# 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, New York, NY, USA
marshall@cs.columbia.edu
[2] University of Maryland, College Park, MD, USA

**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 $n^\delta$ bits, where $n$ is the total number of bits in a codeword and $0 \leq \delta < 1$ a constant. Notably, this tampering class includes $\mathsf{NC}^0$.

## 1 Introduction

Non-malleable codes were first introduced by Dziembowski, Pietrzak and Wichs [24] 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. [24], 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

© International Association for Cryptologic Research 2016
M. Fischlin and J.-S. Coron (Eds.): EUROCRYPT 2016, Part II, LNCS 9666, pp. 881–908, 2016.
DOI: 10.1007/978-3-662-49896-5_31

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. [9,17,24]), where each bit of the codeword can be tampered individually, and its generalization, the split state tampering class (cf. [1–3,13,14,23,31]), 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.

However, beyond some non-explicit (randomized) or inefficient constructions for more general classes (cf. [13,24,26]), 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. [10]—who address certain types of linear tampering functions—and Agrawal et al. [5,6]—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 $\text{Local}^{\ell_o}$, 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 $\mathsf{P}$), 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* $\text{Local}^{\ell_o}$, *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* $\text{poly}(k, \ell_o)$.

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 output length is $n = \Theta(k\ell_o)$ bits, this may result in super-polynomial length encoding. However, using sublinear locality $n^\delta$ yields an efficient code. We highlight this, as well as the special cases of *constant depth* circuits (a subset of $\text{Local}^{O(1)}$), in the following.

**Corollaries:** *There are efficient, explicit, and unconditionally secure non-malleable codes for the following classes:*

- $\text{Local}^{n^\delta}$ *for any constant $0 \leq \delta < 1$, with inverse-polynomial rate.*
- $\mathsf{NC}^0$ *with rate $\Theta(1/\ell_o)$ for any $\ell_o = \omega(1)$.*
- $\mathsf{NC}^0_c$ *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 $\ell_o$ that is super constant (e.g., $\log^*(n)$), and the third by using $\ell_o = 2^c$ (a constant).

While our result for $\mathsf{NC}^0$ 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 $\mathsf{NC}$ circuit, for any constant $0 \leq \delta < 1$. Note that, since separations between $\mathsf{P}$ and $\mathsf{NC}^1$ are not known, constructing (unconditional) non-malleable codes against $\mathsf{NC}^1$ is unlikely, since an attacker in $\mathsf{P}$ 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. [2]. 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. [2] 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 $\text{Local}^{\ell_o}_{\ell_i}$ of local functions, with output locality $\ell_o$ as well as input locality $\ell_i$ (namely each input bit influences at most $\ell_i$ output bits). This class includes bounded-depth circuits with bounded fan-in and

bounded fan-out. Our first intermediate result shows that the class $\text{Local}^{\tilde{O}(\sqrt{n})}_{\tilde{O}(\sqrt{n})}$ (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 $\text{Local}^{\tilde{O}(\sqrt{n})}$ 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 tampering functions that are both local and split-state, as we discuss next. We believe that both $\text{Local}^{\ell_o}_{\ell_i}$ and $\text{Local}^{\ell_o}$ 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 [5], is also a special case of our class, as it is a subset of $\text{Local}^1$ (in fact, even a subset of $\text{Local}^1_1$). The restricted linear tampering class considered by [10], 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., [14] achieve rate $1-o(1)$ non-malleable codes for bit-wise tampering functions), while we only achieve a constant rate for these classes.

We also mention that the original work of Dziembowski et al. [24] 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 [24] (and an improvement in [13]), 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 encoding procedure must be randomized, and since non-malleable codes are only secure in the one-time setting—each time the codeword is decoded it must immediately 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 $\text{Local}^{\ell_o}$ for any $\ell_o \leq n^{1/4}$, using *deterministic* decoding. 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 $\text{Local}^{n^{\delta}}$ or $\text{Local}^{o(n/\log n)}$ without efficiency).

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 $c^{\mathsf{err}}$), and enjoys some additional secret-sharing like properties: given a (smaller) constant fraction $c^{\mathsf{sec}}$ 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 [8] 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., [15,16,19,21].

**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 $(\mathsf{E}, \mathsf{D})$ 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 $(\mathsf{E}, \mathsf{D})$ 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 $s^{\mathsf{L}}$ and $f^R$ only depended on $s^{\mathsf{R}}$, 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 $s^{\mathsf{R}}$ on $f^L$ and $s^{\mathsf{L}}$ on $f^R$.

Let's start with $f^L$, and notice that if its output locality is at most $\ell_o$, then at most $n_L \ell_o$ bits from $s^{\mathsf{R}}$ 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 $\mathsf{R}$), and now $f^L$ will only depend on $s^{\mathsf{L}}$, while $s^{\mathsf{R}}$ can still be completed to a correctly distributed encoding of $\mathsf{R}$. 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 $\ell_i$, then at most $n_L\ell_i$ bits on the right side can be influenced by $s^{\mathsf{L}}$.

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 $s^{\mathsf{L}}$ arbitrarily when computing $f^R$ and the resulting encoding would still be decoded to the same initial value $\mathsf{R}$. While this argument works if the only changes made to $s^{\mathsf{R}}$ (a valid codeword) are caused by the "crossing bits" from $s^{\mathsf{L}}$, it fails to take into account that $f^R$ can in fact apply other changes inside $s^{\mathsf{R}}$, and so it could be that $s^{\mathsf{R}}$ is malformed in such a way that applying $f^R$ will cause it to decode differently in a way that crucially depends on $s^{\mathsf{L}}$. 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 [15], and a forthcoming version by the same authors [16], 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 $\mathsf{R}$. 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 [15,16], 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 $s^{\mathsf{L}}$, we can show that with high probability, there is a large enough "clean" subset of $s^{\mathsf{R}}$ that has *no* influence from $s^{\mathsf{L}}$. 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 $s^{\mathsf{L}}$, 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, $\ell_o$. 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 [22] and has been applied widely in cryptography since. Although it was defined in 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 [27–29].

Dziembowski, Pietrzak and Wichs [24] 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, [23] presented an information theoretic non-malleable code for 1-bit messages against 2 state split state functions, followed by [3], 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 [3,12]. This line of research culminated with the result of [2], 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). [1] improve this to (optimal) rate 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 construction in a random-oracle model). In particular, Dziembowski et al. [24] gave an existential proof for the existence non-malleable codes secure against any 'small-enough' tampering family ($<2^{2^n}$). [13,26] 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, [17], 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 inefficient general constructions and efficient constructions for bit-wise functions and generalizations. [31], 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. [11,20] consider a variant of non-malleable codes that is also locally decodable and updatable. [25] allow continual tampering, and [4] allow for bounded leakage model. As discussed previously, [10] considers a subclass of linear tampering functions. We guide the interested reader to [30,31] for illustrative discussion of various models.

## 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 $\Sigma$), then $x_{i:j} := (x_i, x_{i+1}, \ldots, x_{j-1}, x_j)$ for $i \leq j$. If $\Sigma$ is a set, then $\Sigma^\Sigma := \{f : \Sigma \to \Sigma\}$, the set of all functions from $\Sigma$ to $\Sigma$. We say two vectors $x, y \in \Sigma^n$ are $\varepsilon$-*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 $\varepsilon$-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).** *[24] A* Coding scheme, $(\mathsf{E}, \mathsf{D})$, *consists of a randomized encoding function* $\mathsf{E} : \{0,1\}^k \mapsto \{0,1\}^n$ *and a randomized decoding function* $\mathsf{D} : \{0,1\}^n \mapsto \{0,1\}^k \cup \{\perp\}$ *such that* $\forall x \in \{0,1\}^k, \Pr[\mathsf{D}(\mathsf{E}(x)) = x] = 1$ *(over randomness of* $\mathsf{E}$ *and* $\mathsf{D}$*).*

We note that this definition differs from the original one given in [24], 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).** *[2] 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:*

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

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

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

*and $Tamper_x^f$ is at most $\varepsilon$.*

The above of definition has its origins in [24]. 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. [2] Their definition is based on the notion of non-malleable reduction, which we will use.

**Definition 3 (Non-Malleable Reduction).** *[2] 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 $\mathsf{E}, \mathsf{D}$).*
*(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 $(\mathsf{E}, \mathsf{D})$ is an $(\mathcal{F}, \mathcal{G}, \varepsilon)$-*non-malleable reduction.

**Definition 4 (Non-Malleable Code).** *[2] Let $\mathrm{NM}_k$ 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 $(\mathsf{E}, \mathsf{D})$ 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).** *[2] 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 [2] goes through unchanged with randomized decoding.

### 2.3 Tampering Families

**Definition 5 (Split-State Model).** *[24] The* split-state model, $\mathrm{SS}_k$, *denotes the set of all functions:*

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

**Theorem 2 (Split-State Non-malleable Codes with Constant Rate).** *[2] 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 [7].

**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 $\ell$ *if every input bit* $f_i$ *is affects at most* $\ell$ *output bits.*

**Definition 9 (Local Functions).** *[7] A function* $f : \{0,1\}^n \to \{0,1\}^n$ *is said to be* $(m, \ell)$*-local,* $f \in \mathrm{Local}_\ell^m$*, if it has input locality* $\ell$ *and output locality* $m$*. We denote the class* $\mathrm{Local}_n^m$ *(namely no restriction on the input locality) by* $\mathrm{Local}^m$.

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 $\mathsf{NC}^0$ 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 [15], as well as in [16]), extending a definition given by Decatur, Goldreich and Ron [21]. 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 [8].

**Definition 10 (Binary Reconstructable Probabilistic Encoding).** *[15,16] 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* $\mathsf{E}$*, any two strings in* $\mathcal{W}$ *are* $2c^{\mathsf{err}}$*-far. Moreover,* $\mathsf{D}$ *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.*** $\mathsf{Rec}$ *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 [21] meets the above requirements.

**Lemma 1.** *[15,16] 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 $c^{\mathsf{err}}$ and $c^{\mathsf{sec}}$ parameters, one can simply pad the message to an appropriate length.

Specifically, Decatur, Goldreich and Ron [21] construct a probabilistic encoding scheme that possesses the first two properties listed above. Moreover, since the construction they present, instantiates $\mathsf{E}$ 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. [18] (see also [14]).

### 2.5 Boolean Function Restrictions

The following two definitions are special cases of Boolean function restrictions. It will be convenient to have explicit notation for restrictions of Boolean functions $f$ where the input/output of the function $f$ has a particular form.

**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.** *[32] A generator $\mathsf{prg} : \{0,1\}^m \to (\{0,1\}^n)^k$ is a pseudorandom generator for space($w$) and block size $n$ with parameter $\varepsilon$ 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.** *[32] 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 space($w$) with parameter $2^{-cn}$. Moreover, $\mathsf{prg}$ can be computed in polynomial time (in $n, 2^k$).*

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

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

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

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

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

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

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

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

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

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

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 $\ell_{\mathsf{sec}}, n_{\mathsf{check}}$ be parameters.

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

1. Compute $(s_1^{\mathsf{L}}, \ldots, s_{n_L}^{\mathsf{L}}) \leftarrow \mathsf{E}_L(\mathsf{L})$ and $(s_1^{\mathsf{R}}, \ldots, s_{n_R}^{\mathsf{R}}) \leftarrow \mathsf{E}_R(\mathsf{R})$.
2. Output the encoding $(\boldsymbol{s}^{\mathsf{L}}, \boldsymbol{s}^{\mathsf{R}}) := ([s_i^{\mathsf{L}}]_{i\in[n_L]}, [s_i^{\mathsf{R}}]_{i\in[n_R]})$.

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

1. Let $(\boldsymbol{\sigma}^{\mathsf{L}}, \boldsymbol{\sigma}^{\mathsf{R}}) := ([\sigma_i^{\mathsf{L}}]_{i\in[n_L]}, [\sigma_i^{\mathsf{R}}]_{i\in[n_R]})$.
2. 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 $\mathsf{L} := \perp$.
3. (`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 $\mathsf{R} := \perp$.
4. (`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 $\mathsf{R} := \perp$.
5. (`output`) Output $x := (\mathsf{L}, \mathsf{R})$.

**Fig. 1.** The $(\mathrm{Local}_{\ell_i(k)}^{\ell_o(k)}, \mathrm{SS}, \mathsf{negl}(k))$-non-malleable reduction $(\mathsf{E}, \mathsf{D})$

**Observation 1.** *For $f \in \mathrm{Local}_{\ell_i}^{\ell_o}$, 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 $|\mathcal{S}_{\mathsf{L}\to\mathsf{R}}| \le \ell_i \cdot n_L \le n_R - t$.)*
2. *By choice of parameters $n_L, n_{\mathsf{check}}, c_L^{\mathsf{sec}}$, we have that $|\mathcal{S}_{\mathsf{check}}| \le n_L \cdot c_L^{\mathsf{sec}}$. (Since $\mathcal{S}_{\mathsf{check}} \le \ell_o \cdot n_{\mathsf{check}}$.)*
3. *By choice of parameters $n_L, n_R, c_R^{\mathsf{sec}}$, we have that $|\mathcal{S}_{\mathsf{R}\to\mathsf{L}}| \le \ell_o \cdot n_L \le n_R \cdot c_R^{\mathsf{sec}}$.*

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

- Choose a random subset $R_{\mathsf{check}} \subseteq [n_R]$ of size $n_{\mathsf{check}}$.
- 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 $g := (g_L, g_R) \in \mathrm{SS}_k$ has $I^{\mathsf{L}}, I^{\mathsf{R}}$ hardcoded into it and is specified as follows:

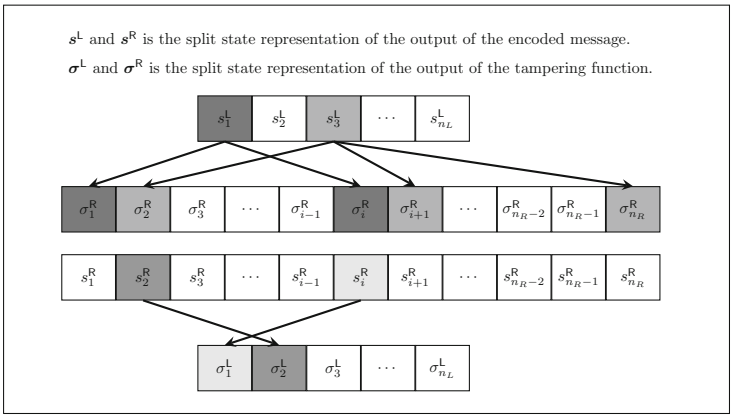


**Fig. 2.** The adversary chooses tampering function $f = (f^{\mathsf{L}}, f^{\mathsf{R}}) \in \mathrm{Local}^{\ell_o(k)}_{\ell_i(k)}$ which takes inputs $(\boldsymbol{s}^{\mathsf{L}}, \boldsymbol{s}^{\mathsf{R}})$ and produces outputs $(\boldsymbol{\sigma}^{\mathsf{L}}, \boldsymbol{\sigma}^{\mathsf{R}})$. The highlighted bits of $\boldsymbol{s}^{\mathsf{L}}$ and $\boldsymbol{s}^{\mathsf{R}}$ are the "bad" bits. E.g. note that bits $s^{\mathsf{R}}_2$ and $s^{\mathsf{R}}_i$ *affect* the output bits $\sigma^{\mathsf{L}}_2$ and $\sigma^{\mathsf{L}}_1$ respectively after $f^{\mathsf{L}}$ 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 $s^{\mathsf{L}}_1$ and $s^{\mathsf{L}}_3$ *affect* the bits $\{\sigma^{\mathsf{R}}_1, \sigma^{\mathsf{R}}_i\}$ and the bits $\{\sigma^{\mathsf{R}}_2, \sigma^{\mathsf{R}}_{i+1}, \sigma^{\mathsf{R}}_{n_R,\}}$ respectively after the tampering function $f^{\mathsf{R}}$ is applied to $(\boldsymbol{s}^{\mathsf{L}}, \boldsymbol{s}^{\mathsf{R}})$. We therefore add 1 to the sets $\mathcal{S}^1_{\mathsf{L}\to\mathsf{R}}$ and $\mathcal{S}^i_{\mathsf{L}\to\mathsf{R}}$, while we add 3 to the sets $\mathcal{S}^2_{\mathsf{L}\to\mathsf{R}}, \mathcal{S}^{i+1}_{\mathsf{L}\to\mathsf{R}}$ and $\mathcal{S}^{n_R}_{\mathsf{L}\to\mathsf{R}}$. We also add both 1 and 3 to the set $\mathcal{S}_{\mathsf{L}\to\mathsf{R}}$.

$g_L(\mathsf{L})$:

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 $(\sigma^{\mathsf{L}}_1, \ldots, \sigma^{\mathsf{L}}_{n_L}) := f^{\mathsf{L}}|_{I^{\mathsf{R}}}(\boldsymbol{s}^{\mathsf{L}})$. Compute $((w^{\mathsf{L}}_1, \ldots, w^{\mathsf{L}}_{n_L}), \widetilde{\mathsf{L}}) \leftarrow \mathsf{D}_L(\sigma^{\mathsf{L}}_1, \ldots, \sigma^{\mathsf{L}}_{n_L})$. 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 $\boldsymbol{s}^{\mathsf{R}} = (s^{\mathsf{R}}_1, \ldots, s^{\mathsf{R}}_{n_R}) := \mathsf{Rec}(\mathcal{S}_{\mathsf{R}\to\mathsf{L}}, I^{\mathsf{R}}, \mathsf{R})$. Let $(\sigma^{\mathsf{R}}_1, \ldots, \sigma^{\mathsf{R}}_{n_R}) := f^{\mathsf{R}}|_{I^{\mathsf{L}}}(\boldsymbol{s}^{\mathsf{R}})$. 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 $\ell \in [J^*]$. Set $\sigma'^{\mathsf{R}}_\ell := 0$ for $\ell \notin [J^*]$. Compute $((w^{\mathsf{R}}_1, \ldots, w^{\mathsf{R}}_{n_R}), \widetilde{\mathsf{R}}) \leftarrow \mathsf{D}_R(\sigma'^{\mathsf{R}}_1, \ldots, \sigma'^{\mathsf{R}}_t)$. If the decoding fails or $(w^{\mathsf{R}}_1, \ldots, w^{\mathsf{R}}_{n_R})$ is not $c^{\mathsf{err}}_R/4$-close to $(\sigma^{\mathsf{R}}_1, \ldots, \sigma^{\mathsf{R}}_{n_R})$, then set $\widetilde{\mathsf{R}} := \perp$.
2. (`codeword-check on right`) For all $\ell \in R_{\mathsf{check}}$, check that $\sigma^{\mathsf{R}}_\ell = w^{\mathsf{R}}_\ell$. If the check fails, set $\widetilde{\mathsf{R}} := \perp$.
3. (`output`) Output $\widetilde{\mathsf{R}}$.

– Output $g = (g_L, g_R)$.

Whenever $\mathsf{Rec}$ 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.

By the definition of a non-malleable reduction (Definition 3), in order to complete the proof of Theorem 4, we must show that $(\mathsf{E}, \mathsf{D})$ 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 $f \in \mathrm{Local}^{\ell_o}_{\ell_i}$,

$$\Delta(\mathsf{D}(f(\mathsf{E}(x))); G_f(x)) \leq \mathsf{negl}(k),$$

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 $f \in \mathrm{Local}^{\ell_o}_{\ell_i}$ (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 $\mathsf{D}'$ is defined as follows:

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

1. (`plain decode on left`) Let $(\boldsymbol{\sigma}^{\mathsf{L}}, \boldsymbol{\sigma}^{\mathsf{R}}) := ([\sigma^{\mathsf{L}}_i]_{i\in[n_L]}, [\sigma^{\mathsf{R}}_\ell]_{\ell\in[n_R]})$. 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 $\mathsf{L} := \perp$.

2. (`decoding-check on right`) 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 $\ell \in J^*$ and $\sigma'^{\mathsf{R}}_\ell := 0$ for $\ell \notin J^*$, where $J^* \subseteq [n_R]$ is the lexicographically first set such that $|J^*| = t$ and $|\mathcal{S}^{J^*}_{\mathsf{L}\to\mathsf{R}}| = 0$. Compute $((w^{\mathsf{R}}_1, \ldots, w^{\mathsf{R}}_{n_R}), \mathsf{R}) \leftarrow \mathsf{D}_R(\sigma'^{\mathsf{R}}_1, \ldots, \sigma'^{\mathsf{R}}_{t_R})$. If the decoding fails or $(w^{\mathsf{R}}_1, \ldots, w^{\mathsf{R}}_{n_R})$ is not $c^{\mathsf{err}}_R/4$-close to $(\sigma^{\mathsf{R}}_1, \ldots, \sigma^{\mathsf{R}}_{t_R})$, set $\mathsf{R} := \perp$.

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

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

*Claim.*

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

*Proof.* Let $\delta := \frac{c^{\mathsf{err}}_R}{4}$. 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 $\rho(n_R, \delta, n_{\mathsf{check}}) = \mathsf{negl}(k)$.

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

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 $\mathsf{D}'$ 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 $\mathsf{D}, \mathsf{D}'$.

Let $\mathsf{D} := (\mathsf{D}^{\mathsf{L}}, \mathsf{D}^{\mathsf{R}})$ (respectively, $\mathsf{D}' := (\mathsf{D}'^{\mathsf{L}}, \mathsf{D}'^{\mathsf{R}})$), where $\mathsf{D}^{\mathsf{R}}$ (respectively, $\mathsf{D}'^{\mathsf{R}}$) 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 $\mathsf{D}, \mathsf{D}'$.

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 $\perp$).

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

So, now consider the two cases:

- Suppose $\boldsymbol{\sigma}^{\mathsf{R}} \in \mathit{\Pi}_{\perp}$. 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 $\boldsymbol{\sigma}^{\mathsf{R}} \notin \mathit{\Pi}_{\perp}$. Then, $\exists w \in \mathcal{W}$ such that $d(\boldsymbol{\sigma}^{\mathsf{R}}, w) \leq \delta$. Moreover, in *both* $\mathsf{D}$ and $\mathsf{D}'$ 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 $\mathsf{D}$ and $\mathsf{D}'$ must decode to the *same* $w$. Fix a set of coins for $\mathsf{D}$ and $\mathsf{D}'$. Therefore, when $\mathsf{D}$ and $\mathsf{D}'$ are run with the same coins, all comparisons made during the codeword-check are identical, and thus the probability (over the coins of $\mathsf{D}, \mathsf{D}'$) that the codeword-check fails in $\mathsf{D}$ and $\mathsf{D}'$ is identical.

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

*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 $n_{\mathsf{check}}$.

– Choose vectors $I^{\mathsf{L}} \in \{0,1\}^{n_L}$, $I^{\mathsf{R}} \in \{0,1\}^{n_R}$ 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 $I^{\mathsf{R}}$ and $g'_R$ depends on $I^{\mathsf{L}}$.

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

### 3.1 Extending to Leaky Local

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

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

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 $\text{Local}^{\ell_o}$ where all but a fraction of the first $n_L$ bits have input locality $\ell_i$. 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* $\text{LL} \subseteq \{\{0,1\}^{n_L} \times \{0,1\}^{n_R} \to \{0,1\}^{n_L} \times \{0,1\}^{n_R}\}$, 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 $c^{\mathsf{err}}_L/3$ of its input bits.*
- *On input $\boldsymbol{s}^{\mathsf{R}} \in \{0,1\}^{n_R}$, $h_2$ outputs a subset of $c^{\mathsf{err}}_R/3$ of its input bits.*
- *On input $h_1(\boldsymbol{s}^{\mathsf{L}}), h_2(\boldsymbol{s}^{\mathsf{L}}) \in \{0,1\}^{c^{\mathsf{err}}_L/3} \times \{0,1\}^{c^{\mathsf{err}}_R/3}$, $f$ 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 $\ell_o$. Of the first $n_L$ input bits, all but at most $c^{\mathsf{err}}_L/3$-fraction have input-locality at most $\ell_i$.*

The following corollary can be easily verified.

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

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

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

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

- $\ell_o{}' := c^{\mathsf{sec}}/12 \cdot \ell_i$, *where $\ell_i$ 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 Fig. 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 $(\text{Local}^{\ell_o{}'}, \text{LL}, \mathsf{negl}(n))$-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 $\text{SS}_k$, the encoding 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 $\text{Local}^{\ell_o{}'}$.

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

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

We parametrize our construction for $\text{Local}^{\ell_o{}'} \Rightarrow \text{LL}$ 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 $\mu \in \text{Local}^{\ell_o{}'}$ 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 $G_\mu$ over LL. A draw from $G_\mu$ 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 $\rho$ has less than $n_L$ number of ones, then set $h_1, h_2, f$ all to the constant function 0.
- Otherwise, choose vector $I^{\mathsf{L}} \in \{0,1\}^{+n_R}$ such that $\forall i$ such that $1 \leq i \leq$ if $\rho_i = 1$ then $I_i^{\mathsf{L}} = *$ and otherwise, $I_i^{\mathsf{L}}$ is chosen uniformly at random.
- The function $h_1$ is defined as follows: $h_1$ outputs the bits in input $x^{\mathsf{L}}$ that affect the output bits of $\mu^{\zeta}$ (at most $r \cdot \ell_o{}' \leq c_L^{\mathsf{sec}}/3 \cdot n_L$).
- The function $h_2$ is defined as follows: $h_2$ outputs the bits in $x^{\mathsf{R}}$ that affect the output bits of $\mu^{\zeta}$ (at most $r \cdot \ell_o{}' \leq c_R^{\mathsf{sec}}/3 \cdot n_R$).
- The function $f$ is defined as follows:
  - $f$ computes $\widetilde{\zeta}$, given $\zeta$ 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 $\mathsf{pos}^* < i \leq \tau, \widetilde{\rho}^* = 0$, where $\mathsf{pos}^*$ is the index of the $n_L$-th one in $\widetilde{\rho}$ (and is set to $\tau$ 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 $\zeta$ hardcoded in it.

Let $\mathsf{prg}$ 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 $\phi$ 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 $\phi$. Let $\rho = \rho_1, \ldots, \rho_\tau$. Let $\mathsf{num}$ denote the number of positions of $\rho$ that are set to 1.
3. If $\mathsf{num} < n_L$, set $\rho := 1^n_L 0^{\tau - n_L}$.
4. Otherwise, flip all but the first $n_L$ 1's in $\rho$ to 0.
5. Output $\rho$.

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

$\mathsf{E}'(x^{\mathsf{L}} := x^{\mathsf{L}}_1, \ldots, x^{\mathsf{L}}_{n_L}, x^{\mathsf{R}})$:

1. Choose $\zeta \leftarrow \{0,1\}^r$ uniformly at random. Choose $\zeta \leftarrow \{0,1\}^r$ uniformly at random. Compute $\rho := \mathrm{G}(\zeta)$.
2. For $j \in [\mathsf{num}]$, let $\mathsf{pos}_j$ 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 $j \in [n_L]$, $X^{\mathsf{L}}_{\mathsf{pos}_j} := x^{\mathsf{L}}_j$. In all other locations, $X^{\mathsf{L}}_i$ 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 $\mathsf{pos}_j$ denote the $j$-th position $i$ such that $\widetilde{\rho}_i = 1$.
3. Let $\widetilde{x}^{\mathsf{L}}_j \in \{0,1\}^{n_L}$ be defined in the following way: For $j \in [\min(\widetilde{\mathsf{num}}, n_L)]$, $\widetilde{x}^{\mathsf{L}}_j := \widetilde{X}^{\mathsf{L}}_{\mathsf{pos}_j}$.
4. (`output`) Output $(\widetilde{x}^{\mathsf{L}}, \widetilde{x}^{\mathsf{R}})$.

**Fig. 3.** The $(\mathrm{Local}^{\ell_o{}'}, \mathrm{LL}, \mathsf{negl}(n))$-non-malleable reduction $(\mathsf{E}', \mathsf{D}')$

- 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 $\mu \in \text{Local}^{\ell_o{}'}$,

$$\Delta(\mathsf{D}'(\mu(\mathsf{E}'(x)));G_\mu(x)) \leq \mathsf{negl}(n),$$

where $G_\mu$ 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 $\rho$ does not contain at least $n_L$ number of ones or $C$ is not in LL.

We first argue that if $\rho$ 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 $\rho$ is chosen via a PRG with appropriate security guarantees.

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

We now argue that these events must also occur with negligible probability when $\rho$ 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 $\mu$ 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 $y = y_1, \ldots, y$ (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 :
   (a) Run $\phi(y_i)$ to obtain $\rho_i$.
   (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, $\mathsf{prg}$ 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

Here we sketch how to improve parameters. We refer readers to the full paper for the complete proof.

**Theorem 6.** *There exists an explicit* $(\text{Local}^{\ell_o} \Rightarrow SS, \mathsf{negl}(n))$*-non-malleable reduction,* $(\mathsf{E} : \{0,1\}^{2k} \to \{0,1\}^n, \mathsf{D} : \{0,1\}^n \to \{0,1\}^{2k})$*, for any* $\ell_o = o(n/\log n)$ *where* $n = O(\ell_o k)$*.*

Roughly, the reduction $(\mathsf{E}, \mathsf{D})$ is simply a composition of the reductions presented previously. Recall that the encoding scheme is independent on the left and right, $\mathsf{E}(\mathsf{L}, \mathsf{R}) = (\mathsf{E}_L(\mathsf{L}), \mathsf{E}_R(\mathsf{R}))$. The left side, $\mathsf{E}_L(\mathsf{L})$, is comprised of a seed for a PRG that describes where to pseudorandomly embedded a (small) RPE of $\mathsf{L}$ and that very embedding. The right side, $\mathsf{E}_R(\mathsf{R})$, is simply a (longer) RPE of $\mathsf{R}$. Decoding is the same as before as well. The parameters are slightly different, but we will gloss over that here.

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 $\ell_o$ 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, $(\mathsf{L}, \mathsf{R})$).

We will present the "good" event described above and sketch the hybrid argument.

**Definition 14 (informal).** *The event* $\mathsf{Good}_f$ *occurs if for tampering function* $f \in \text{Local}^{\ell_o}$ *all of the following hold:*

1. *$\rho$ contains at least $n_L$ ones, where $n_L$ is the length of the left side RPE.*
2. *$|S|$ is below the security threshold of the left-side RPE, where $S$ is the set of bits in the "hidden" RPE of $\mathsf{L}$ that have (1) "high" input locality, (2) effect the consistency check on the right (consider this chosen secretly and randomly at the time of encoding), or (3) effect the PRG seed used in decoding.*

3. *There is some (large enough) set, $J^*$, of bits that is not effected by any bit in the RPE of* $\mathsf{L}$ *which does not have "high" input locality.*
4. *The bits on the right that effect the output of decoding on the left is below the security threshold of the RPE.*

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

The first two items follow from Chernoff bounds. The main difference in the new analysis is that the hidden RPE of $\mathsf{L}$ is now very small size. Whereas previously the events (2), (3) described in the second item held simply because the total number of bits on the left affecting the consistency check and PRG seed was below the security threshold of the left RPE, now, since the left RPE is now very small, we must rely on Chernoff bounds and the fact that the relevant bits are hidden to argue that (2) and (3) hold. The second two items are similar to Observation 1, given item (2).

Next as in the previous section, we argue that with high probability, the pseudorandomness of the PRG is sufficient to obtain that the event $\mathsf{Good}_f$ holds even when $\rho$ is chosen via the PRG (instead of being truly random). This gives us the bounds on the "bad" bits in the output of the encoding of the left input, $\mathsf{L}$, mentioned previously.

Now we are in essentially a similar situation to the proof of Theorem 4 and we can apply a very similar sequence of hybrids.

First, we use hybrids to effectively sample the bits on the left and the right that effect some other block, or are in the "bad" set $S$. By our claim above, all of these sets together will be below the security properties of the respective RPEs. So, the distribution over the randomness of the encoding procedure will be identical, for any message.

Second, we use hybrids to effectively simulate decoding on the right from the set $J^*$ that is not effected by the RPE on the left. This completes the proof.

**Acknowledgments.** We thank Seung Geol Choi and Hoeteck Wee for sharing with us an in-submission journal version of [15], as well as the manuscript [16]. We also thank Yevgeniy Dodis for helpful discussions and clarifications regarding [2] and other previous work. Finally, we thank Eran Tromer for enlightening discussions on practical tampering attacks, which inspired the class of attacks considered in this work.

This work was done in part while all authors were visiting the Simons Institute for the Theory of Computing, supported by the Simons Foundation and by the DIMACS/Simons Collaboration in Cryptography through NSF grant #CNS-1523467. The first and fourth authors are supported in part by the Defense Advanced Research Project Agency (DARPA) and Army Research Office (ARO) under Contract #W911NF-15-C-0236, and NSF grants #CNS-1445424 and #CCF-1423306. The second and third authors are supported by an NSF CAREER award #CNS-1453045 and by a Ralph E. Powe Junior Faculty Enhancement Award. Any opinions, findings and conclusions or recommendations expressed are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency, Army Research Office, the National Science Foundation, or the U.S. Government.

## References


1. Aggarwal, D., Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Optimal computational split-state non-malleable codes. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A. LNCS, vol. 9563, pp. 393–417. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49099-0_15
2. Aggarwal, D., Dodis, Y., Kazana, T., Obremski, M.: Non-malleable reductions and applications. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th ACM STOC, Portland, OR, USA, 14–17 June 2015, pp. 459–468. ACM Press (2015)
3. Aggarwal, D., Dodis, Y., Lovett, S.: Non-malleable codes from additive combinatorics. In: Shmoys, D.B. (ed.) 46th ACM STOC, NY, USA, May 31–June 3, 2014, pp. 774–783 (2014)
4. Aggarwal, D., Dziembowski, S., Kazana, T., Obremski, M.: Leakage-resilient non-malleable codes. Cryptology ePrint Archive, Report 2014/807 (2014). http://eprint.iacr.org/2014/807
5. Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Explicit non-malleable codes against bit-wise tampering and permutations. In: Gennaro, R., Robshaw, M. (eds.) CRYPTO 2015. LNCS, vol. 9215, pp. 538–557. Springer, Heidelberg (2015). doi:10.1007/978-3-662-47989-6_26
6. Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 375–397. Springer, Heidelberg (2015)
7. Applebaum, B.: Cryptography in Constant Parallel Time. Information Security and Cryptography. Springer, Heidelberg (2014). http://dx.doi.org/10.1007/978-3-642-17367-7
8. Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: 20th ACM STOC, Chicago, Illinois, USA, 2–4 May 1988, pp. 1–10. ACM Press (1998)
9. Chabanne, H., Cohen, G.D., Flori, J., Patey, A.: Non-malleable codes from the wire-tap channel. CoRR abs/1105.3879 (2011). http://arxiv.org/abs/1105.3879
10. Chabanne, H., Cohen, G.D., Patey, A.: 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, 1–6 July 2012, pp. 2546–2550. IEEE (2012). http://dx.doi.org/10.1109/ISIT.2012.6283976
11. Chandran, N., Kanukurthi, B., Raghuraman, S.: Information-theoretic local non-malleable codes and their applications. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A. LNCS, vol. 9563, pp. 367–392. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49099-0_14
12. Chattopadhyay, E., Zuckerman, D.: Non-malleable codes against constant split-state tampering. In: 55th FOCS, Philadelphia, PA, USA, 18–21 October 2014, pp. 306–315. IEEE Computer Society Press (2014)
13. Cheraghchi, M., Guruswami, V.: Capacity of non-malleable codes. In: Naor, M. (ed.) ITCS, Princeton, NJ, USA, 12–14 January 2014, pp. 155–168. ACM (2014)
14. Cheraghchi, M., Guruswami, V.: Non-malleable coding against bit-wise and split-state tampering. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 440–464. Springer, Heidelberg (2014)
15. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.M.: Black-box construction of a non-malleable encryption scheme from any semantically secure one. In: Canetti, R. (ed.) TCC 2008. LNCS, vol. 4948, pp. 427–444. Springer, Heidelberg (2008)

16. Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: A note on improved, black-box constructions of non-malleable encryption from semantically-secure encryption. Manuscript (2015)
17. Choi, S.G., Kiayias, A., Malkin, T.: BiTR: built-in tamper resilience. In: Lee, D.H., Wang, X. (eds.) ASIACRYPT 2011. LNCS, vol. 7073, pp. 740–758. Springer, Heidelberg (2011)
18. Coretti, S., Dodis, Y., Tackmann, B., Venturi, D.: Non-malleable encryption: Simpler, shorter, stronger. Cryptology ePrint Archive, Report 2015/772 (2015). http://eprint.iacr.org/2015/772
19. Coretti, S., Dodis, Y., Tackmann, B., Venturi, D.: Non-malleable encryption: simpler, shorter, stronger. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A. LNCS, vol. 9562, pp. 306–335. Springer, Heidelberg (2016). doi:10.1007/978-3-662-49096-9_13
20. Dachman-Soled, D., Liu, F.-H., Shi, E., Zhou, H.-S.: Locally decodable and updatable non-malleable codes and their applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015, Part I. LNCS, vol. 9014, pp. 427–450. Springer, Heidelberg (2015)
21. Decatur, S.E., Goldreich, O., Ron, D.: Computational sample complexity. SIAM J. Comput. **29**(3), 854–879 (2000)
22. Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM J. Comput. **30**(2), 391–437 (2000)
23. Dziembowski, S., Kazana, T., Obremski, M.: Non-malleable codes from two-source extractors. In: Canetti, R., Garay, J.A. (eds.) CRYPTO 2013, Part II. LNCS, vol. 8043, pp. 239–257. Springer, Heidelberg (2013)
24. Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: Yao, A.C.C. (ed.) ICS, 5–7 January 2010, pp. 434–452. Tsinghua University Press, Tsinghua University, Beijing, China (2010)
25. Faust, S., Mukherjee, P., Nielsen, J.B., Venturi, D.: Continuous non-malleable codes. In: Lindell, Y. (ed.) TCC 2014. LNCS, vol. 8349, pp. 465–488. Springer, Heidelberg (2014)
26. Faust, S., Mukherjee, P., Venturi, D., Wichs, D.: Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In: Nguyen, P.Q., Oswald, E. (eds.) EUROCRYPT 2014. LNCS, vol. 8441, pp. 111–128. Springer, Heidelberg (2014)
27. Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (ATP) security: theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC 2004. LNCS, vol. 2951, pp. 258–277. Springer, Heidelberg (2004)
28. Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits II: keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) EUROCRYPT 2006. LNCS, vol. 4004, pp. 308–327. Springer, Heidelberg (2006)
29. Ishai, Y., Sahai, A., Wagner, D.: Private circuits: securing hardware against probing attacks. In: Boneh, D. (ed.) CRYPTO 2003. LNCS, vol. 2729, pp. 463–481. Springer, Heidelberg (2003)
30. Kalai, Y.T., Kanukurthi, B., Sahai, A.: Cryptography with tamperable and leaky memory. In: Rogaway, P. (ed.) CRYPTO 2011. LNCS, vol. 6841, pp. 373–390. Springer, Heidelberg (2011)
31. Liu, F.-H., Lysyanskaya, A.: Tamper and leakage resilience in the split-state model. In: Safavi-Naini, R., Canetti, R. (eds.) CRYPTO 2012. LNCS, vol. 7417, pp. 517–532. Springer, Heidelberg (2012)
32. Nisan, N.: Pseudorandom generators for space-bounded computation. Combinatorica **12**(4), 449–461 (1992)