := \phi(\widetilde{y}_i)$. Let $\widetilde{\mathsf{num}}$ denote the number of ones in $\widetilde{\rho} := \widetilde{\rho}_1, \ldots, \widetilde{\rho}_\tau$.
2. (`Recover` x) For $j \in [\widetilde{\mathsf{num}}]$, let pos_j denote the j-th position i such that $\widetilde{\rho}_i = 1$.
3. Let $\boldsymbol{\sigma}^{\mathsf{L}} \in \{0, 1\}^{n_L}$ be defined in the following way: For $j \in [\min(\widetilde{\mathsf{num}}, n_L)]$, $\sigma_j^{\mathsf{L}} := \widetilde{X}_{\mathsf{pos}_j}^{\mathsf{L}}$.
4. (`plain decoding on left`) Compute $((w_1^{\mathsf{L}}, \ldots, w_{n_L}^{\mathsf{L}}), \mathsf{L}) \leftarrow \mathsf{D}_L(\sigma_1^{\mathsf{L}}, \ldots, \sigma_{n_L}^{\mathsf{L}})$. If the decoding fails, set $\mathsf{L} := \perp$.
5. (`decoding-check on right`) Let $t := \lceil n_R(1 - c_R^{\mathsf{err}}/4) \rceil$ Define $\boldsymbol{\sigma}'^{\mathsf{R}} := \sigma_1'^{\mathsf{R}}, \ldots, \sigma_{n_R}'^{\mathsf{R}}$ as follows: Set $\sigma_\ell'^{\mathsf{R}} := \sigma_\ell^{\mathsf{R}}$ for $\ell = 1, \ldots, t$. Set $\sigma_\ell'^{\mathsf{R}} := 0$ for $\ell = t + 1, \ldots, n_R$. Compute $((w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}}), \mathsf{R}) \leftarrow \mathsf{D}_R(\sigma_1'^{\mathsf{R}}, \ldots, \sigma_t'^{\mathsf{R}})$. If the decoding fails or $(w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}})$ is not $c_R^{\mathsf{err}}/4$-close to $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{t_R}^{\mathsf{R}})$, set $\mathsf{R} := \perp$.
6. (`codeword-check on right`) Pick a random subset $R_{\mathsf{check}} \subset [n_R]$ of size $n_{\mathsf{check}} < c_R^{\mathsf{sec}} \cdot n_R$. For all $\ell \in R_{\mathsf{check}}$, check that $\sigma_\ell^{\mathsf{R}} = w_\ell^{\mathsf{R}}$. If the check fails, set $\mathsf{R} := \perp$.
7. (`output`) Output $x := (\mathsf{L}, \mathsf{R})$.

Fig. 4. THE $(\mathrm{Local}^{\ell_o}, \mathsf{SS}, \mathsf{negl}(k))$-NON-MALLEABLE REDUCTION (E, D)

3. *There is some set $J^* \subset [n_R]$ such that $|J^*| = t$ and $|\mathcal{S}_{L \to R}^{J^*} \setminus S_{in}(1/\delta \cdot \ell_o)| = 0$ (from now on, J^* will denote the lexicographically first such set).*

4. $|\mathcal{S}_{R \to L}| \leq \ell_o \cdot (r + n_L) \leq n_R \cdot c_R^{\text{sec}}$.

Claim. Suppose ρ is chosen truly at random (ones occuring with bias $p = 3n_L/2\tau$). Then for every $f \in$ Local^{ℓ_o}, $\Pr[\text{Good}_f] \geq 1 - \text{negl}(n)$.

Proof. We consider each part of the event Good_f separately.

1. Recall that $|\rho| = \tau$, $n_L = c^{\text{rate}}k$, $n = O(\ell_o k)$ and $p = \frac{3n_L}{2\tau}$.
   Let the number of ones in $\rho = \rho_1$. Note that ρ_1 is a Binomial random variable with parameters (τ, p). Therefore, let $\mu = \mathbf{E}[\rho_1] = \tau p = \tau \frac{3n_L}{2\tau} = \frac{3n_L}{2}$ Using Chernoff's bound we can write,
   $\Pr[\rho_1 \leq 2/3\mu] \leq e^{\frac{-1/3^2}{2}\mu}$. Therefore,

   $$\Pr[\rho_1 \leq n_L] \leq e^{-\frac{1}{9}\frac{\mu}{2}}$$
   $$= e^{-\frac{1}{9}\frac{3n_L}{4}}$$
   $$= e^{-\frac{n_L}{12}},$$

   which is negligible.

2. We know that, $|\mathcal{S}_{\text{check}}^+| \leq \ell_o n_{\text{check}}$, $|S_{in}^+(1/\delta \cdot \ell_o)| \leq \delta n \leq 2\delta\tau$ and $|\mathcal{S}_{L \to \zeta}^+| \leq \ell_o r$. Therefore, we can upper bound $|\mathcal{S}_{\text{check}}^+ \cup S_{in}^+(1/\delta \cdot \ell_o) \cup \mathcal{S}_{L \to \zeta}^+| \leq 3\delta\tau$.

   We would now like to show that with high probability, $|\mathcal{S}_{\text{check}} \cup S_{in}(1/\delta \cdot \ell_o) \cup \mathcal{S}_{L \to \zeta}| \leq c^{\text{sec}}n_L$. We consider a mental experiment, in which the adversary fixes the tampering function f and only after f is fixed, R_{check} and ρ are chosen. In this case, each position in $\mathcal{S}_{\text{check}}^+ \cup S_{in}^+(1/\delta \cdot \ell_o) \cup \mathcal{S}_{L \to \zeta}^+$ corresponds to an $\mathbf{s}_i^L$ input variable (selected by ρ) with independent probability p. We therefore observe that the size of $|\mathcal{S}_{\text{check}} \cup S_{in}(1/\delta \cdot \ell_o) \cup \mathcal{S}_{L \to \zeta}|$ can be upper bounded by a Binomial random variable V with parameters $(3\delta\tau, p)$. Let $\mu = \mathbf{E}[\mathsf{V}] = 3\delta\tau p = 3\tau\delta\frac{3n_L}{2\tau} = \frac{9}{2}n_L \cdot \delta$ Using Chernoff's bound we can write,
   $\Pr[\mathsf{V} \geq 2\mu] \leq e^{-\frac{3 \cdot n_L \cdot \delta}{2}}$. Therefore, since $\delta = \frac{c^{\text{sec}}}{9}$, we have that $\Pr[|\mathcal{S}_{\text{check}} \cup S_{in}(1/\delta \cdot \ell_o) \cup \mathcal{S}_{L \to \zeta}| \geq 9n_L \cdot \delta = c^{\text{sec}} \cdot n_L] \leq e^{-\frac{c^{\text{sec}} \cdot n_L}{6}}$, which is negligible.

3. Clearly, $|\mathcal{S}_{L \to R} \setminus S_{in}(1/\delta \cdot \ell_o)| \leq n_L \cdot 1/\delta \cdot \ell_o$. Thus, in order to prove this part of the claim, it is sufficient to show that $n_R - n_L \cdot 1/\delta \cdot \ell_o \geq t$, where $t = \lceil n_R(1 - c_R^{\text{err}}/4) \rceil$. Plugging in our choice of parameters, we get

   $$n_R - n_L \cdot 1/\delta \cdot \ell_o \geq \lceil \frac{2\ell_o c^{\text{rate}}k}{\delta c^{\text{dec}}c^{\text{sec}}} \rceil - \lceil \frac{c^{\text{rate}}k \cdot \ell_o}{\delta} \rceil$$
   $$\geq \lceil n_R(1 - \frac{c^{\text{dec}}c^{\text{sec}}}{2}) \rceil$$

   Since $\frac{c^{\text{dec}}c^{\text{sec}}}{2} \leq c_R^{\text{err}}/4$, the inequality holds.

4. We need to show that $|\mathcal{S}_{R \to L}| \leq \ell_o \cdot (r + n_L) \leq n_R \cdot c_R^{\text{sec}}$.
   The first inequality $|\mathcal{S}_{R \to L}| \leq \ell_o \cdot (r + n_L)$, holds from the definition of output locality. Recall that $c_R^{\text{sec}} \cdot n_R = c_R^{\text{sec}} \cdot \lceil \frac{2\ell_o c^{\text{rate}}k}{\delta c^{\text{dec}}c^{\text{sec}}} \rceil \geq 2\ell_o c^{\text{rate}}k$. Thus, by substituting parameters we get $\ell_o \cdot (r + n_L) = \ell_o \cdot (k + c^{\text{rate}}k) \leq 2\ell_o c^{\text{rate}}k$, since $c^{\text{rate}} > 1$ and so the second inequality holds as well.

Now, for every $f \in \text{Local}^{\ell_o}$, we define the distribution G_f over SS_k. A draw, $g = (g_L, g_R)$, from G_f is defined as follows:

- Choose $\zeta \leftarrow \{0, 1\}^r$ uniformly at random. Compute $\rho := G(\zeta)$.

- If Good_f does not hold, then set g to the constant function 0.
- Otherwise, choose vectors $I^{\mathsf{R}} \in \{0,1\}^{n_R}, I^{\mathsf{L}} \in \{0,1\}^{n_L}$ uniformly at random. $I^{\mathrm{mask}} \in \{0,1,*\}^\tau$ such that if $i \le \mathsf{pos}$ (the location of the n_L-th 1 in ρ) and $\rho_i = 1$, then $I_i^{\mathrm{mask}} = *$ and is uniformly independently drawn from $\{0,1\}$ otherwise.
  Let I^X be I^{mask} where the i-th $*$ is replaced by the ith bit of I^{L}. ($\mathrm{id}|_{I^{\mathrm{mask}}}(I^{\mathsf{L}})$.)
- Compute $\widetilde{\zeta}$ according to f_ζ, given inputs ζ and $I^{\mathsf{L}}_{\mathcal{S}_{\mathsf{L} \to \zeta}}, I^{\mathsf{R}}_{\mathcal{S}_{\mathsf{R} \to \zeta}}$.
- Compute $\widetilde{\rho} := \mathrm{G}(\widetilde{\zeta})$, where $\widetilde{\rho} = \widetilde{\rho}_1, \ldots, \widetilde{\rho}_\tau$. Let $\widetilde{U} = \{i \in [\tau] : \widetilde{\rho}_i = 1\}$.
- Choose a random subset $R_{\mathsf{check}} \subseteq [n_R]$ of size n_{check}.
- Let J^* be the the lexicographically first subset of $[n_R]$ such that $|J^*| = t$ and $|\mathcal{S}^{J^*}_{\mathsf{L} \to \mathsf{R}} \setminus S_{\mathrm{in}}(\ell_o/\delta)| = 0$. Note that such J^* must exist since Good_f occurs.
- The split-state tampering function $g := (g_L, g_R) \in \mathrm{SS}_k$ has $I^{\mathsf{L}}, I^{\mathsf{R}}, I^{\mathrm{mask}}, \zeta$ hardcoded into it (as well as $\widetilde{U}, \widetilde{\zeta}, \widetilde{\rho}$) and is specified as follows:
  $g_L(\mathsf{L})$:
  1. (`apply tampering and plain decode on left`) Let $\boldsymbol{s}^{\mathsf{L}} := \mathsf{Rec}(\mathcal{S}_{\mathsf{check}} \cup \mathcal{S}_{\mathsf{L} \to \zeta} \cup S_{\mathrm{in}}(\ell_o/\delta), I^{\mathsf{L}}, \mathsf{L})$. Let $(\sigma_1^{\mathsf{L}}, \ldots, \sigma_{n_L}^{\mathsf{L}}) := (f_X|_{\zeta, I^{\mathrm{mask}}, I^{\mathsf{R}}}(\boldsymbol{s}^{\mathsf{L}}))_{\widetilde{U}}$. Recall that the above notation denotes (1) applying the function f_X to the input $(\zeta, X^{\mathsf{L}}, I^{\mathsf{R}})$, where X^{L} is I^{mask} where the i-th $*$ is replaced by the ith bit of $\boldsymbol{s}^{\mathsf{L}}$. (2) outputting the positions of f_X indexed by $\widetilde{U}$. Compute $((w_1^{\mathsf{L}}, \ldots, w_{n_L}^{\mathsf{L}}), \widetilde{\mathsf{L}}) \leftarrow \mathsf{D}_L(\sigma_1^{\mathsf{L}}, \ldots, \sigma_{n_L}^{\mathsf{L}})$. If the decoding fails, set $\widetilde{\mathsf{L}} := \perp$.
  2. (`output`) Output $\widetilde{\mathsf{L}}$.
  $g_R(\mathsf{R})$:
  1. (`apply tampering and decoding-check on right`) Let $\boldsymbol{s}^{\mathsf{R}} = (s_1^{\mathsf{R}}, \ldots, s_{n_R}^{\mathsf{R}}) := \mathsf{Rec}(\mathcal{S}_{\mathsf{R} \to \mathsf{L}}, I^{\mathsf{R}}, \mathsf{R})$. Let $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{n_R}^{\mathsf{R}}) := f^{\mathsf{R}}|_{\zeta, I^X}(\boldsymbol{s}^{\mathsf{R}})$. Define $\boldsymbol{\sigma}'^{\mathsf{R}} := \sigma_1'^{\mathsf{R}}, \ldots, \sigma_{n_R}'^{\mathsf{R}}$ as follows: Set $\sigma_\ell'^{\mathsf{R}} := \sigma_\ell^{\mathsf{R}}$ for $\ell \in [J^*]$. Set $\sigma_\ell'^{\mathsf{R}} := 0$ for $\ell \notin [J^*]$. Compute $((w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}}), \widetilde{\mathsf{R}}) \leftarrow \mathsf{D}_R(\sigma_1'^{\mathsf{R}}, \ldots, \sigma_t'^{\mathsf{R}})$. If the decoding fails or $(w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}})$ is not $c_R^{\mathrm{err}}/4$-close to $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{n_R}^{\mathsf{R}})$, then set $\widetilde{\mathsf{R}} := \perp$.
  2. (`codeword-check on right`) For all $\ell \in R_{\mathsf{check}}$, check that $\sigma_\ell^{\mathsf{R}} = w_\ell^{\mathsf{R}}$. If the check fails, set $\widetilde{\mathsf{R}} := \perp$.
  3. (`output`) Output $\widetilde{\mathsf{R}}$.
- Output $g = (g_L, g_R)$.

Whenever Rec is run above, we assume that enough positions are set by $\mathcal{S}$ such that there is only a single consistent codeword. If this is not the case, then additional positions are added to $\mathcal{S}$ from $I^{\mathsf{L}}, I^{\mathsf{R}}$, respectively.

Remark 5. Note that $g = (g_L, g_R)$ drawn from the distribution G_f is a split-state function with probability 1. This is true since if the event Good_f does not hold, then g is set to the constant function 0 (which is in split-state). If the event Good_f does hold, then as can be seen by inspection above, g_L takes only L as input, g_R takes only R as input.

By the definition of a non-malleable reduction (Definition 3), in order to complete the proof of Theorem 4, we must show that (E, D) have the following properties:

1. For all $x \in \{0,1\}^{2k}$, we have $\mathsf{D}(\mathsf{E}(x)) = x$ with probability 1.
2. For all $f \in \mathrm{Local}^{\ell_o}$,
$$\Delta(\mathsf{D}(f(\mathsf{E}(x))); G_f(x)) \le \mathsf{negl}(k),$$
where G_f is the distribution defined above.

Item (1) above is trivial and can be immediately verified.

In the following, we prove Item (2) above by considering the following sequence of hybrid arguments for each function $f \in \mathrm{Local}^{\ell_o}$ (for the intermediate hybrids, we highlight the step in which they are different from the desired end distributions).

Our reduction will move in three phases to essentially disconnect influence between the 3 blocks of our encoding (that is dependent on the underlying message, x.

Firstly, we will argue that with high probability, the pseudorandomness of the PRG is sufficient to obtain that the event Good_f holds even when ρ is chosen via the PRG (instead of being truly random). This will give us bounds on the "bad" bits in the output of the encoding of the left input, L.

Next, we will use two hybrids to show that we can safely sample bits in $\mathcal{S}_{\mathsf{R}\to\zeta}$, $\mathcal{S}_{\mathsf{R}\to\mathsf{L}}$, $\mathcal{S}_{\mathsf{L}\to\zeta}$, $\mathcal{S}_{\mathsf{check}}$, and S_{in} uniformly at random. Given our first claim, the sizes of all of these sets together will be below the secrecy threshold of the respective Reconstructible Probablitistic Encodings. As such, the distribution over the randomness of the encoding procedure will be identical, for any message. Notice that at this stage we can simulate the entire left hand side, as well as codeword check on the right. All that we need to handle is influence from $\boldsymbol{s}^{\mathsf{L}}$ on the rest of $\boldsymbol{\sigma}^{\mathsf{R}}$.

So for our final hybrid, we will use a technique from [CDMW08] to show that we can decode from a "clean" portion of $\boldsymbol{\sigma}^{\mathsf{R}}$, J^*. This is possible because, by Claim 5 (and previous hybrids), the only non-uniformly-random bits in $\boldsymbol{s}^{\mathsf{L}}$ have bounded input locality. Thus, these bits on the left can only effect a constant fraction of bits on the right.

To begin, we prove the following claim to bound the set of "bad" bits on the left side.

Claim. In any given tampering experiment, with tampering function $f \in\in \mathrm{Local}^{\ell_o}$, the event Good_f holds with all but negligible probability, even when ρ generated according to Figure 4.

Proof. Suppose not, then there exists some f that violates the above constraints, or any f will because the PRG will not select enough ones. Notice that f determines $S_{\mathsf{in}}(\ell_o/\delta)$. ($\mathcal{S}_{\mathsf{L}\to\zeta}, \mathcal{S}_{\mathsf{check}}$ are determined by f and the randomness of E.)

Recall by choice of parameters when ρ is chosen at random, there can be at most $\ell_o n_R + \delta n + \ell_o r \le c^{\mathsf{sec}}\tau/3$ number of positions in X^{L} in $\mathcal{S}_{\mathsf{check}} \cup S_{\mathsf{in}}(\ell_o/\delta) \cup \mathcal{S}_{\mathsf{L}\to\zeta}$. Since $p = 3n_L/2\tau$, we *expect* $c_L^{\mathsf{sec}} \cdot n_L/2$ number of positions i in X^{L} where (1) $X_i^{\mathsf{L}} \in \mathcal{S}_{\mathsf{check}} \cup S_{\mathsf{in}}(\ell_o/\delta) \cup \mathcal{S}_{\mathsf{L}\to\zeta}$ (is "bad") and (2) $\rho_i = 1$. Therefore, by multiplicative Chernoff bounds, the probability that more than $c_L^{\mathsf{sec}} \cdot n_L$ number of positions i in X^{L} are such that (1) X_i^{L} is "bad" and (2) $\rho_i = 1$ is negligible.

We now argue that these events must also occur with negligible probability when ρ is pseudorandom. Assume the contrary, then the following is a distinguisher T that can distinguish truly random strings y from strings $\rho := \mathrm{G}(\zeta)$ with non-negligible probability.

T is a circuit that has a string $w \in \{0,1\}^\tau$ hardwired into it. w corresponds to the characteristic vector of $S_{\mathsf{in}}(\ell_o/\delta) \cup \mathcal{S}_{\mathsf{L}\to\zeta} \cup \mathcal{S}_{\mathsf{check}}$.

On input $y = y_1, \ldots, y_\tau$ (where either $y := \mathsf{prg}(\zeta)$ or y is chosen uniformly at random), $T(y)$ does the following:

1. Set $count_1 = 0$, $count_2 = 0$.
2. For $i = 1$ to τ:
   (a) Run $\phi(y_i)$ to obtain ρ_i.
   (b) If $\rho_i = 1$, set $count_2 := count_2 + 1$
   (c) If $\rho_i = 1$ and $w_i = 1$, set $count_1 := count_1 + 1$.
3. If $count_1 > c_L^{\mathsf{sec}} \cdot n_L/3$ or $count_2 < n_L$, output 0. Otherwise, output 1.

T can clearly be implemented by a read-once, Finite State Machine (FSM) with $2^{O(\log^2(\tau))}$ number of states. However, note that by Theorem 3, prg is a pseudorandom generator for space $\log^3(k)$ with parameter 2^{-k}. Thus, existence of distinguisher T as above, leads to contradiction to the security of the Nisan PRG.

Hybrid H_0. This is the original distribution $\mathsf{D}(f(\mathsf{E}(x)))$

Hybrid H_1. H_1 corresponds to the distribution $\mathsf{D}(f(\mathsf{E}'(x)))$, where E' is defined as follows:
$\mathsf{E}'(x := (\mathsf{L}, \mathsf{R}))$:

1. Given f, find $\mathcal{S}_{\mathsf{R}\to\zeta}$, $\mathcal{S}_{\mathsf{R}\to\mathsf{L}}$, $\mathcal{S}_{\mathsf{L}\to\zeta}$, $\mathcal{S}_{\mathsf{check}}$, and $S_{\mathsf{in}}(\ell_o/\delta)$.

2. Compute $s^{\mathsf{L}} = (s^{\mathsf{L}}_1, \ldots, s^{\mathsf{L}}_{n_L}) \leftarrow \mathsf{Rec}(\mathcal{S}_{\mathsf{check}} \cup \mathcal{S}_{\mathsf{L} \to \zeta} \cup S_{\mathrm{in}}(\ell_o/\delta), \mathsf{E}_L(\mathsf{L}), \mathsf{L})$
   and $s^{\mathsf{R}} = (s^{\mathsf{R}}_1, \ldots, s^{\mathsf{R}}_{n_R}) \leftarrow \mathsf{Rec}(\mathcal{S}_{\mathsf{R} \to \mathsf{L}}, \mathsf{E}_R(\mathsf{R}), \mathsf{R})$.

3. Choose $\zeta \leftarrow \{0,1\}^r$ uniformly at random. Compute $\rho := \mathrm{G}(\zeta)$.
4. Otherwise, for $j \in [n_L]$, let pos_j denote the j-th position i such that $\rho_i = 1$.
5. Let $X^{\mathsf{L}} \in \{0,1\}^\tau$ be defined in the following way: For $j \in [n_L]$, $X^{\mathsf{L}}_{\mathsf{pos}_j} := s^{\mathsf{L}}_j$. In all other locations, X^{L}_i is set uniformly at random.
6. Output the encoding $(\zeta, X^{\mathsf{L}}, s^{\mathsf{R}})$.

Note that the only difference between E and E' is that we are reconstructing codewords before outputting. As the original codewords are already complete, the output is identical.

Claim.

$$H_0 \equiv H_1.$$

The claim follows trivially from the definition of the reconstruction procedure.

Hybrid H_2 H_2 corresponds to the distribution $\mathsf{D}(f(\mathsf{E}^{(2)}(x)))$, where $\mathsf{E}^{(2)}$ is defined as follows:
$\mathsf{E}^{(2)}(x := (\mathsf{L}, \mathsf{R}))$:

1. Given f, find $\mathcal{S}_{\mathsf{R} \to \zeta}$, $\mathcal{S}_{\mathsf{R} \to \mathsf{L}}$, $\mathcal{S}_{\mathsf{L} \to \zeta}$, $\mathcal{S}_{\mathsf{check}}$, and $S_{\mathrm{in}}(\ell_o/\delta)$.

2. Choose $I^{\mathsf{L}} \in \{0,1\}^{n_L}, I^{\mathsf{R}} \in \{0,1\}^{n_R}$ uniformly at random.
3. Compute $s^{\mathsf{L}} = (s^{\mathsf{L}}_1, \ldots, s^{\mathsf{L}}_{n_L}) \leftarrow \mathsf{Rec}(\mathcal{S}_{\mathsf{check}} \cup \mathcal{S}_{\mathsf{L} \to \zeta} \cup S_{\mathrm{in}}(\ell_o/\delta), I^{\mathsf{L}}, \mathsf{L})$
   and $s^{\mathsf{R}} = (s^{\mathsf{R}}_1, \ldots, s^{\mathsf{R}}_{n_R}) \leftarrow \mathsf{Rec}(\mathcal{S}_{\mathsf{R} \to \mathsf{L}}, I^{\mathsf{R}}, \mathsf{R})$.

4. Choose $\zeta \leftarrow \{0,1\}^r$ uniformly at random. Compute $\rho := \mathrm{G}(\zeta)$.
5. For $j \in [n_L]$, let pos_j denote the j-th position i such that $\rho_i = 1$.
6. Let $X^{\mathsf{L}} \in \{0,1\}^\tau$ be defined in the following way: For $j \in [n_L]$, $X^{\mathsf{L}}_{\mathsf{pos}_j} := s^{\mathsf{L}}_j$. In all other locations, X^{L}_i is set uniformly at random.
7. Output the encoding $(\zeta, X^{\mathsf{L}}, s^{\mathsf{R}})$.

Claim.

$$H_1 \equiv H_2.$$

By Claim 5, the sets reconstructed from are below the security thresholds of the respective RPE schemes. Thus by definition, the distribution of $\mathsf{E}^{(2)}(x)$ is identical to that of $\mathsf{E}(x)$.

Hybrid H_3. H_3 corresponds to the distribution $\mathsf{D}'(f(\mathsf{E}^{(2)}(x)))$, where D' is defined as follows:
$\mathsf{D}(\boldsymbol{\sigma} := (\widetilde{\zeta}, \widetilde{X}^{\mathsf{L}}, \boldsymbol{\sigma}^{\mathsf{R}}))$:

1. (`Recover` $\widetilde{\rho}$) Let $\widetilde{y} := \mathsf{prg}(\widetilde{\zeta})$, where $\widetilde{y} = \widetilde{y}_1, \ldots, \widetilde{y}_\tau$. For $i \in [\tau]$, compute $\widetilde{\rho}_i := \phi(\widetilde{y}_i)$. Let $\widetilde{\mathsf{num}}$ denote the number of ones in $\widetilde{\rho} := \widetilde{\rho}_1, \ldots, \widetilde{\rho}_\tau$.
2. (`Recover` x) For $j \in [\widetilde{\mathsf{num}}]$, let pos_j denote the j-th position i such that $\widetilde{\rho}_i = 1$.
3. Let $\boldsymbol{\sigma}^{\mathsf{L}} \in \{0,1\}^{n_L}$ be defined in the following way: For $j \in [\min(\widetilde{\mathsf{num}}, n_L)]$, $\sigma^{\mathsf{L}}_j := \widetilde{X}^{\mathsf{L}}_{\mathsf{pos}_j}$.
4. (`plain decoding on left`) Compute $((w^{\mathsf{L}}_1, \ldots, w^{\mathsf{L}}_{n_L}), \mathsf{L}) \leftarrow \mathsf{D}_L(\sigma^{\mathsf{L}}_1, \ldots, \sigma^{\mathsf{L}}_{n_L})$. If the decoding fails, set $\mathsf{L} := \bot$.
5. (`decoding-check on right`) Let $t := \lceil n_R(1 - c^{\mathrm{err}}_R/4) \rceil$ Define $\boldsymbol{\sigma}'^{\mathsf{R}} := \sigma'^{\mathsf{R}}_1, \ldots, \sigma'^{\mathsf{R}}_{n_R}$ as follows: Set $\sigma'^{\mathsf{R}}_\ell := \sigma^{\mathsf{R}}_\ell$ for $\ell = 1, \ldots, t$. Set $\sigma'^{\mathsf{R}}_\ell := 0$ for $\ell = t+1, \ldots, n_R$. Compute $((w^{\mathsf{R}}_1, \ldots, w^{\mathsf{R}}_{n_R}), \mathsf{R}) \leftarrow \mathsf{D}_R(\sigma'^{\mathsf{R}}_1, \ldots, \sigma'^{\mathsf{R}}_t)$. If the decoding fails or $(w^{\mathsf{R}}_1, \ldots, w^{\mathsf{R}}_{n_R})$ is not $c^{\mathrm{err}}_R/4$-close to $(\sigma^{\mathsf{R}}_1, \ldots, \sigma^{\mathsf{R}}_{t_R})$, set $\mathsf{R} := \bot$.
6. (`codeword-check on right`) Pick a random subset $R_{\mathsf{check}} \subset [n_R]$ of size $n_{\mathsf{check}} < c^{\mathrm{sec}}_R \cdot n_R$. For all $\ell \in R_{\mathsf{check}}$, check that $\sigma^{\mathsf{R}}_\ell = w^{\mathsf{R}}_\ell$. If the check fails, set $\mathsf{R} := \bot$.
7. (`output`) Output $x := (\mathsf{L}, \mathsf{R})$.

8. (`decoding-check on right`) Define $\boldsymbol{\sigma}'^{\mathsf{R}} := \sigma_1'^{\mathsf{R}}, \ldots, \sigma_{n_R}'^{\mathsf{R}}$ as follows: Set $\sigma_\ell'^{\mathsf{R}} := \sigma_\ell^{\mathsf{R}}$ for $\ell \in J^*$ and $\sigma_\ell'^{\mathsf{R}} := 0$ for $\ell \notin J^*$, where $J^* \subseteq [n_R]$ is the lexicographically first set such that $|J^*| = t$ and $|\mathcal{S}_{\mathsf{L} \to \mathsf{R}}^{J^*} \cap S_{\mathrm{in}}(\bar{\ell}_o / \delta)| = 0$. Compute $((w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}}), \mathsf{R}) \leftarrow \mathsf{D}_R(\sigma_1'^{\mathsf{R}}, \ldots, \sigma_{t_R}'^{\mathsf{R}})$. If the decoding fails or $(w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}})$ is not $c_R^{\mathrm{err}}/4$-close to $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{t_R}^{\mathsf{R}})$, set $\mathsf{R} := \bot$.

9. (`codeword-check on right`) For all $\ell \in R_{\mathsf{check}}$, check that $\sigma_\ell^{\mathsf{R}} = w_\ell^{\mathsf{R}}$. If the check fails, set $\mathsf{R} := \bot$.
10. (`output`) Output $x := (\mathsf{L}, \mathsf{R})$.

Note that the only difference between D and D' is that in `decoding-check on right`, $\boldsymbol{\sigma}^{\mathsf{R}}$ is decoded from J^*, instead of the first n_{check} positions.

Claim.
$$H_2 \overset{s}{\approx} H_3.$$

We want to show that for every $\boldsymbol{\sigma} = (\boldsymbol{\sigma}^{\mathsf{L}}, \boldsymbol{\sigma}^{\mathsf{R}}) \leftarrow f(\mathsf{E}(x))$, $\mathsf{D}(\boldsymbol{\sigma}) = \mathsf{D}'(\boldsymbol{\sigma})$ with high probability, over the coins of D, D'. The proof is identical to the proof of Claim 3.

Hybrid H_4. Hybrid H_4 is simply the distribution $G_f(x)$, defined previously.

Claim.
$$H_3 \equiv H_4.$$

This claim follows by inspection.

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 Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 393–417, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

ADKO14. Divesh Aggarwal, Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Leakage-resilient non-malleable codes. Cryptology ePrint Archive, Report 2014/807, 2014. http://eprint.iacr.org/2014/807.

ADKO15. Divesh Aggarwal, Yevgeniy Dodis, Tomasz Kazana, and Maciej Obremski. Non-malleable reductions and applications. In Rocco A. Servedio and Ronitt Rubinfeld, editors, *47th ACM STOC*, pages 459–468, Portland, OR, USA, June 14–17, 2015. ACM Press.

ADL14.   Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In David B. Shmoys, editor, *46th ACM STOC*, pages 774–783, New York, NY, USA, May 31 – June 3, 2014. ACM Press.

AGM+15a.   Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Explicit non-malleable codes against bit-wise tampering and permutations. In Rosario Gennaro and Matthew Robshaw, editors, *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, volume 9215 of *Lecture Notes in Computer Science*, pages 538–557. Springer, 2015.

AGM+15b.   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 Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 375–397, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

App14.   Benny Applebaum. *Cryptography in Constant Parallel Time*. Information Security and Cryptography. Springer, 2014.

BOGW88.   Michael Ben-Or, Shafi Goldwasser, and Avi Wigderson. Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In *20th ACM STOC*, pages 1–10, Chicago, Illinois, USA, May 2–4, 1988. ACM Press.

CCFP11.   Hervé Chabanne, Gérard D. Cohen, Jean-Pierre Flori, and Alain Patey. Non-malleable codes from the wire-tap channel. *CoRR*, abs/1105.3879, 2011.

CCP12.   Hervé Chabanne, Gérard D. Cohen, and Alain Patey. Secure network coding and non-malleable codes: Protection against linear tampering. In *Proceedings of the 2012 IEEE International Symposium on Information Theory, ISIT 2012, Cambridge, MA, USA, July 1-6, 2012*, pages 2546–2550. IEEE, 2012.

CDMW08.   Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. Black-box construction of a non-malleable encryption scheme from any semantically secure one. In Ran Canetti, editor, *TCC 2008*, volume 4948 of *LNCS*, pages 427–444, San Francisco, CA, USA, March 19–21, 2008. Springer, Heidelberg, Germany.

CDMW15.   Seung Geol Choi, Dana Dachman-Soled, Tal Malkin, and Hoeteck Wee. A note on improved, black-box constructions of non-malleable encryption from semantically-secure encryption. Manuscript, 2015.

CDTV15.   Sandro Coretti, Yevgeniy Dodis, Björn Tackmann, and Daniele Venturi. Non-malleable encryption: Simpler, shorter, stronger. Cryptology ePrint Archive, Report 2015/772, 2015. `http://eprint.iacr.org/2015/772`.

CDTV16.   Sandro Coretti, Yevgeniy Dodis, Björn Tackmann, and Daniele Venturi. Non-malleable encryption: Simpler, shorter, stronger. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part I*, volume 9562 of *LNCS*, pages 306–335, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

CG14a.   Mahdi Cheraghchi and Venkatesan Guruswami. Capacity of non-malleable codes. In Moni Naor, editor, *ITCS 2014*, pages 155–168, Princeton, NJ, USA, January 12–14, 2014. ACM.

CG14b.   Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 440–464, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

CKM11.   Seung Geol Choi, Aggelos Kiayias, and Tal Malkin. BiTR: Built-in tamper resilience. In Dong Hoon Lee and Xiaoyun Wang, editors, *ASIACRYPT 2011*, volume 7073 of *LNCS*, pages 740–758, Seoul, South Korea, December 4–8, 2011. Springer, Heidelberg, Germany.

CKR16.   Nishanth Chandran, Bhavana Kanukurthi, and Srinivasan Raghuraman. Information-theoretic local non-malleable codes and their applications. In Eyal Kushilevitz and Tal Malkin, editors, *TCC 2016-A, Part II*, volume 9563 of *LNCS*, pages 367–392, Tel Aviv, Israel, January 10–13, 2016. Springer, Heidelberg, Germany.

CZ14.   Eshan Chattopadhyay and David Zuckerman. Non-malleable codes against constant split-state tampering. In *55th FOCS*, pages 306–315, Philadelphia, PA, USA, October 18–21, 2014. IEEE Computer Society Press.

DDN00.   Danny Dolev, Cynthia Dwork, and Moni Naor. Nonmalleable cryptography. *SIAM Journal on Computing*, 30(2):391–437, 2000.

DGR00.   Scott E Decatur, Oded Goldreich, and Dana Ron. Computational sample complexity. *SIAM Journal on Computing*, 29(3):854–879, 2000.

DKO13.   Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In Ran Canetti and Juan A. Garay, editors, *CRYPTO 2013, Part II*, volume 8043 of *LNCS*, pages 239–257, Santa Barbara, CA, USA, August 18–22, 2013. Springer, Heidelberg, Germany.

DLSZ15. Dana Dachman-Soled, Feng-Hao Liu, Elaine Shi, and Hong-Sheng Zhou. Locally decodable and updatable non-malleable codes and their applications. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *TCC 2015, Part I*, volume 9014 of *LNCS*, pages 427–450, Warsaw, Poland, March 23–25, 2015. Springer, Heidelberg, Germany.

DPW10. Stefan Dziembowski, Krzysztof Pietrzak, and Daniel Wichs. Non-malleable codes. In Andrew Chi-Chih Yao, editor, *ICS 2010*, pages 434–452, Tsinghua University, Beijing, China, January 5–7, 2010. Tsinghua University Press.

FMNV14. Sebastian Faust, Pratyay Mukherjee, Jesper Buus Nielsen, and Daniele Venturi. Continuous non-malleable codes. In Yehuda Lindell, editor, *TCC 2014*, volume 8349 of *LNCS*, pages 465–488, San Diego, CA, USA, February 24–26, 2014. Springer, Heidelberg, Germany.

FMVW14. Sebastian Faust, Pratyay Mukherjee, Daniele Venturi, and Daniel Wichs. Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In Phong Q. Nguyen and Elisabeth Oswald, editors, *EUROCRYPT 2014*, volume 8441 of *LNCS*, pages 111–128, Copenhagen, Denmark, May 11–15, 2014. Springer, Heidelberg, Germany.

GLM⁺04. Rosario Gennaro, Anna Lysyanskaya, Tal Malkin, Silvio Micali, and Tal Rabin. Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In Moni Naor, editor, *TCC 2004*, volume 2951 of *LNCS*, pages 258–277, Cambridge, MA, USA, February 19–21, 2004. Springer, Heidelberg, Germany.

IPSW06. Yuval Ishai, Manoj Prabhakaran, Amit Sahai, and David Wagner. Private circuits II: Keeping secrets in tamperable circuits. In Serge Vaudenay, editor, *EUROCRYPT 2006*, volume 4004 of *LNCS*, pages 308–327, St. Petersburg, Russia, May 28 – June 1, 2006. Springer, Heidelberg, Germany.

ISW03. Yuval Ishai, Amit Sahai, and David Wagner. Private circuits: Securing hardware against probing attacks. In Dan Boneh, editor, *CRYPTO 2003*, volume 2729 of *LNCS*, pages 463–481, Santa Barbara, CA, USA, August 17–21, 2003. Springer, Heidelberg, Germany.

KKS11. Yael Tauman Kalai, Bhavana Kanukurthi, and Amit Sahai. Cryptography with tamperable and leaky memory. In Phillip Rogaway, editor, *CRYPTO 2011*, volume 6841 of *LNCS*, pages 373–390, Santa Barbara, CA, USA, August 14–18, 2011. Springer, Heidelberg, Germany.

LL12. Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *CRYPTO 2012*, volume 7417 of *LNCS*, pages 517–532, Santa Barbara, CA, USA, August 19–23, 2012. Springer, Heidelberg, Germany.

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

A A Scheme with Deterministic Decoding

Let $(\mathsf{E}_L, \mathsf{D}_L, \mathsf{Rec}_L)$ be a RPE with parameters $(k, n_L, c^{\mathsf{err}}, c^{\mathsf{sec}})$, and let $(\mathsf{E}_R, \mathsf{D}_R, \mathsf{Rec}_R)$ be a RPE with parameters $(k, n_R, c^{\mathsf{err}}, c^{\mathsf{sec}})$. Let prg be a PRG for space bounded computations, with inputs of length r and outputs of length $\log(\tau) \cdot \tau$. Consider the construction (E, D), where $\mathsf{E} : \{0,1\}^{2k} \to \{0,1\}^{r+\tau+n_R}$, described in Figure 5, which we will parametrize as follows:

Parametrization.

- $n_L \geq k/c^{\mathsf{sec}}$, so can take $n_L = \Theta(k)$
- $n_R \geq (n_L \cdot \ell_o)/c^{\mathsf{sec}}$ so can take $n_R = \Theta(n_L \cdot \ell_o)$
- $\tau \geq (2n_R \cdot \ell_o)/c^{\mathsf{sec}}$ so can take $\tau = \Theta(n_1(\ell_o)^2)$
- $r \leq (n_R \cdot c^{\mathsf{sec}})/\ell_o$ and $r \leq (n_L \cdot c^{\mathsf{sec}})/(2\ell_o)$ so we can take $r = \Theta(n_L/\ell_o)$ (which satisfies the second, and thus also the first).

Theorem 7. (E, D) *is a* $(\mathrm{Local}^{\ell_o} \to \mathrm{SS}_k, \mathsf{negl}(k))$ *non-malleable reduction given the above parameters (where* $c^{\mathsf{rate}}, c^{\mathsf{err}}, c^{\mathsf{sec}}$ *are taken from Lemma 1).*

Consider a tampering function f (represented as a circuit) in Local^{ℓ_o}. The intuition for security is the following: Note that during decoding, only n_L number of bits (the ones indicated by $\widetilde{\zeta}$) are relevant from within $\widetilde{X}^{\mathsf{L}}$. These n_L output bits can be affected by at most $n_L \cdot \ell_o$ number of input bits on the right. Since

Let prg be a pseudorandom generator for space bounded computations (see Definition 12), with inputs of length r and outputs of length $\log(\tau) \cdot \tau$.

Let $G(\zeta)$ be defined as follows:

1. Compute $y := \mathsf{prg}(\zeta)$.
2. Divide pseudorandom tape y into blocks of bit strings $y_1, \ldots, y_\tau$. Let ϕ be the randomized function that chooses a bit $b \in \{0, 1\}$ with bias $p := 3n_L/2\tau$. For $i \in [\tau]$, let $\rho_i = \phi(y_i)$, where y_i is the explicit randomness of ϕ. Let $\rho = \rho_1, \ldots, \rho_\tau$. Let num denote the number of positions of ρ that are set to 1.
3. If $\mathsf{num} < n_L$, set $\rho := 1_L^n 0^{\tau - n_L}$.
4. Otherwise, flip all but the first n_L 1's in ρ to 0.
5. Output ρ.

$\mathsf{E}(x^\mathsf{L} := x_1^\mathsf{L}, \ldots, x_k^\mathsf{L}, x^\mathsf{R} := x_1^\mathsf{R}, \ldots, x_k^\mathsf{R})$:

1. Let $s^\mathsf{L} := \mathsf{E}_L(x^\mathsf{L})$, $s^\mathsf{R} := \mathsf{E}_R(x^\mathsf{R})$.
2. Choose $\zeta \leftarrow \{0, 1\}^r$ uniformly at random. Compute $\rho := G(\zeta)$.
3. For $j \in [\mathsf{num}]$, let pos_j denote the j-th position i such that $\rho_i = 1$.
4. Let $X^\mathsf{L} \in \{0, 1\}^\tau$ be defined in the following way: For $j \in [n_L]$, $X_{\mathsf{pos}_j}^\mathsf{L} := x_j^\mathsf{L}$. In all other locations, X_i^L is set uniformly at random.
5. Output the encoding $(\zeta, X^\mathsf{L}, s^\mathsf{R})$.

$\mathsf{D}(Z := (\widetilde{\zeta}, \widetilde{X}^\mathsf{L}, \widetilde{s^\mathsf{R}}))$:

1. (Recover $\widetilde{\rho}$) Let $\widetilde{\rho} := G(\widetilde{\zeta})$. Let $\widetilde{\mathsf{num}} \geq n_L$ denote the number of ones in $\widetilde{\rho} := \widetilde{\rho}_1, \ldots, \widetilde{\rho}_\tau$.
2. (Recover x) For $j \in [\widetilde{\mathsf{num}}]$, let pos_j denote the j-th position i such that $\widetilde{\rho}_i = 1$.
3. Let $\widetilde{s^\mathsf{L}}_j \in \{0, 1\}^{n_L}$ be defined in the following way: For $j \in [n_L)]$, $\widetilde{s^\mathsf{L}}_j := \widetilde{X}_{\mathsf{pos}_j}^\mathsf{L}$.
4. (output) Let $\widetilde{x^\mathsf{L}} = \mathsf{D}_L(\widetilde{s^\mathsf{L}})$, $\widetilde{x^\mathsf{R}} = \mathsf{D}_R(\widetilde{s^\mathsf{R}})$. Output $(\widetilde{x^\mathsf{L}}, \widetilde{x^\mathsf{R}})$.

Fig. 5. A non-malleable reduction of output-local to Split State SS_k with deterministic decoding

we choose parameters such that $n_L \cdot \ell_o < c^{\mathsf{sec}} \cdot n_R$, we have—due to the properties of the RPE—that all of the incoming bits from the right, contain absolutely no information about x^{R}. Moreover, there are at most $n_R \cdot \ell_o$ number of bits from the left that affect the decoded output on the right. However, since only a small fraction of the bits on the left are "relevant" and since their positions are "hidden" and determined by the PRG, we expect that only $n_R \cdot \ell_o \cdot (n_L/\tau)$ of these incoming bits are "relevant." Moreover, we can show that with all but negligible probability at most $2 \cdot n_R \cdot \ell_o \cdot (n_L/\tau) = 2 \cdot n_R \cdot \ell_o \cdot (c^{\mathsf{sec}} \cdot n_L/(2n_R\ell_o)) = c^{\mathsf{sec}} \cdot n_L$ number of incoming bits are "relevant." Once again, this means that by the properties of the RPE, all of the incoming bits from the left contain absolutely no information about x^{L}.