Non-malleable codes with optimal rate for poly-size circuits

Marshall Ball[*] Ronen Shaltiel[†] Jad Silbak[‡]

November 13, 2023

Abstract

We give an explicit construction of non-malleable codes with rate $1 - o(1)$ for the tampering class of poly-size circuits. This rate is optimal, and improves upon the previous explicit construction of Ball, Dachman-Soled and Loss [BDL22] which achieves a rate smaller than $\frac{1}{n}$. Our codes are based on the same hardness assumption used by Ball, Dachman-Soled and Loss, namely, that there exists a problem in $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ that requires nondeterministic circuits of size $2^{\Omega(n)}$. This is a standard complexity theoretic assumption that was used in many papers in complexity theory and cryptography, and can be viewed as a scaled, nonuniform version of the widely believed assumption that EXP $\not\subseteq$ NP. Our result is incomparable to that of Ball, Dachman-Soled and Loss, as we only achieve computational (rather than statistical) security. Non-malleable codes with Computational security (with lower error than what we get) were obtained by [BDK+19, DKP21] under strong cryptographic assumptions. We show that our approach can potentially yield statistical security if certain explicit constructions of pseudorandom objects can be improved.

By composing our new non-malleable codes with standard (information theoretic) error-correcting codes (that recover from a p fraction of errors) we achieve the *best of both worlds*. Namely, we achieve explicit codes that recover from a p-fraction of errors and have the same rate as the best known explicit information theoretic codes, while *also* being non-malleable for poly-size circuits.

Moreover, if we restrict our attention to errors that are introduced by poly-size circuits, we can achieve best of both worlds codes with rate $1 - H(p)$. This is superior to the rate achieved by standard (information theoretic) error-correcting codes, and this result is obtained by composing our new non-malleable codes with the recent codes of Shaltiel and Silbak [SS23].

Our technique combines ideas from non-malleable codes and pseudorandomness. We show how to take a low rate "small set non-malleable code (this is a variant of non-malleable codes with a different notion of security that was introduced by Shaltiel and Silbak [SS22]) and compile it into a (standard) high-rate non-malleable code. Using small set non-malleable codes (as well as seed-extending PRGs) bypasses difficulties that arise when analysing standard non-malleable codes, and allows us to use a simple construction.

[*]New York University, Email:marshall.ball@cs.nyu.edu.
[†]University of Haifa, Email: ronen@cs.haifa.ac.il.
[‡]Northeastern University, Email: jadsilbak@gmail.com.

Contents

1 Introduction

1.1 Error correcting codes and non-malleable codes

Standard error-correcting codes. Coding theory studies message transmission in noisy channels. The objective is to design an error-correcting code (namely, a pair $(\mathrm{Enc}, \mathrm{Dec})$ of encoding and decoding algorithms) that correct from a specified number of errors. Error correction relies on adding redundancy to the message, and the code's rate is the ratio between the length of a message and the length of its encoding. The major goals of coding theory is to design codes with the largest possible rate for a specified number of errors, and achieve this with explicit constructions (namely, with poly-time encoding and decoding algorithms).

Non-malleable Codes. Non-malleable codes, introduced by Dziembowski, Pietrzak, and Wichs [DPW18], consider a more extreme scenario where there is no a-priori restriction on the number of bits that the channel may alter. Such a channel might choose to erase the encoded message and replace it with a different string. Obviously, in this case, one cannot expect the decoding algorithm to recover the original message. Instead, it is required that encoding and decoding satisfy the following:

- Recovery from no errors. If the channel does not alter the encoded message, then decoding produces the original message.
- Non-malleability. If the channel alters the encoded message, then the decoded message is either the original or unrelated to the original message.

The definition of non-malleable codes also includes two modifications over the standard coding scenario:

- It is impossible to handle all channels, and so, a non-malleable code is defined against a specific family $\mathcal{C}$ of channels (a.k.a "tampering functions").
- The encoding algorithm Enc is allowed to be randomized.

This leads to the following definition by [DPW18] (which is stated informally below, and more formally in Definition 2.6).

Definition 1.1 (non-malleable codes [DPW18], informal)**.** *A randomized encoding function* $\mathrm{Enc} : \{0,1\}^{Rn} \to \{0,1\}^n$ *and a deterministic decoding function* $\mathrm{Dec} : \{0,1\}^n \to \{0,1\}^{Rn} \cup \{\bot\}$ *form a rate* R*, non-malleable code against a class* $\mathcal{C}$ *of tampering functions, if for every* $m \in \{0,1\}^{Rn}$*,* $\mathrm{Dec}(\mathrm{Enc}(m)) = m$*, and for every* $C : \{0,1\}^n \to \{0,1\}^n$ *in* $\mathcal{C}$*, there exists a distribution* D_C *over* $\{0,1\}^{Rn} \cup \{same, \bot\}$*, such that for every* $m \in \{0,1\}^{Rn}$*, the experiment*

$$\mathrm{Tamper}_C(m) = \left\{ \begin{array}{c} z \leftarrow \mathrm{Enc}(m), v \leftarrow C(z), \bar{m} \leftarrow Dec(v) \\ \textit{Output } \bar{m}. \end{array} \right\}$$

is indistinguishable from the following experiment

$$\mathrm{Simulated}_C(m) = \left\{ \begin{array}{c} \bar{m} \leftarrow D_C \\ \textit{Output } m \textit{ if } \bar{m} = \mathrm{same}\textit{, and } \bar{m} \textit{ otherwise.} \end{array} \right\}$$

Note that in the simulated experiment, the distribution D_C does not depend on m, and this is the sense in which "$\bar{m}$ is unrelated to m".

Since the seminal work of Dziembowski, Pietrzak and Wichs [DPW18], various different classes of tampering functions were considered in the literature. Loosely speaking, these can be split into "tampering functions with limited information," such as bit-wise tampering [DPW18, CG14], split-state tampering [LL12,

DKO13, ADL14, CGL16, AAG+16, Li17, Li19, AO20, AKO+22, Li23], interleaved tampering [CL20]; and "tampering functions with restricted computational power," such as local tampering [BDKM16, GMW19, BGW19], AC^0 tampering [CL17, BDKM18, BGW19, BDG+18], low degree polynomial tampering [CL17, BCL+20], streaming tampering [BDKM18], and polynomial size circuit tampering [BDK+19, DKP21, BDL22]. This paper belongs to the latter category, and considers the family of tampering functions implemented by circuits of fixed polynomial size.

Non-malleable codes for poly-size circuits. Dziembowski, Pietrzak and Wichs, [DPW18] showed (by a probabilistic argument) that there exist non-malleable codes against poly-size circuit (in fact, against any "small" class of tampering functions). Later work by Faust et al. [FMVW16], and Cheraghchi and Guruswami [CG16], improved the probabilistic analysis and showed that there exist non-malleable codes against poly-size channel with rate $1 - o(1)$.

It is easy to see that the decoding function Dec of a non-malleable code against circuits of size n^c, cannot be computed by circuits of size n^c. This implies that in an explicit construction of non-malleable codes against size n^c circuit, we must allow the decoding algorithm to run in a polynomial time that is larger than n^c, and furthermore, that such an explicit construction implies circuit lower bounds (which are referred to as hardness assumptions).[1]

Ball, Dachman-Soled and Loss [BDL22] gave an explicit construction of non-malleable codes for poly-size circuits (based on a hardness assumption). They showed that for every constant $c > 1$, there is a pair of algorithms $(\mathrm{Enc}, \mathrm{Dec})$ that run in time $\mathrm{poly}(n^c)$, and form a non-malleable code against circuits of size n^c. The rate achieved by this construction is $\frac{1}{n^{\Theta(1)}}$ which is quite small.

Hardness assumptions against nondeterministic circuits. This construction relies on a standard hardness assumption from complexity theory: Namely, the assumption that E is hard for exponential size nondeterministic circuits. This assumption (stated precisely in Assumption 2.1) loosely says that there is a problem in $\mathrm{E} = \mathrm{DTIME}(2^{O(n)})$ that cannot be computed by nondeterministic circuits of size $2^{\Omega(n)}$. The assumption that E is hard for exponential size (deterministic) circuits was used by the celebrated work of Impagliazzo and Wigderson [IW97] to imply that BPP = P. The stronger assumption that E is hard for exponential size nondeterministric circuits was introduced for implying that AM = NP, and used in many papers in complexity theory and cryptography [KvM02, MV05, TV00, SU05, BOV07, GW02, GST03, SU06, SU09, Dru13, AASY15, BV17, AIKS16, HNY17, DMOZ22, BDL22, CT22]. This assumption can be viewed as a scaled, nonuniform version of the assumption: $\mathrm{EXP} \not\subseteq \mathrm{NP}$.

Other (nearly) explicit constructions of non-malleable codes for poly-size circuits were given by Ball, Dachman-Soled, Kulkarni, Lin, and Malkin [BDK+19] as well as by Dachman-Soled, Komargodski and Pass [DKP21]. These constructions follow a template laid out in [BDKM18] (whose codes, in contrast, were mostly in the common random string model) and rely on a variety of very strong cryptographic assumptions, particularly assumptions that are not known how to provably instantiate outside of the random oracle model.[2]

[1] Cheraghchi and Guruswamui [CG16] and Faust et al. [FMVW16] considered an intermediate notion of explicitness where the encoding and decoding algorithms also receive a uniformly chosen string of length $\mathrm{poly}(n^c)$ (which is chosen and published, once and for all, in a pre-processing stage) and the non-malleability is guaranteed w.h.p. over this random choice. This is termed a "Monte-Carlo construction" in coding theory, and a "construction in the CRS model" in cryptography.

[2] [BDK+19] relies on plain model P-certificates (with uniform soundness) among other assumptions and [DKP21] relies on keyless multi-collision resistant hash functions. Neither of these assumptions are known how to provably instantiate from standard cryptographic assumptions, which is why we refer to the codes as "nearly explict." Note that [BDK+19] also relies on the same hardness assumption made in this work and [BDL22] ([DKP21] uses a strong assumption about the hardness of repeated squaring in place of this hardness assumption).

Rate was not the focus of these works, and accordingly the codes achieve rate that is just $\frac{1}{n^{\Theta(1)}}$.[3]

1.2 Our results: non-malleable codes with optimal rate

In this paper we give an explicit construction of non-malleable codes for poly-size circuits that achieve optimal rate of $R = 1 - o(1)$, under the same hardness assumption used by Ball, Dachman-Soled and Loss [BDL22].

Theorem 1.2 (Non-malleable codes with rate $1 - o(1)$, informal)**.** *If E is hard for exponential size nondeterministic circuits, then for every constant c, there is a non-malleable code* $(\mathrm{Enc}, \mathrm{Dec})$ *against circuits of size* n^c*, with rate* $R = 1 - o(1)$*. Furthermore,* Enc *and* Dec *can be computed in time* $\mathrm{poly}(n^c)$.

Theorem 1.2 is stated in a more formal way in Theorem 3.1.

Improving the rate of non-malleable codes. In order to prove Theorem 1.2 we show how to improve the rate of a variant of non-malleable codes (called "small set non-malleable code"), and obtain (standard) non-malleable codes with rate $1 - o(1)$.[4]

More specifically, Shaltiel and Silbak [SS22] defined a notion called "small set non-malleable codes" (which is closely related to a notion of "bounded non-malleable codes" introduced by Faust et al. [FMVW16]). Loosely speaking, this notion of small-set non-malleability (which is incomparable to the standard notion of non-malleability) requires that for every tampering function C, there exists a small set H of messages, such that when C gets to corrupt the encoding of a uniformly chosen message m, it is unlikely that the decoding will produce a message that is neither in H, nor the original message m. (See Definition 2.8 for a precise formulation).

Loosely speaking, in small set non-malleability we are interested in the *variety* of outcomes a tampering function can effectively produce in the tampering experiment in which a uniform message is encoded. Note that a poly-size tampering circuit is *nonuniform*, and can be hardwired with a polynomial size set of messages and their codewords. This enables the tampering circuit to lead the decoding to produce any one of these small set of messages. Intuitively, for a small-set non-malleable code there is no markedly better approach than to implement this behavior. We remark that this security guarantee seems incomparable to the standard definition of non-malleable codes.[5]

Shaltiel and Silbak [SS23] showed how to convert the aforementioned non-malleable codes of [BDL22] into small set non-malleable codes (with the same parameters, and under the same hardness assumption). The result reported in Theorem 1.2 follows by applying our new "rate improvement technique" to the small set non-malleable codes of [SS23]. In Section 1.3 we elaborate on the ideas that are used in this "rate

[3] The conclusions of [BDL22] and [DKP21] are incomparable as the former achieves security with statistical indistinguishability, and the latter achieves only computational indistinguishability, but the latter achieves indistinguishability with a negligible advantage, whereas the former only achieves indistinguishability with an arbitrary fixed inverse polynomial advantage.

[4] We note that non-malleable code rate compilers for other tampering classes have been constructed in the past. Loosely speaking, these compilers as well as our own follow a similar framework akin to key-encapsulation mechanisms. However, they all differ dramatically in both construction and analysis. We refer the reader to Section 1.4 for more details.

[5] Loosely speaking, small-set non-malleable codes give a weaker form of non-malleability (called "bounded non-malleability in [FMVW16]) as they allow the tampering function to correlate the original message with the choice of which message in H is being decoded. On the other hand, the security definition in small-set non-malleable codes implies that the tampering function cannot produce an encoding of a uniform and independent random message (as this would allow it to break small-set non-malleability). This is in contrast to standard non-malleability which does not seem to rule out that the tampering function can compute the encoding (as demonstrated by Dachman-Soled, Komargodski and Pass [DKP21]). Another difference, is that the security definition of non-malleable codes rules out tampering functions that can compute the decoding algorithm, whereas small-set non-malleable codes do not. See discussion in [SS22].

improvement", and explain how we use the security guarantee of small-set non-malleable codes, and why the security of (standard) non-malleable codes does not seem to suffice.

Comparison of Theorem 1.2 with previous work. The non-malleable codes of Theorem 1.2 achieve rate $1 - o(1)$, but with a weaker security guarantee than that of [BDL22]. We only achieve computational indistinguishability in Definition 1.1, whereas [BDL22] achieves statistical indistinguishability. Additionally, like [BDL22], we only achieve a distinguishing advantage that is an arbitrary fixed inverse polynomial, rather than negligible (as is the case in [BDK+19, DKP21], which achieves negligible computational indistinguishability under very strong cryptographic assumptions). There are known barriers for achieving negligible (statistical) indistinguishabity from nondeterministic hardness assumptions (like the one we use) via black-box reductions [BDL22].[6]

Towards achieving statistical indistinguishability. While we do not obtain statistical indistinguishability in Theorem 1.2, our technique can potentially yield statistical indistinguishability (and the same security guarantee as in [BDL22]) if the parameters of certain "hard to sample functions" (HTS) that were introduced and constructed by Shaltiel and Silbak [SS23] could be improved.

More specifically, Shaltiel and Silbak [SS23] defined a notion called "HTS" which is a function that is "hard to sample on distributions with sufficient min-entropy" (a precise definition of an HTS is given in Definition 5.1). Loosely speaking, an HTS is a function f, such that for every size n^c circuit A that samples some distribution over pairs (X, Y), there exists a small set H of inputs, such that the probability that $Y = f(X)$ and $X \notin H$, is small. This notion is similar in spirit to the aforementioned notion of small set non-malleable codes: A *nonuniform* poly-size sampling circuit A may be hardwired with a polynomial size set H of inputs x, together with their output $f(x)$. This enables such a circuit to sample a distribution $(X, f(X))$ where X has small support. Intuitively, there is no markedly better approach to sample a distribution of the form $(X, f(X)$, that to implement this behavior, and a poly-size circuit cannot sample a distribution $(X, f(X))$ where X has min-entropy significantly larger than $\log |H|$.

Shaltiel and Silbak [SS23] gave explicit constructions of HTS (under the assumption that E is hard for exponential size nondeterministic circuits). (In fact, the aforementioned construction of small set non-malleable codes of [SS23] is achieved by composing the non-malleable codes of [BDL22] with a suitable HTS). The HTS constructions of [SS23] rely on components from pseudorandomness, and specifically on "high error seeded dispersers" [Zuc07]. The min-entropy thresholdof known explicit constructions of such dispersers [BKS+10, Zuc07] are quantitatively inferior compared to the parameters that can be achieved by a probabilistic argument, and this makes some of the HTS constructions of [SS23] have sets size that is small, but still larger than a polynomial.

Our next result states that if the bound on the size of small sets in the HTS construction of [SS23] can be improved, then we can achieve statistical indistinguishability in our Theorem 1.2.

Theorem 1.3 (Statistical indistinguishability assuming improved parameters for HTS, informal)**.** *If (under the hardness assumption) there is an explicit HTS $f : \{0,1\}^n \to \{0,1\}^{o(n)}$ against poly-size circuits, with polynomial set size, then Theorem 1.2 holds with statistical indistinguishability.*

By a standard application of the probabilistic method, a random function f, is w.h.p. an HTS with these parameters. Moreover, a future improvement in the parameters of explicit constructions of high error seeded

[6]The barriers of [BDL22] generalize barriers on black-box constructions of negligible error Nisan-Wigderson style PRGs from nondeterministic hardness assumptions [GSV18, AASY15]. As our "rate improvement" relies on Nisan-Wigderson style PRGs (see Section 1.3 for an overview) these barriers also apply on our rate improvement technique.

dispersers will give an HTS with parameters that are sufficient to apply Theorem 1.3 and achieve statistical indistinguishability in Theorem 1.2.

We give a formal statement of Theorem 1.3 and explain this direction, in Section 5. In Section 1.3.6 we explain how an HTS with improved quantitative parameters can be used to convert our non-malleable codes into ones with a statistical security guarantee.

1.2.1 Best of both worlds: Composing non-malleable codes with codes that correct from errors

It is natural to try and combine non-malleable codes with codes that correct from errors, aiming to get the best of both worlds. More specifically, given a parameter $0 \leq p < \frac{1}{4}$ (measuring the specified fraction of errors we need to recover from) we would like to obtain codes with the following properties:

- Recovery from a p-fraction of errors. If the channel does not alter more than a p-fraction of the encoded message, then decoding produces the original message.
- Non-malleability. If the channel alters more than a p-fraction of the encoded message, then the decoded message is either the original or unrelated to the original message.

A natural way to construct such codes is by composition. Namely, to encode a message m, we first encode it by a non-malleable code $\mathrm{Enc}_{\mathrm{nm}}$, and then encode $\mathrm{Enc}_{\mathrm{nm}}(m)$, by a code Enc_p that is designed to recover from a p-fraction of errors. In order to argue that this composition is non-malleable, we need that in addition to the tampering function, the encoding and decoding algorithms $(\mathrm{Enc}_p, \mathrm{Dec}_p)$ are also in the class $\mathcal{C}$. We are considering the case where $\mathcal{C}$ is the class of poly-size circuits, and so, can use any *explicit* construction of codes that are designed to recover from p errors.

When composing two codes $\mathrm{Enc}_{nm}, \mathrm{Enc}_p$, the rate of the composed code is the product of the two rates. As by Theorem 1.2 we now have non-malleable codes with rate $1 - o(1)$, applying this composition, we can get codes that are both non-malleable and recover from a p-fraction of errors, at the *same rate* of the best explicit codes that recover from a p-fraction of errors. In particular, we get the following corollary (which is stated more formally in Corollary 4.3.

Corollary 1.4 (Codes that are best of both worlds, informal)**.** *Let $R_{\mathrm{explicit}}(p)$ denote the best rate for which there are explicit codes that recover from a p fraction of errors. If E is hard for exponential size nondeterministic circuits, then for every $0 \leq p < \frac{1}{4}$, and every constant c, there is a non-malleable code $(\mathrm{Enc}, \mathrm{Dec})$ against circuits of size n^c, with rate R approaching $R_{\mathrm{explicit}}(p)$, that recovers from a p-fraction of errors. Furthermore, Enc and Dec can be computed in time $\mathrm{poly}(n^c)$.*

Let $R_{\mathrm{best}}(p)$ denote the best rate for which there are codes that recover from a p-fraction of errors (without the explicitness requirement). Obviously, $R_{\mathrm{explicit}}(p) \leq R_{\mathrm{best}(p)}$. Determining these rates is a notoriously difficult longstanding major open problem of coding theory. It is known that $1 - H(2p) \leq R_{\mathrm{best}}(p) < 1 - H(p)$. (The left inequality is the Gilbert-Varshamov bound, and the right inequality is a consequence of the Elias-Bassalygo bound). The best known explicit codes are inferior to the Gilbert-Varshamov bound, and have $R_{\mathrm{explicit}}(p) < 1 - H(2p)$. (Recently, there has been progress on explicit codes with rate that is close to the Gilbert-Varshamov bound for p approaching $\frac{1}{4}$ [TS17, JST21, BD22]).

Codes against poly-size circuits with rate that is superior to that of coding theory. In our scenario, we are already assuming that channels are poly-size circuits, and that encoding maps are randomized. For this scenario, there are recent explicit constructions by Shaltiel and Silbak [SS23] of codes that recover from a p-fraction of errors (w.h.p) and have rate $R(p) = 1 - H(p)$. That is, these codes have rate $R(p)$ that is

superior to $R_{\text{best}}(p)$ (let alone $R_{\text{explicit}}(p)$). These constructions rely on the same hardness assumption used in Theorem 1.2, and by applying composition with these codes, we obtain the following corollary (which is stated more formally in Corollary 4.4).

Corollary 1.5 (Codes that are best of both worlds, with rate $R = 1 - H(p)$, informal)**.** *If E is hard for exponential size nondeterministic circuits, then for every $0 \leq p < \frac{1}{4}$, and every constant c, there is a non-malleable code $(\mathrm{Enc}, \mathrm{Dec})$ against circuits of size n^c, with rate R approaching $1-H(p)$, such that for every message $m \in \{0,1\}^{Rn}$, and every size n^c circuit C that alters at most a p-fraction of the bits of $\mathrm{Enc}(m)$, with high probability over the randomness of Enc it holds that $\mathrm{Dec}(\mathrm{Enc}(m)) = m$. Furthermore, Enc and Dec can be computed in time $\mathrm{poly}(n^c)$.*

It is instructive to compare Corollary 1.4 to Corollary 1.5. As we have explained above, the latter achieves codes with rate $1 - H(p)$ that is superior to $R_{\text{best}}(p)$, and thus also to $R_{\text{explicit}}(p)$ (that is achieved by the former). In fact, the rate of $R(p) = 1 - H(p)$ achieved in [SS23], and inherited in Corollary 1.5, matches the capacity of codes for Shannon's binary symmetric channel.

However, the former gives a stronger guarantee of recovering from a p-fraction of errors in the sense that decoding is guaranteed to produce the original message with probability one, whenever the channel does not alter more than a p-fraction of the encoded message (and this holds also for tampering functions that are not computationally bounded). In contrast, Corollary 1.5 achieves a weaker guarantee, correct decoding is only guaranteed with high probability (say probability $1 - \frac{1}{n^c}$), and only in the case that the tampering function is a size n^c circuit). These weaker decoding guarantees are inherited from the codes of Shaltiel and Silbak [SS23], and are in some sense unavoidable, see [SS23] for a discussion.

Perspective. It is our view that as the scenario of non-malleable codes already assumes that tampering functions are computationally bounded, and already incorporates an error parameter in the security definition, the weaker decoding guarantee of Corollary 1.5 (which only holds against channels that are poly-size circuits) is quite natural in this scenario, and allows achieving rate that is superior to the best possible rate of standard error correcting codes.

1.3 Overview of the technique

In this section we give an overview of the main ideas that we use. For this purpose we will allow ourselves to be informal, and not entirely precise. The later technical sections do not build on the content of this section, and the reader can skip to the technical section if they wish.

1.3.1 Improving the rate of non-malleable codes for poly-size circuits

As explained in the previous section, in order to prove Theorem 1.2 we will try to take a non-malleable code Enc' that does not have good rate, and "compile" it into new non-malleable code Enc with rate $1-o(1)$. Our initial plan is to apply this transformation on the non-malleable codes of [BDL22], but as we will explain later in Section 1.3.3, it will turn out that we need Enc' to have a different variant of non-malleability.

A naive attempt using PRGs Let us start with the following naive attempt (that we will refine later on), which will nonetheless be instructive. In order to construct a non-malleable code Enc with block length n and message length k, we will take $k' \ll k$, and a PRG $G : \{0,1\}^{k'} \to \{0,1\}^k$. We will also require and a non-malleable code $\mathrm{Enc}' : \{0,1\}^{k'} \to \{0,1\}^{n'}$ (that may have bad rate, say $n' = \mathrm{poly}(k')$). By choosing

the stretch of G to be sufficiently large, we can arrange that $n' = o(k)$. In order to encode a message $m \in \{0,1\}^k$, we will choose a uniform seed $S \leftarrow \{0,1\}^{k'}$ and define:

$$\mathrm{Enc}(m) = m \oplus G(S), \mathrm{Enc}'(S).$$

That is, we mask m with $G(S)$ (here, G will be a PRG that fools the tampering function C, which is a poly-size circuit) and append $\mathrm{Enc}'(S)$.

The decoding algorithm Dec (of the constructed code) will apply the decoding algorithm Dec' (of the initial code) on the second block, to obtain some seed $\bar{S}$, and complete the decoding, by computing $G(\bar{S})$, and xoring it with the first block to produce the decoded message $\bar{m}$. More formally:

$$\bar{m} = \mathrm{Dec}(V1, V2) = V_1 \oplus G(\bar{S}) = V_1 \oplus G(\mathrm{Dec}'(V_2)).$$

The advantage of this naive approach is that the rate of the constructed code Enc is indeed

$$\frac{k}{n} = \frac{k}{k+n'} = \frac{k}{k+o(k)} = 1 + o(1).$$

A simple attack that we ignore for now. Note that a poly-size circuit $C : \{0,1\}^n \to \{0,1\}^n$ can easily attack this construction by simply ignoring the second block, and xoring the first block with some fixed value $v \in \{0,1\}^k$. As C does not alter the second block, the decoding $\mathrm{Dec}'(\mathrm{Enc}'(S))$ will produce $\bar{S} = S$, which gives that:

$$\bar{m} = \mathrm{Dec}(C(\mathrm{Enc}(m)) = ((m \oplus G(S)) \oplus v) \oplus G(\bar{S}) = m \oplus G(S) \oplus G(S) \oplus v = m \oplus v,$$

This attack completely breaks non-malleability as $\bar{m} = m \oplus v$ is related to m. We ignore this attack for now. This is because using a MAC, it is not difficult to modify the construction to handle this simple attack (which does not alter the second block). We will deal with this concern later in Section 1.3.4. It is more problematic to handle attacks that alter the second block, and handling these kinds of attacks is the main problem that we contend with in this paper.

How do we plan to use the non-malleability of Enc'? Let us focus on attacks that only modify the second block. When preparing the second block, we have encoded S with the non-malleable code Enc'. It is instructive to ask what would happen if when preparing the second block, we did not encode S, and instead sent S "in the clear". This would have allowed C to replace S with some $\psi(S)$ (for a function ψ that is computable by a small circuit). Note that we have no non-malleability guarantee for the PRG G, and as far as we know, it may be the case that there exists a fixed $v \in \{0,1\}^k$, such that for every $S \in \{0,1\}^{k'}$, $G(S) \oplus G(\psi(S)) = v$.[7]

If this happens then a tampering circuit C that by ignores the first block, and applying ψ on the second block, leads the decoding algorithm Dec to decode to

$$\bar{m} = m \oplus G(S) \oplus G(\psi(S)) = m \oplus v,$$

and once again we have that $\bar{m} = m \oplus v$ and this attack completely break non-malleability. We will refer to this attack as *the ψ-attack*.

[7] It is easy to construct a PRG G with this property, given an arbitrary PRG G'. More specifically, given a PRG G', we construct a PRG G where the seed of G has one additional bit. If this bit is zero, we output the output of G on the first part of the seed, and if not, we xor the output of G with the fixed string v. It is easy to check that this is a PRG that satisfies $G(S) \oplus G(\psi(S)) = v$, for the function ψ that flips the additional bit of the seed S.

In the ψ-attack, C was able to arrange that the "decoded seed" $\bar{S}$ is $\psi(S)$ that is *correlated* with S. Intuitively, encoding S by the non-malleable code Enc' is supposed to guarantee that the decoded seed $\bar{S}$ is *not correlated* with S, and rule out the ψ-attack. More formally, we will want to argue that having encoded S with a non-malleable code we have that if $\bar{S} \neq S$ then:

1. $\bar{S}$ is independent of S, which will be used to argue that:
2. $G(S) \oplus G(\bar{S})$ is pseudorandom (as xoring the pseudorandom string $G(S)$ with the independent string $G(\bar{S})$ yields a pseudorandom string).

If we can do this, this will mean that an attack that only modifies the second block, leads the decoding to decode to:

$$\bar{m} = (m \oplus G(S)) \oplus G(\bar{S}) = m \oplus (G(S) \oplus G(\bar{S})),$$

which is a pseudorandom distribution that "masks out" m. This intuitively implies that $\bar{m}$ is unrelated to m (as required).

Difficulties in implementing the plan above. In the intuition above, we did not take into account that when the tampering circuit C tampers the second block of $\mathrm{Enc}(m)$ (which is the string $\mathrm{Enc}'(m)$) it also receives the first block of $\mathrm{Enc}(m)$ (which is the string $m \oplus G(S)$). This in particular means that when C tries to break the non-malleability of Enc', it receives additional information about S that the definition of non-malleability does not account for.

As far as we know, it may be the case that G and Enc' are related in a way that makes it possible for a size n^c circuit C, that sees both $m \oplus G(S)$ and $\mathrm{Enc}'(S)$ to lead the decoding Dec' to decode to $\bar{S} = \psi(S)$ which *is* correlated with S.[8] Even worse, such a C can break non-malleability of Enc by implementing the ψ-attack described above.

Summing up, it seems that encoding S by a non-malleable code Enc', does not have the desired effect, and does not rule out the ψ-attack.

1.3.2 Using seed-extending PRGs

Can we argue that seeing the first block (that is seeing $m \oplus G(S)$) (in addition to the second block $\mathrm{Enc}(S)$) does not help a circuit C to lead Dec' to decode to a string $\bar{S}$ that is correlated with S?

It turns out that we can achieve such behavior if we require that the PRG G is *seed-extending*. A seed extending PRG is a PRG G that remains secure even when its seed is revealed to the distinguisher. More formally, (see Definition 2.16) if the function $G'(x) = G(x), x$ is also a PRG. Before explaining why this is the case, let us discuss the notion of seed-extending PRGs.

Seed-extending PRGs. Seed-extending PRGs exist only in a scenario where the intended class of distinghuishers cannot run the PRG. In particular, cryptographic PRGs (which fool distinguishers that can run the PRG) cannot be seed-extending. In contrast, known constructions of Nisan-Wigderson style PRGs [NW94, IW97, SU05] are seed-extending, and these can be instantiated using the hardness assumption that E is hard for exponential size deterministic circuits (that we are already assuming).[9]

[8] For example, it does not seem that we can rule out the case that a prefix of $G(S)$ is a string $\mathrm{Enc}'(\psi(S))$ where ψ is a OWF. In such a case (and assume for simplicity that m starts with a prefix of zeros) a small circuit C can lead the Dec' to output $\psi(S)$, by simply replacing the second block with a prefix of the first block.

[9] Another drawback of seed-extending PRGs, is that their error is not negligible, but rather an arbitrary fixed inverse polynomial. It is known that black-box proofs cannot achieve such PRGs that run in poly-time and have negligible error, under standard hardness

Why seed extending PRGs help. We now explain how seed-extending PRGs bypass the problem above. This argument is inspired by a related approach that was used by Shaltiel and Silbak [SS23] in their construction of codes that recover from errors induced by poly-size channels.

We will require that in addition to the circuit C, the seed-extending PRG G, also fools $\mathrm{Enc}', \mathrm{Dec}'$ (and this can be achieved as $\mathrm{Enc}', \mathrm{Dec}'$ indeed run in fixed polynomial time). For a seed extending PRG, we have that:

$$(G(S), S) \equiv_c (U_k, S).$$

Using the fact that G fools Enc', we conclude that:

$$(G(S), \mathrm{Enc}'(S)) \equiv_c (U_k, \mathrm{Enc}'(S)).$$

As we have required that G also fools C and Dec', this means that even after applying the tampering function C, and then applying Dec' on the second block, to obtain $\bar{S}$, we have that the triplet $(G(S), S, \bar{S})$ (obtained in the left hand side) is computationally indistinguishable from the triplet $(G(S), S, \tilde{S})$ (obtained in the right hand side). (Here, $\tilde{S}$ is the string obtained when applying Dec' in the experiment in the right hand side, where $G(S)$ is replaced with U_k).

In the right hand side we have that S and $\tilde{S}$ are independent. This is because in the experiment in the right hand side, the additional string that C sees is U_k (rather than $G(S)$), and as C could have sampled this string on its own, it does not help C to break the non-malleability of Enc'. We can therefore use the non-malleability of Enc', and conclude that S and $\tilde{S}$ are independent.

From the indistinguishability of the left hand side and the right hand side, we can now conclude that S and $\bar{S}$ are computationally indistinguishable from being independent.

Summing this discussion, using seed-extending PRGs, we are able to obtain a weakened version of item (1) in our plan above. While we can't show that if $\bar{S} \neq S$, then $\bar{S}$ is *statistically independent* from S (as in the original item (1)), we can show that if $\bar{S} \neq S$, then $\bar{S}$ and S are *computationally indistinguishable* from being independent.

Unfortunately, this weaker conclusion is not sufficient to imply item (2) in our plan above. Specifically, we cannot argue that $G(S) \oplus G(\bar{S})$ is pseudorandom. This is because such a conclusion requires the distinguisher to the PRG G, to compute $G(\bar{S})$ given $\bar{S}$, and as we have already observed, because G is seed-extending, distinguishers for G cannot run G.

While this doesn't succeed, we note that we would have been able to argue that $G(S) \oplus G(\bar{S})$ is pseudorandom, if we could guarantee that $\bar{S}$ is either S or a *fixed* string $\bar{s}$. In that case, the fixed string $\bar{s}$ (as well as the fixed string $G(\bar{s})$) could be hardwired to the distinguisher, and we would be able to argue that $G(S) \oplus G(\bar{s})$ is pseudorandom.

1.3.3 Replacing non-malleable codes with small set non-malleable codes

Fortunately, the notion of "small-set non-malleable codes" (abbreviated as SS-non-malleable codes) described in the introduction, essentially allows us to assume that $\bar{S}$ is a fixed string, and implement our plan.

More specifically, if we take Enc' to be an SS-non-malleable code, then the security guarantee of SS-non-malleable codes (see Definition 2.8 for a precise formulation) gives that for every tampering circuit C,

assumptions [GSV18, AASY15]. There are also extensions of these negative results, that rule out certain black-box constructions of non-malleable codes with negligible error, from the assumption that E is hard for exponential size nondeterministic circuits [BDL22]. This is (one of the) reasons why [BDL22] (as well as our Theorem 1.2) do not obtain non-malleable code with negligible distinguishing advantage.

there exists a small set $H \subseteq \{0,1\}^{k'}$, such that it is unlikely that C will lead the Dec' to decode to a string $\bar{S}$ that is neither in H, nor the original seed S.

For the sake of intuition, note that if H was of size one, this would exactly say that $\bar{S}$ is either S or a fixed string. While we cannot expect H to be of size one, the SS-non-malleable codes of [SS23] achieve H with a polynomial size. By setting G to fool circuits that are sufficiently larger than the size of H, we can allow a distinguisher for G to be hardwired with all the pairs $(s, G(s))$ for $s \in H$. This suffices to implement the plan outlined above, and show that in our construction, either $\bar{S} = S$, or $G(S) \oplus G(\bar{S})$ is pseudorandom.

Summing up, by refining our naive attempt, taking G to be a seed-extending PRG, and Enc' to be an SS-non-malleable code, we are able to circumvent the ψ-attack.

1.3.4 The final construction of Enc

We now turn our attention to the simple attack (that we outlined in Section 1.3.1) and modify the construction of Enc so that it also bypasses the simple attack. This will be done as follows:

In addition to the PRG G and the SS-non-malleable code Enc', we will also use a pairwise independent family of hash functions mapping n bits to k' bits. In order to encode a message $m \in \{0,1\}^k$, we will choose a two uniform seeds $S_{\mathrm{PRG}}, S_{\mathrm{Hash}} \leftarrow \{0,1\}^{k'}$. We define $S = (S_{\mathrm{PRG}}, S_{\mathrm{Hash}})$, $K = G(S_{\mathrm{Hash}})$, and set $T = h_K(m)$. That is, we use $K = G(S_{\mathrm{Hash}})$ as a "key" to choose a function h_K from the pairwise independent hash family, and set $T = h_K(m)$. Finally, we encode m by:

$$\mathrm{Enc}(m) = m \oplus G(S_{\mathrm{PRG}}), \mathrm{Enc}'((S,T)).$$

The decoding algorithm Dec proceeds as before, namely, Dec will apply the decoding algorithm Dec' (of the SS-non-malleable code) on the second block, to obtain some string $(\bar{S}, \bar{T})$. It will then compute $G(\bar{S}_{\mathrm{PRG}})$, and xor it with the first block to produce a candidate message $\bar{m}$. At this point, Dec will compute $\bar{K} = G(\bar{S}_{\mathrm{Hash}})$, and check whether $h_{\bar{K}}(\bar{m}) = \bar{T}$. If this is the case, Dec will output $\bar{m}$, and otherwise Dec will fail.

Circumventing the simple attack. The addition of the hash function is done in order to circumvent the simple attack described in Section 1.3.1. Recall that in that simple attack the tampering function does not alter the second block, and only alters the first block. In this refined construction, we will be able to argue that it is unlikely that Dec will produce a message $\bar{m}$ (rather than failing) on this simple attack.

Here is a rough sketch of this argument. Let C be a tampering circuit. Recall that G is a seed extending PRG that fools both C and Dec'. If C implements the simple attack, we are willing to reveal m, S and T to C (and note that this reveals all the randomness of Enc as well as the message m). However, because G is seed-extending, S_{Hash} is (computationally) independent of $K = G(S_{\mathrm{Hash}})$, which intuitively means that even after we revealed S_{Hash}, C does not have information on the key K used to choose the hash function. By a standard application of pairwise independence, this can be used to argue that C cannot find a message $\bar{m} \neq m$, on which $h_K(\bar{m}) = h_K(m)$, and so, Dec will fail w.h.p., and the simple attack cannot make Dec output a message $\bar{m}$ (let alone one that is related to m).

We remark that here, we once again benefited from the fact that G is a seed-extending PRG. Specifically, this allowed us to reveal the seed S_{Hash}, and still maintain that $K = G(S_{\mathrm{Hash}})$ is "secret".

1.3.5 A sketch of the non-malleability proof

So far, we have only explained how our construction avoids very specific attacks. In this section we will give a sketch of the non-malleability proof. The high level idea is to compare two experiments: The first

experiment is the "real experiment" in which the encoding Enc, tampering function C, and decoding Dec run as designed. The second is an "imagined experiment" in which the strings $G(S_{\mathrm{PRG}})$, and $G(S_{\mathrm{Hash}})$ are replaced by uniform strings.

A simplifying assumption. Let's assume for simplicity that the pseudorandomness of G implies that these two experiments are computationally indistinguishable. Note that this does not directly follow from the fact that G is a PRG, because Dec (which is applied in the two experiments) needs to run G as part of the decoding, and as G is seed-extending, it cannot fool adversaries that run G.

An important observation is that the imagined experiment can be carried out *without* knowing m. This follows because in this experiment, in the encoding phase, the first block of $\mathrm{Enc}(m)$ is $m \oplus U_k$ (which is independent of m). Furthermore, in the imagined experiment, the pair (S, T) is a uniformly distributed string (that is independent of m). Therefore, one can can sample it, *without* knowing m.[10]

The fact that the imagined experiment can be carried out without knowing m, allows us to get a simulator for the non-malleable code. Loosely speaking, this simulator will sample from the imagined experiment, and we will use the computational indistinguishability of the real experiment and the imagined experiment to argue the correctness of the simulator. We remark that this description hides many details.

Justifying the simplifying assumption using SS-non-malleability. While the computational indistinguishability of the real experiment and the imagined experiment does not follow from the pseudorandomness of G. We are able to show that it does follow, by also using the SS-non-malleability of Enc'. This argument is similar in flavor to the argument that we explained in Section 1.3.3.

Loosely speaking, the idea is that using SS-non-malleability (as well as the fact that G is a seed-extending PRG, and the ideas explained in Sections 1.3.2 and 1.3.3) we can show that in the imagined experiment, for every tampering function C, there exists a small set H of seeds, such that it is unlikely that C can lead Dec' to output a seed $\bar{S}$ which is neither in H, nor the original S. As in Section 1.3.3, we can set the PRG G to fool distinguishers of size slightly larger than $|H|$. This will allow us to argue that the two experiments are computationally indistinguishable.

Recall that previously, we could not argue that the real experiment and the imagined experiment are computationally indistinguishable, because in both experiments a potential distinguisher needed to compute G, and is therefore "too large" to be fooled by G.

However, as it is unlikely that an $\bar{S} \notin H$ would come up, it is now sufficient to fool a distinguisher that is hardwired with the triplet $s, G(s_{\mathrm{PRG}}), G(s_{\mathrm{Hash}})$ for every $s \in H$. (As this allows the distinguisher to run G on all inputs that are likely to come up in the experiment).

We can indeed set G so that it fools such distinguishers, and this allows us to argue that the real experiment and the imagined experiment are computationally indistinguishable.

While we are hiding some details here, this argument is similar in spirit to our explanation of how to handle the ψ-attack. In particular, it critically relies on the fact that G is seed-extending, and that Enc' is SS-non-malleable.[11]

[10] Note that here we crucially use that when preparing the key K for the hash function, we applied a seed-extending PRG G, and took $K = G(S_{\mathrm{Hash}})$ (rather than $K = S_{\mathrm{Hash}}$). In the former case, when we replace $G(S_{\mathrm{Hash}})$ with U_k, we obtain a key K that is independent of S_{Hash}, and this is why we get that (S, T) is uniformly distributed. Had we used the more natural choice of $K = S_{\mathrm{Hash}}$, then the pair $S_{\mathrm{Hash}}, h_{S_{\mathrm{Hash}}(m)}$ contains information about m, and we can't sample such a pair without knowing m.

[11] Curiously, our simulator is non-black box in the following manner: It is not sufficient for our simulator to receive black-box access to the tampering circuit C, and it also requires to be hardwired with the set H (associated with C, by SS-non-malleability) as well as the triplet $s, G(s_{\mathrm{PRG}}), G(s_{\mathrm{Hash}})$ for every $s \in H$.

1.3.6 Achieving statistical indistinguishability with a suitable HTS

The construction that we have outlined in the previous sections only achieves computational indistinguishability. In this section we explain the direction stated in Theorem 1.3 which shows that we can get a code with statistical indistinguishability if we have a suitable HTS.

More specifically, given an HTS $f : \{0,1\}^k \to \{0,1\}^{o(k)}$ (as promised in Theorem 1.3) we will modify the code $(\mathrm{Enc}, \mathrm{Dec})$ of Theorem 1.2 as follows: To encode a message m, we will apply Enc on the string $(m, f(m))$. (Note that as $|f(m)| = o(|m|)$ this only slightly reduces the rate of the initial code). Having made this change, when decoding, we expect to get a pair of the form $(X, f(X))$, and decoding will fail if this is not the case.

Recall that an HTS is a function that is hard to sample, and we plan to apply the HTS against the simulator of $(\mathrm{Enc}, \mathrm{Dec})$. The transformation described above applies to any non-malleable codes with the following property: For every tampering circuit C of size n^c, there is a simulator Sim_C that is a circuit of size $n^{c_{\mathrm{Sim}}}$ that produces a distribution D_C (as required in Definition 1.1) where computational indistinguishability holds against distinguishers that are circuits of size $n^{c'}$ (for a constant c' that is sufficiently larger than c, c_{Sim}). The code construction of Theorem 1.2 has this property.

The security definition of the HTS (when applied against circuits of size $n^{c_{\mathrm{Sim}}}$) guarantees that for every circuit C of size n^c, the HTS f is secure against Sim_C. This formally means that there exists a small set H (that depends on C) such that Sim_C is unlikely to sample a pair (X, Y) such that $X \notin H$, and $Y = f(X)$.

This implies that the ditribution D_C (sampled by Sim_C) is statistically close to having small support. We can set the parameters so that this support is much smaller than $n^{c'}$. By the definition of non-malleability, we get that D_C cannot be distinguished from $\mathrm{Tamper}_C(m)$ by circuits of size $n^{c'}$.

However, when a distribution has support size $n^{c'}$, any statistical test can be implemented by a circuit of size roughly $n^{c'}$. This means that statistical indistinguishability immediately follows from computational indistinguishability.

Shaltiel and Silbak [SS23] constructed an HTS from the hardness assumption that we are already assuming. However, their construction does not achieve set size that is sufficiently small for our purposes, for a function $f : \{0,1\}^n \to \{0,1\}^{o(n)}$.

We remark that the set size in the construction of [SS23] would be sufficiently small for our purposes, if the parameters of explicit constructions of high-error seeded dispersers are improved. See [SS23] for details.

1.4 Other Rate Compilers for Non-Malleable Codes

Our work is not the first to consider rate optimizing compilers for non-malleable codes. All of these compilers, including the one in this paper, share a similar structure that mimics "hybrid" encryption paradigm: in each compiler some form of a low rate non-malleable code (NMC) is combined with some form of symmetric key "authenticated encryption." That said, despite the high level similarity the specific instantiations of each primitive vary widely, and each setting requires overcoming very different analytical challenges.

The rate compiler in this paper instantiates high-rate "authenticated encryption" by hiding the message with a (non-cryptographic) seed-extending PRG and applying a information-theoretic MAC with a pseudorandom key. Moreover, low-rate the non-malleable code used is not an NMC for polynomial tampering, but instead a small-set NMC. As seen in Section 1.3, these choices were made to deal with challenges specific to the setting of tampering by polynomial-size circuits.

In contrast to this work, all other compilers are for settings where the tampering must obey some kind of locality constraint: the tampering function corresponding to any particular tampered output bit only has access to partial information about the original codeword.

Local tampering. Perhaps the first rate compiler was introduced by [AGM+15] who showed how to compile a polynomial rate non-malleable code for the family of 1-local tampering functions (where each output bit depends on at most one input bit) to a rate 1 non-malleable code for the same class. In this compiler "authenticated encryption" is instantiated by encoding the message in a high rate error-correcting threshold secret-sharing scheme, flipping a few bits, and writing down the indices changed. This scheme is obviously highly tailored to this very restricted tampering setting, yet nonetheless the analysis is quite delicate.

Later [GMW19] techniques from the compiler of [AGM+15] as well as techniques used by [BDKM16] to construct non-malleable codes for local tampering to achieve rate 1 non-malleable codes for $c \lg n$-local functions where $c < 1$ (i.e. tampering functions where each bit depends on the output). This work did not compile non-malleable codes for local functions directly, but instead compiled codes for a tampering class (called "leaky input-output local" tampering) which was a class considered in [BDKM16] in order to build NMC for local functions.

Computationally bounded split-state tampering. After [AGM+15]'s initial compiler, [AAG+16] showed how to compile a polynomial rate NMC for split-state tampering (which remains secure special leakage resilient properties, a so-called *augmented* split-state NMC) to a rate 1 NMC for computational split-state tampering. Computational split-state tampering is split-state tampering (where the left half of the codeword must be tampered independently of the right half) which is additionally restricted to be computable in polynomial time.[12] Note that without this additional computational restriction rate 1/2 is the best possible [CG16].

Their compiler simply encoded the key to a high rate authenticated encryption scheme with using a split-state NMC, then the ciphertext is simply appended to one of the NMC codeword halves. To handle the fact that the tampering function can jointly tamper the ciphertext text (which depends on the secret key) with half of the codeword, augmented non-malleability was introduced.

Split-state tampering. A sequence of works [KOS17, KOS18, GMW17, AO20, AKO+22] used rate compilers to improve the rate of NMCs for split-state tampering and t-state tampering (where the codeword is broken into $t > 2$ blocks which are tampered independently).

[KOS17] showed how to compile a low rate split-state NMC to a rate 1/3 4-state NMC. [KOS18, GMW17] later showed reduce the states in the resulting code to 3. Finally, [AKO+22] showed how to reduce the states in the resulting code to just 2, yielding a rate 1/3 split-state NMC, which is currently the best known rate of any explicit NMC in this setting. There are slight differences between these compilers but they largely follow the template established in [KOS17]: a seeded extractor is used to build a high rate leakage-resilient encoding scheme which is then tagged using an information theoretic MAC, then the (short) seed for the leakage resilient encoding scheme and the key for the information theoretic MAC are encoded using a low-rate split-state NMC. The analysis critically relies on the independence between the states (making the template work for just 2 tampering states is particularly subtle).

1.5 Organization of this paper

In Section 2 we give preliminaries, provide formal definitions, and the statements of earlier work that we use. In Section 3 we formally restate Theorem 1.2 and present our construction and analysis of our non-malleable codes of Theorem 1.2. In Section 4 we formally restate Corollaries 1.4 and 1.5 and present our

[12] In contrast, to the tampering model considered in this work the polynomial time bound need not be fixed a priori. However, the model considered here each tampered codeword bit may depend on all the bits of the original codeword.

results on codes that achieve the best of both worlds. In Section 5 we formally restate Theorem 1.3 and show how to use an improved HTS to achieve non-malleable codes with statistical security.

2 Preliminaries and ingredients

In this section we give formal definitions of the notions and ingredients used in the construction. We also cite previous results from coding theory and pseudorandomness that are used in the construction.

General notation. We use $[n]$ to denote $\{1, \ldots, n\}$.

Probability distributions. We use U_n to define the uniform distribution on n bits. The ***statistical distance*** between two distributions P, Q over Ω is $\Delta(P, Q) = \max_{A \subseteq \Omega} |P(A) - Q(A)|$. Given a distribution P, we use $X \leftarrow P$ to denote the experiment in which the random variable X is chosen according to P. For a set A, we use $X \leftarrow A$ to denote the experiment in which X is chosen uniformly from A. We use $X_1, \ldots, X_n \leftarrow A$ to denote the experiment in which n variables are chosen uniformly from A with replacement. Given two random variables X and Y distributed over the same support we use $X \approx^s_\epsilon Y$ to denote that X and Y are sampled from distributions that have statistical distance at most ϵ. X and Y are said to be ϵ-indistinguishable for the class of functions $\mathcal{C}$, denoted as $X \approx^{\mathcal{C}}_\epsilon Y$, if for every $D \in \mathcal{C}$ it holds that $|\Pr[D(X) = 1] - \Pr[D(Y) = 1]| \leq \epsilon$.

Hamming distance. The *Hamming distance* between $x, y \in [q]^n$ is $\Delta(x, y) = |\{i : x_i \neq y_i\}|$. The *relative Hamming distance* between $x, y \in [q]^n$ is $\delta(x, y) = \frac{\Delta(x,y)}{n}$.

2.1 Circuits and hardness assumptions

The assumption that E is hard for exponential size circuits of a certain type is formally stated below.

Assumption 2.1 (E is hard for exponential size circuits)**.** *We say that "E is hard for exponential size circuits of type X" if there exists a language L in $E = DTIME(2^{O(n)})$ and a constant $\beta > 0$, such that for every sufficiently large n, the characteristic function of L on inputs of length n cannot be computed by circuits of size $2^{\beta n}$ of type X.*

This assumption was used by Impagliazzo and Wigderson [IW97] with hardness against deterministic circuits to imply that BPP = P. Later work in complexity theoey and cryptography [KvM02, MV05, TV00, SU05, BOV07, GW02, GST03, SU06, SU09, Dru13, AASY15, BV17, AIKS16, HNY17, DMOZ22, BDL22, CT22] used the assumption that E is hard for nondeterministic circuits for various conclusions. In this paper, we also rely on this assumption. For completeness, we give a formal definition of nondeterministic circuits below.

Definition 2.2 (randomized circuits and nondeterministic circuits)**.** *A* randomized *circuit C has additional wires that are instantiated with uniform and independent bits. A* nondeterministic *circuit C has additional "nondeterministic input wires". We say that the circuit C evaluates to 1 on x iff there exist an assignment to the nondeterministic input wires that makes C output 1 on x.*

2.2 Coding schemes: non-malleable codes, SS-non-malleable codes and error correcting codes.

In this section, we define the various notions of codes that come up in this paper. We start from the following baseline notion of coding schemes, which requires that the original message is recovered if no errors are introduced. The later definitions of various notion of codes will all have this baseline property.

Definition 2.3. *A pair of functions* $(\mathrm{Enc}, \mathrm{Dec})$*, where* $\mathrm{Enc} : \{0,1\}^k \to \{0,1\}^n$ *is a randomized function and* $\mathrm{Dec} : \{0,1\}^n \to \{0,1\}^k \cup \{fail\}$ *is a deterministic function is defined to be a* coding scheme *with block length* n *and message length* k*, if for every* $m \in \{0,1\}^k$*,* $\Pr[\mathrm{Dec}(\mathrm{Enc}(m)) = m] = 1$*. We say that a coding scheme is* deterministic *if* Enc *is a deterministic function. The* rate *of a coding scheme is* $\frac{k}{n}$*. A coding scheme is* explicit *if* Enc *and* Dec *run in polynomial time. (Naturally, this makes sense only for a family of encoding and decoding functions with varying block length* n*, and message length* $k(n)$*.*

We will often be interested in coding schemes that work against a specified family of channels $C : \{0,1\}^n \to \{0,1\}^n$ (which will often be allowed to be randomized).

Definition 2.4 (Tampering Channels)**.** *For an integer* n*, we define the following.*

- $\mathcal{F}_n$ *is the class of all function* $C : \{0,1\}^n \to \{0,1\}^n$*.*
- $\mathcal{F}_n^{\mathrm{rand}}$ *is the class of all randomized functions* $C : \{0,1\}^n \to \{0,1\}^n$*.*

In this paper we will be interested in the class of (deterministic or randomized) circuits of a fixed polynomial size.

2.2.1 Non-malleable codes

Non-malleable codes are coding schemes that provide some decoding guarantee, even against powerful adversaries that can completely modify the codeword. The definition below (which extends the informal definition given in the introduction) is due to Dziembowski, Pietrzak, and Wichs [DPW18].

Definition 2.5 (The function Copy)**.** *The function* Copy *is defined as follows:*

$$\mathrm{Copy}(x, y) = \begin{cases} x, & x \neq \text{same} \\ y, & x = \text{same} \end{cases}$$

Definition 2.6 (non-malleable codes [DPW18])**.** *A coding scheme* $(\mathrm{Enc}, \mathrm{Dec})$ *with block length* n *and message length* k *is said to be* ϵ-non-malleable *for a class* $\mathcal{C} \subseteq \mathcal{F}_n^{\mathrm{rand}}$*, if for every* $C \in \mathcal{C}$*, there exists a distribution* D_C *that is supported on* $\{0,1\}^k \cup \{\text{same}\}$*, and for every* $m \in \{0,1\}^k$*,*

$$\mathrm{Copy}(D_C, m) \approx_\epsilon^s \mathrm{Dec}(C(\mathrm{Enc}(m))).$$

Let $\mathcal{C}'$ *be a class of functions. The coding scheme is only* $\mathcal{C}'$-computationally ϵ-non-malleable *for the class* $\mathcal{C}$*, if*

$$\mathrm{Copy}(D_C, m) \approx_\epsilon^{\mathcal{C}'} \mathrm{Dec}(C(\mathrm{Enc}(m))).$$

(That is if statistical distance, is replaced by indistinguishability for distinguishers in $\mathcal{C}'$*).*

We say that the codong scheme is simulatable *by a size* s*-circuits, if for every* $C \in \mathcal{C}$*, there exist a size* s *randomized circuit* Sim_C *that samples* D_C*.*

In this paper, we focus on the case where both $\mathcal{C}$ and $\mathcal{C}'$ is the class of circuits of fixed polynomial size.

Remark 2.7 (Non-malleability for randomized circuits, and black-box simulation)**.** *We will often use non-malleability against the class $\mathcal{C}$ of* randomized *circuits of a certain size. It should be noted that when one is only interrested in non-malleability, and does not care whether the code is simulatable by a small circuit, non-malleability against (deterministic) circuits immediately implies non-malleability against randomized circuits.*

Moreover, efficient simulation against randomized circuits immediately follows from efficient simulation against deterministic circuits, in the case that the simulator is black-box. More precisely, if for every deterministic C. the simulator Sim_C can be implemneted using only oracle access to C (that is, if $\mathrm{Sim}_C = \mathrm{Sim}^C$) then, as the simulator works for every choice of random coins of a randomized circuit, it also simulates against the randomized circuit.

However, the simulators that we construct in this paper do not have this property. They are non-black-box in the sense that in addition to oracle access to C, the simulator Sim_C also needs to be hardwired with a polynomial length nonnuniform string (that depend on C).

In such a scenario, simulation against deterministic circuits does not immediately imply simulation against randomized circuits. Nevertheless, our techniques do yield simulators against randomized circuits, and this is why we are careful in Definition 2.6 (as well as the rest of the paper) about the distinction between deterministic and randomized circuits.

2.2.2 Small set non-malleable codes

Shaltiel and Silbak [SS22, SS23] defined a notion called "small set non-malleable codes" (which is closely related to a notion of "bounded non-malleable codes" introduced by Faust et al. [FMVW16]). In this paper, we do not require the full power of the definition made in [SS22, SS23], and define a weaker object (while still referring to it as an SS-non-malleable code). The reader is referred to [SS22, SS23] for the definition of the stronger object, and to [SS22] for a discussion on how SS-non-malleable codes capture a different intuition than the standard notion of non-malleability, and why it seems that small set non-malleability is incomparable to standard non-malleability.

Loosely speaking, the definition below requires that for every tampering function C, there exists a small set H of messages, such that when C gets to corrupt the encoding of a uniformly chosen message X, it is unlikely that the decoding will produce a message that is neither in H, nor the original message X.

Definition 2.8 (SS-non-malleable codes [SS22, FMVW16])**.** *A coding scheme $(\mathrm{Enc}, \mathrm{Dec})$ with block length n and message length k, is (h, ρ)-*SS-non-malleable *for a class $\mathcal{C} \subseteq \mathcal{F}_n^{\mathrm{rand}}$, if for every $C \in \mathcal{C}$, there exists a set $H_C \subseteq \{0,1\}^k$, with $|H_C| \leq h$, such that:*

$$\Pr_{X \leftarrow \{0,1\}^k}[\mathrm{Dec}\left(C(\mathrm{Enc}(X))\right)) \notin H_C \cup \{X\} \cup \{fail\}] \leq \rho.$$

Shaltiel and Silbak [SS23] gave the following construction of SS-non-malleable codes. Once again, we remark that the construction of [SS23] achieves a stronger object, and that the statement we provide here is weaker, and yet, sufficient for our purposes.

Theorem 2.9 (Construction of SS-non-malleable codes construction, [SS23])**.** *There exists a constant $c_0 > 1$ such that if E is hard for exponential size nondeterministic circuits then for every constant $c_{\mathrm{ssnm}} > 1$, there exist constants $t_{\mathrm{ssnm}} > c_{\mathrm{ssnm}} \geq 1$ and constants $a'_{\mathrm{ssnm}} \geq 1$ such that for every constant $a_{\mathrm{ssnm}} \geq a'_{\mathrm{ssnm}}$ and for every sufficiently large n, setting $k = a_{\mathrm{ssnm}} \cdot \log n$ and $k' = k^{c_0}$, there is a coding scheme (Enc, Dec)*

with message length k and block length k' that is a $(n^{a'_{\text{ssnm}}}, \frac{1}{n^{c_{\text{ssnm}}}})$-SS-non-malleable codes for randomized circuits of size $n^{c_{\text{ssnm}}}$. Furthermore, Enc, Dec *can be computed in time $n^{t_{\text{ssnm}}}$.*

Remark 2.10. *We remark that an SS-non-malleable code against deterministic circuits, does not imply SS-non-malleability against randomized circuits. This is because, while the former does imply that for every fixing of random coins, there exists a suitable small set H, these sets may be different for different choices of random coins. Theorem 2.9 is stated for randomized circuits, and gives the stronger guarantee, that for every* randomized *circuit of size n^c, there exists a suitable small set H.*

2.2.3 Standard error correcting codes: decoding from errors

We now define the standard notion of error correcting codes used in coding theory. Such codes are usually defined by requiring a minimal distance property, namely, that for every two messages $m_0, m_1 \in \{0,1\}^k$, the Hamming distance between $\text{Enc}(m_0)$ and $\text{Enc}(m_1)$ is large. In this paper it will be more convenient to use the notation of coding schemes. (In contrast to non-malleable codes, here, coding schemes can be deterministic).

Definition 2.11 (Decoding from errors)**.** *A deterministic coding scheme* (Enc, Dec) *with block length n and message length k,* decodes *from a p fraction of errors if for every $m \in \{0,1\}^k$ and every $v \in \{0,1\}^n$ with $\delta(\text{Enc}(m), v) \leq p$, $\text{Dec}(v) = m$.*

It is standard that a deterministic coding scheme is decodable from a p fraction of errors if and only if the Hamming distance between every two codewords is at least $2pn + 1$. It is also known that for $p \geq \frac{1}{4}$, coding schemes that decode from a p fraction of errors must have vanishing rate, and this is why we restrict our attention to $p < \frac{1}{4}$ in this paper.

2.2.4 Decoding from errors induced by a specified class of channels

The standard scenario considered in error-correcting codes (described in the earlier section) assumes that there is an upper bound p on the fraction of errors introduced by the channel, but places no computational restriction on the channel. Lipton [Lip94] suggested to consider the case where channels are computationally bounded (in addition to the upper bound on the fraction of errors that they induce). The definitions below which considers such a situation was given by Guruswami and Smith [GS16].

Definition 2.12 (Decoding from errors introduced by a class of channels)**.** *A coding scheme* (Enc, Dec) *with block length n and message length k, is* decodable *for the class $\mathcal{C} \subseteq \mathcal{F}_n^{\text{rand}}$, with success probability $1 - \nu$, if for every $m \in \{0,1\}^k$ and every $C \in \mathcal{C}$, $\Pr[\text{Dec}(C(\text{Enc}(m))) = m] \geq 1 - \nu$, where the probability is over coin tosses of the encoding* Enc *and the channel C.*

Following the discussion above, we will mostly be interested in the case that $\mathcal{C}$ is the class of poly-size circuits that induce at most a p fraction of errors. This is captured by the definition below.

Definition 2.13 (A channel that induces a p-fraction of errors)**.** *We say that a class of channels $\mathcal{C} \subseteq \mathcal{F}_n^{\text{rand}}$ induces at most a p-fractions of errors, if for every $C \in \mathcal{C}$ and $z \in \{0,1\}^n$ it holds that $\delta(z, C(z)) \leq p$.*

In recent years, there have been several constructions of coding schemes that decode against classes of compuationally bounded channels that induce a p-fraction of errors [GS16, SS21a, KSS19, SS21b, SS22, SS23]. The reader is referred to [SS23] for more details.

In this paper, we will use the following code for poly-size circuits that induce at most a p-fraction of errors, due to Shaltiel and Silbak [SS23].

Theorem 2.14 (Coding schemes for poly-size channels [SS23])**.** *If E is hard for exponential size nondeterministic circuits then for every constants $0 \leq p < \frac{1}{4}$, $c > 1$, and for every sufficiently small constant $\epsilon > 0$, there exists a constant d, such that for infinitely many n, there is a coding scheme $(\mathrm{Enc}, \mathrm{Dec})$ for circuits of size n^c that induce at most a p-fraction of error, with rate $R \geq 1 - H(p) - \epsilon$, and success probability $1 - \frac{1}{n^c}$. Furthermore, Enc and Dec can be computed in time n^d.*

Remark 2.15. *While Theorem 2.14 is stated for (deterministic) circuits, it immediately extends also to randomized circuits.*

2.3 Pseudorandom generators

We need the following standard definition of pseudorandom generators.

Definition 2.16 (Pseudorandom generators)**.** *A distribution X on n bits is ϵ-***pseudorandom** *for a class $\mathcal{C}$ of functions, if for every $C \in \mathcal{C}$, $|\Pr[C(X) = 1] - \Pr[C(U_n)] = 1]| \leq \epsilon$. A function $G : \{0,1\}^d \to \{0,1\}^n$ is an ϵ-*PRG *for $\mathcal{C}$ if $G(U_d)$ is ϵ-pseudorandom for $\mathcal{C}$. G is* seed-extending *if the function $G'(x) = x \circ G(x)$ is an ϵ-PRG for $\mathcal{C}$.*

The classical result of Impagliazzo and Wigderson [IW97] gives a PRG for poly-size circuits, under the assumption that E is hard for exponential size circuits.

Theorem 2.17 (PRGs from hardness assumptions [IW97])**.** *If E is hard for exponential size circuits then for every constant $c > 1$, there exists a constant $a > 1$ such that for every sufficiently large n, there is a $G : \{0,1\}^{a \cdot \log n} \to \{0,1\}^n$ that is a seed-extending $\frac{1}{n^c}$-PRG for circuits of size n^c. Furthermore, G is computable in time $\mathrm{poly}(n^c)$.*

2.4 Pairwise independent hash functions

We need the following standard definition of pairwise independent hash functions.

Definition 2.18 (Pairwise independent hash functions)**.** *A family $\mathrm{Hash}_{n,d}$ of functions $\mathrm{Hash}_w \colon \{0,1\}^n \to \{0,1\}^d$ indexed by $w \in \{0,1\}^\ell$ is said to be pairwise independent if for every messages $x_0 \neq x_1 \in \{0,1\}^n$ and every $y_0, y_1 \in \{0,1\}^d$ it holds that:*

$$\Pr_{w \leftarrow \{0,1\}^\ell}[\mathrm{Hash}_w(x_0) = y_0 \cap \mathrm{Hash}_w(x_1) = y_1] \leq 1/2^{2d},$$

In particular, for every $x \in \{0,1\}^n$, it holds that $\mathrm{Hash}_{U_\ell}(x) \equiv U_d$.

We use the following standard construction of pairwise independent hash functions due to Carter and Wegeman.

Theorem 2.19 (Pairwise independent hash functions)**.** *For every $d \leq n$ there exists a family $\mathrm{Hash}_{n,d}$ of functions $\mathrm{Hash}_w \colon \{0,1\}^n \to \{0,1\}^d$ indexed by $w \in \{0,1\}^{2n}$ such that $\mathrm{Hash}_{n,d}$ is pairwise independent. Moreover, there is a poly-time algorithm that runs in time $n^{t_{\mathrm{Hash}}}$ for some fixed universal constant t_{Hash}, that given $d, x, \in \{0,1\}^n$ and $w \in \{0,1\}^{2n}$ outputs $\mathrm{Hash}_w(x)$.*

3 Non malleable codes with rate $1 - o(1)$ for poly-size circuits

In this section we formally state Theorem 1.2 which is the main result for this paper. We give an explicit construction of a computationally secure non-malleable codes with rate $1 - o(1)$.

3.1 The construction

We remind the reader that an overview of the construction and analyis appears in Section 1.3. Our construction is presented in the figures below. More specifically in Figure 1 we specify the parameters and ingredients used in our construction, and in Figures 2 and 3 we present the encoding and decoding algorithms that make use of these ingredients.

The following theorem is a our main result and is a formal version for Theorem 1.2 stated in the introduction.

Theorem 3.1 (Main Theorem)**.** *If E is hard for exponential size nondeterministic circuits then for every constant $c > 1$ there exists a constant $t_{\mathrm{Sim}} > c$ such that for every constant $c' > 0$ and for every sufficiently large n the following holds: If the parameters and ingredients are chosen as in Figure 1 then $(\mathrm{Enc}_{\mathrm{nm}}, \mathrm{Dec}_{\mathrm{nm}})$ specified in Figures 2, 3, is a coding scheme from n_{data} bits to n bits, has a rate $R = 1 - 1/n^{0.5}$, and is a $\mathcal{C}$-computationally $1/n^c$-non-malleable for randomized circuits of size at most n^c, where $\mathcal{C}$ is the class of randomized circuits of size at most $n^{c'}$. Furthermore, $(\mathrm{Enc}_{\mathrm{nm}}, \mathrm{Dec}_{\mathrm{nm}})$ is simulatable by a size $n^{t_{\mathrm{Sim}}}$ randomized circuit (where t_{Sim} depends only on c), and $(\mathrm{Enc}_{\mathrm{nm}}, \mathrm{Dec}_{\mathrm{nm}})$ can be computed in time $n^{t_{\mathrm{nm}}}$ where t_{nm} is a constant that depends on c and c'.*

The remainder of this section is devoted to the proof of Theorem 3.1.

3.2 Proof of main theorem 3.1.

We will use the construction specified in Figures 1,2 and 3. In the proof below, we will choose a sufficiently large universal constant c_0 (that will be chosen later).

We are given a constant c, and Figure 1 specifies constants $c_{\mathrm{Hash}}, c_{\mathrm{ssnm}}, a'_{\mathrm{ssnm}}, t_{\mathrm{ssnm}}$. It then chooses $t_{\mathrm{Sim}} = t_{\mathrm{ssnm}} + t_{\mathrm{Hash}} + a'_{\mathrm{ssnm}} + c + c_0/2$, as a function of c (and the aforementioned constants that were determined from c). We are then given a constant c' and Figure 1 proceeds to choose constants $c_{\mathrm{PRG}}, a_{\mathrm{PRG}}$ and a_{ssnm} (as a function of c and c').

Figures 1,2 and 3 specify the construction for the functions $(\mathrm{Enc}_{\mathrm{nm}}, \mathrm{Dec}_{\mathrm{nm}})$. We will show that this construction has the properties guaranteed in Theorem 3.1.

In Figure 4 we specify the simulator that we use to prove Theorem 3.1. More specifically, for every sufficiently large n, and a tampering function $C\colon \{0,1\}^n \to \{0,1\}^n$ that is a size n^c randomized circuit and for every set $H \subseteq \{0,1\}^{\ell + d_{\mathrm{Hash}}}$ we define a probabilistic procedure $\mathrm{Sim}^{C,H}$ which produces an output in $\{0,1\}^{n_{\mathrm{data}}} \cup \{fail, \mathrm{same}\}$. The simulator $\mathrm{Sim}^{C,H}$ of Figure 4 makes use of the following definition.

Definition 3.2 (Set-bounded-PRG)**.** *Given a set $H \subseteq \{0,1\}^{\ell + d_{\mathrm{Hash}}}$ and a* PRG *$\hat{G}$ we define the* set-bounded-PRG *function $\hat{G}^H$ that given $(s, \tau) \in H$, outputs $\hat{G}(s)$. We will often think of $\hat{G}^H$ as a truth table (of length $|H| \cdot 4n$ bits) and the simulator will be hardwired with such a truth table.*

For every randomized channel C (that is a size n^c randomized circuit) we define the tampering experiment. This is the experiment in which $\mathrm{Enc}_{\mathrm{nm}}$ is applied on message m with uniform randomness (S, R)). Then the channel corrupts the codeword, and finally, the decoder decodes.

Definition 3.3. *For a circuit $C\colon \{0,1\}^n \to \{0,1\}^n$ and a message $m \in \{0,1\}^n$ define:*

$$\mathrm{Tamper}_{\mathrm{C}}(m) = \mathrm{Dec}_{\mathrm{nm}}(C(\mathrm{Enc}_{\mathrm{nm}}(m; S, R)))$$

where $S \leftarrow U_\ell$ and $R \leftarrow U_{d_{\mathrm{ssnm}}}$.

Figure 1: Parameters and Ingredients for the proof of Theorem 3.1

Parameters given to the construction: The order in which parameters are chosen in Theorem 3.1 is as follows:

- We are given a constant c and are aiming to construct an ϵ-non-malleable code for the class $\mathcal{C} = size(n^c)$ and $\epsilon = 1/n^c$. In the proof of Theorem 3.1 we will choose a constant t_{Sim} as a function of c. (We are aiming that the simulator for the code will run in time $n^{t_{\mathrm{Sim}}}$).
- After choosing the constant t_{Sim}, we are given a constant c', and are aiming to get a non-malleable code that is $\mathcal{C}'$-computationally secure for $\mathcal{C}' = size(n^{c'})$.

Message length and block length of the coding scheme:

- The coding scheme that we construct will have block length n. Throughout, we assume that n is sufficiently large, and that other parameters are chosen as a function of n.
- Let $n_{\mathrm{data}} = n - n^{0.1}$. The coding scheme that we will construct will have message length n_{data}. Let $n_{\mathrm{ctrl}} = n - n_{\mathrm{data}} = n^{0.1}$. In the construction the n bit codeword will composed of two parts of length n_{data} and n_{ctrl}.

Ingredients and additional parameters:

A large constant c_0: Let $c_0 > 1$ be a sufficiently large universal constant that we will choose in the proof of Theorem 3.1.

A hash function Hash**:** Let $c_{\mathrm{Hash}} = c + c_0$, and $d_{\mathrm{Hash}} = c_{\mathrm{Hash}} \cdot \log(n)$. We apply Theorem 2.19 to obtain a hash function $\mathrm{Hash}\colon \{0,1\}^{n_{\mathrm{data}}} \times \{0,1\}^{\ell_{\mathrm{Hash}}} \to \{0,1\}^{d_{\mathrm{Hash}}}$ with $\ell_{\mathrm{Hash}} = 2 \cdot n_{\mathrm{data}}$, and also have that Hash runs in time $n^{t_{\mathrm{Hash}}}$.

SS-non-malleable code: Let $c_{\mathrm{ssnm}} = c + c_0$. We apply Theorem 2.9 for circuits of size $n^{c_{\mathrm{ssnm}}}$ and obtain a constant a'_{ssnm} and t_{ssnm}. Theorem 2.9 guarantees that we can choose any constant $a_{\mathrm{ssnm}} > a'_{\mathrm{ssnm}}$ and obtain a coding scheme $(\mathrm{Enc}_{\mathrm{ssnm}}, \mathrm{Dec}_{\mathrm{ssnm}})$ with message length $a_{\mathrm{ssnm}} \cdot \log(n)$ and block length n_{ctrl}. (This follows as Theorem 2.9 guarantees that the block length is a fixed polynomial in the message length, and we have chosen block length $n_{\mathrm{ctrl}} = n^{0.1}$ which is larger than a fixed polynomial in the message length). By Theorem 2.9 we also have that $(\mathrm{Enc}_{\mathrm{ssnm}}, \mathrm{Dec}_{\mathrm{ssnm}})$ is a $(n^{a'_{\mathrm{ssnm}}}, 1/n^{c_{\mathrm{ssnm}}})$-SS-non-malleable code, and the encoding and decoding can run in time $n^{t_{\mathrm{ssnm}}}$. Note that we have not yet chosen the constant a_{ssnm}, and will do so later.

The constant t_{Sim}: We are aiming to prove Theorem 3.1 in which the constant c' can be chosen as a function of a constant t_{Sim} that is determined by the proof. Note that indeed, all parameters and ingredients chosen so far, were chosen as a function of c At this point, we choose $t_{\mathrm{Sim}} = t_{\mathrm{ssnm}} + t_{\mathrm{Hash}} + a'_{\mathrm{ssnm}} + c + c_0/2$, and receive the constant c'.

Seed extending PRG G: Let $c_{\mathrm{PRG}} = t_{\mathrm{ssnm}} + t_{\mathrm{Hash}} + a'_{\mathrm{ssnm}} + c + c' + c_0$. We apply theorem 2.17 for the constant $c_{\mathrm{PRG}} + 1$ and obtain a constant a_{PRG} such that $G\colon \{0,1\}^{a_{\mathrm{PRG}} \cdot \log n} \to \{0,1\}^{2n}$ is $1/n^{c_{\mathrm{PRG}}+1}$-pseudorandom for circuits of size $n^{c_{\mathrm{PRG}}+1}$. Note that the output length of G is $2n \geq n_{\mathrm{data}}$. This allows us to truncate the output to length of G to n_{data}, without harming its properties.

The parameters ℓ and ℓ': We will use $\ell' = a_{\mathrm{PRG}} \cdot \log(n)$ to denote the seed length of the seed-extending PRG G. We set $\ell = 2 \cdot \ell'$, and will define another seed-extending PRG with seed length ℓ below.

A seed-extending PRG $\hat{G}$: Given a seed $s \in \{0,1\}^{\ell}$ we think of it as consisting of two strings $(s_{\mathrm{PRG}}, s_{\mathrm{Hash}}) \in \{0,1\}^{\ell'} \times \{0,1\}^{\ell'}$. We define the seed extending PRG $\hat{G}\colon \{0,1\}^{\ell} \to \{0,1\}^{n_{\mathrm{data}}+\ell_{\mathrm{Hash}}}$ by $\hat{G}(s) = G(s_{\mathrm{PRG}}), G(s_{\mathrm{Hash}})$ (where the output of $G(s_{\mathrm{PRG}})$ is truncated to its first n_{data} and the output of $G(s_{\mathrm{Hash}})$ is truncated to its first $\ell_{\mathrm{Hash}} = 2n$ bits). By a simple hybrid argument we get that $\hat{G}$ is a seed extending PRG that is $1/n^{c_{\mathrm{PRG}}}$-pseudorandom for circuits of size $n^{c_{\mathrm{PRG}}}$.

Message length for SS-non-malleable code: We can now specify our choice for the constant a_{ssnm} stated above, and we take $a_{\mathrm{ssnm}} = 2 \cdot a_{\mathrm{PRG}} + c_{\mathrm{Hash}}$. This choice was made, so that the message length of the ssnm code is $a_{\mathrm{ssnm}} \cdot \log(n) = \ell + d_{\mathrm{Hash}}$. Indeed, we will use the ssnm code to encdoe pairs $(s, \tau) \in \{0,1\}^{\ell} \times \{0,1\}^{d_{\mathrm{Hash}}}$.

Figure 2: Encoding algorithm for the non-malleable code

Definition: We define a randomized function $\mathrm{Enc}_{\mathrm{nm}} : \{0,1\}^{n_{\mathrm{data}}} \to \{0,1\}^{n}$ as follows:

Input:

- A message $m \in \{0,1\}^{n_{\mathrm{data}}}$.
- A "random coin" for the encoding:
  - A string $s = (s_{\mathrm{PRG}}, s_{\mathrm{Hash}})$ of length $\ell = 2\ell'$. Recall that we think of $s_{\mathrm{PRG}}, s_{\mathrm{Hash}} \in \{0,1\}^{\ell'}$.
  - $r \in \{0,1\}^{d_{\mathrm{ssnm}}}$ where d_{ssnm} is the number of random coins needed for the randomized function $\mathrm{Enc}_{\mathrm{ssnm}}$.

Output: A codeword $z = \mathrm{Enc}(m; (s, r))$ of length n.

Operation: Compute $(w_{\mathrm{PRG}}, w_{\mathrm{Hash}}) = \hat{G}(s)$ and do the following:

Prepare data part: We prepare a string z_{data} of length n_{data} by masking the message using the PRG. That is, $z_{\mathrm{data}} = m \oplus w_{\mathrm{PRG}}$. (Recall that we truncate the output of w_{PRG} to length n_{data}).

Prepare control part: We prepare a string z_{ctrl} of length n_{ctrl} as follows:

- Compute $\tau = \mathrm{Hash}_{w_{\mathrm{Hash}}}(m)$.
- Compute $z_{\mathrm{ctrl}} = \mathrm{Enc}_{\mathrm{ssnm}}((s, \tau); r)$ (that is apply $\mathrm{Enc}_{\mathrm{ssnm}}$ on the message (s, τ) using r as randomness).

Output: $z = (z_{\mathrm{data}}, z_{\mathrm{ctrl}})$ (that is, the concatenation of the data part and the control part).

Figure 3: Decoding algorithm for the non-malleable code

Definition: We define $\mathrm{Dec}_{\mathrm{nm}} : \{0,1\}^{n} \to \{0,1\}^{n_{\mathrm{data}}} \cup \{fail\}$ as follows:

Input: A "received word" $v \in \{0,1\}^{n}$.

Output: A message $\bar{m} \in \{0,1\}^{n_{\mathrm{data}}}$ or $fail$.

Operation:

On input $v = (v_{\mathrm{data}}, v_{\mathrm{ctrl}})$ where $v_{\mathrm{data}} \in \{0,1\}^{n_{\mathrm{data}}}$ and $v_{\mathrm{ctrl}} \in \{0,1\}^{n_{\mathrm{ctrl}}}$ do the following:

- Recover control part: Compute $(\bar{s}, \bar{\tau}) = \mathrm{Dec}_{\mathrm{ssnm}}(v_{\mathrm{ctrl}})$. If $(\bar{s}, \bar{\tau}) = fail$ output $fail$, else continue to the data part.
- Recover data part: Compute $(\bar{w}_{\mathrm{PRG}}, \bar{w}_{\mathrm{Hash}}) = \hat{G}(\bar{s})$ and compute $\bar{m} = v_{\mathrm{data}} \oplus \bar{w}_{\mathrm{PRG}}$.
- Check consistency: If $\bar{\tau} = \mathrm{Hash}_{\bar{w}_{\mathrm{Hash}}}(\bar{m})$, output $\bar{m}$, else output $fail$.

The following is our main technical lemma, which implies Theorem 3.1. Specifically, the the lemma states that for every (possibly randomized) tampering channel C of size at most n^c, there exists a small set H such that the simulator $\mathrm{Sim}^{C,H}$ asserts the non-malleability against C.

Lemma 3.4. *For every constant $c > 0$, using the choice of $t_{\mathrm{Sim}} = t_{\mathrm{ssnm}} + t_{\mathrm{Hash}} + a'_{\mathrm{ssnm}} + c + c_0/2$ made in Figure 1, for every constant $c' > 0$ and for every sufficiently large n the following holds: For every randomized channel C of size at most n^c, there exists a set $H \subseteq \{0,1\}^{\ell + d_{\mathrm{Hash}}}$, such that for every message*

Figure 4: Simulator

Definition: We define the simulator $\mathrm{Sim}^{H,C}$, for a set H and tampering function C:

Parameters: A set $H \subseteq \{0,1\}^{\ell+d_{\mathrm{Hash}}}$ and a tampering function $C\colon \{0,1\}^n \to \{0,1\}^n$.
Non-uniform advice: The "set-bounded-PRG" function $\hat{G}^H$. (See Definition 3.2).
Input: 1^n.
Output: A "message" $\bar{m} \in \{0,1\}^{n_{\mathrm{data}}} \cup \{fail, \mathrm{same}\}$.
Operations of the simulator:

Sample Randomness: We start by uniformly sampling the following strings

- $s \leftarrow \{0,1\}^{\ell}$,
- $\tau \leftarrow \{0,1\}^{d_{\mathrm{Hash}}}$
- $z_{\mathrm{data}} \leftarrow \{0,1\}^{n_{\mathrm{data}}}$
- $r \leftarrow \{0,1\}^{d_{\mathrm{ssnm}}}$

Compute the SS-non-malleable encoding:

- $z_{\mathrm{ctrl}} = \mathrm{Enc}_{\mathrm{ssnm}}((s,\tau);r)$.

Run the tampering function:

- $v = (v_{\mathrm{data}}, v_{\mathrm{ctrl}}) = C(z_{\mathrm{data}}, z_{\mathrm{ctrl}})$.

Attempt to decode: Compute $(\bar{s},\bar{\tau}) = \mathrm{Dec}_{\mathrm{ssnm}}(v_{\mathrm{ctrl}})$, if $(\bar{s},\bar{\tau}) = fail$ output $fail$ else continue:

1. If $(\bar{s},\bar{\tau}) = (s,\tau)$,
   - if $v_{\mathrm{data}} = z_{\mathrm{data}}$ output same.
   - else, output $fail$.
2. If $(\bar{s},\bar{\tau}) \in H$, compute $\hat{G}^H(\bar{s}) = (\bar{w}_{\mathrm{PRG}}, \bar{w}_{\mathrm{Hash}})$ and $\bar{m} = v_{\mathrm{data}} \oplus \bar{w}_{\mathrm{PRG}}$.
   - If $\bar{\tau} = \mathrm{Hash}_{\bar{w}_{\mathrm{Hash}}}(\bar{m})$, output $\bar{m}$
   - else output $fail$.
3. Else, output $fail$.

$m \in \{0,1\}^{n_{\mathrm{data}}}$:

$$\mathrm{Copy}(m, \mathrm{Sim}^{C,H}(1^n)) \approx^{\mathcal{C}}_{\epsilon} \mathrm{Tamper}_{\mathrm{C}}(m).$$

Where $\mathrm{Sim}^{C,H}$ *is as defined in Figure 4,* $\epsilon = 1/n^{c_{\mathrm{ssnm}}} + 2 \cdot 1/n^{c_{\mathrm{PRG}}} + 1/n^{c_{\mathrm{Hash}}}$ *and* $\mathcal{C}$ *is the class of randomized circuits of size at most* $n^{c'}$*. Moreover, the simulator* $\mathrm{Sim}^{C,H}$ *can be implemented by a size* $n^{t_{\mathrm{Sim}}}$ *randomized circuit.*

The proof of Lemma 3.4 is given in the subsection below, but first we formally prove that Theorem 3.1 follows from Lemma 3.4.

Proof of Theorem 3.1.

Proof of Theorem 3.1. Let n be sufficiently large, and note that by the choice of parameters, it indeed holds that the code has rate $R = \frac{n_{\mathrm{data}}}{n} = \frac{n_{\mathrm{data}}}{n_{\mathrm{data}}+n_{\mathrm{ctrl}}} \geq 1 - 1/n^{0.5}$ since $n_{\mathrm{ctrl}} = n^{0.1}$.

The correctness of the construction follows immediately by definition from Lemma 3.4. More specifically, note that given a constant c, Lemma 3.4 guaranties that there exists a constant t_{Sim} such that for every constant c', and every randomized channel C of size at most n^c, the simulator $\mathrm{Sim}^{C,H}$ can be implemented

by a randomized circuit of size $n^{t_{\mathrm{Sim}}}$, and correctly simulates the tampering experiment. Also note that by choosing $c_0 > 0$ to be sufficiently large $\epsilon = 1/n^{c_{\mathrm{ssnm}}} + 2 \cdot 1/n^{c_{\mathrm{PRG}}} + 1/n^{c_{\mathrm{Hash}}} < 1/n^c$.

Finally the overall construction runs in some fixed polynomial $n^{t_{\mathrm{nm}}}$, where t_{nm} depends on c and c'. This holds since given c and c' each ingredient in the construction runs in some fixed poly time (that might depend on c and c').

□

3.3 Proof of Lemma 3.4.

This section is dedicated to proving Lemma 3.4. During this subsection we fix a randomized channel C (of size at most n^c).

The strategy of the proof. Our goal is to show that there exists a small set H such that for every message $m \in \{0,1\}^{n_{\mathrm{data}}}$, the tampering experiment $\mathrm{Tamper}_{\mathrm{C}}(m)$, and the output of the simulator $\mathrm{Sim}^{C,H}(1^n)$ are computationally indistinguishable.

For every set $H \subseteq \{0,1\}^{\ell+d_{\mathrm{Hash}}}$, and every message $m \in \{0,1\}^{n_{\mathrm{data}}}$ we will define a circuit $\mathrm{Dist}_m^{C,H}$ that receives input (Y,S), and has size that is proportional to H. This circuit is defined in Figure 5. Loosely speaking, the circuit $\mathrm{Dist}_m^{C,H}$ will simulate $\mathrm{Tamper}_{\mathrm{C}}(m)$, with two important modifications:

- In the encoding phase, $\mathrm{Dist}_m^{C,H}$ will use the string Y instead of $\hat{G}(S)$. This will mean that when $\mathrm{Dist}^{C,H}(m)$ receives input $S \leftarrow \{0,1\}^\ell$, and $Y = \hat{G}(S)$, the encoding phase is run exactly like in the real experiment $\mathrm{Tamper}_{\mathrm{C}}(m)$. However, when $\mathrm{Dist}^{C,H}(m)$ receives input $S \leftarrow \{0,1\}^\ell$, and $Y \leftarrow U_{4n}$, the output becomes more similar to the experiment of the simulator.
- In the decoding phase, $\mathrm{Dist}_m^{C,H}$ will use the set-bounded PRG $\hat{G}^H$ instead of the PRG $\hat{G}$. This means that when H is the full set $\{0,1\}^{\ell+d_{\mathrm{Hash}}}$, then $\hat{G}^H = \hat{G}$. However, we can hope to use the SS-non-malleability of $(\mathrm{Enc}_{\mathrm{ssnm}}, \mathrm{Dec}_{\mathrm{ssnm}})$ to show that there exists a small set H, such that the difference between $\hat{G}^H$ and $\hat{G}$ is immaterial.

More specifically, we will show through a series of hybrids that in the probability space where $S \leftarrow U_\ell$ and $Y \leftarrow U_{4n}$:

- When H is the full set $\mathrm{Full} = \{0,1\}^{\ell+d_{\mathrm{Hash}}}$, the output $\mathrm{Dist}_m^{C,\mathrm{Full}}(\hat{G}(S),S)$, is distributed exactly like $\mathrm{Tamper}_{\mathrm{C}}(m)$.
- We will use SS-non-malleability to argue that there exists a small set H, such that the previous distribution is close (in statistical distance) to $\mathrm{Dist}_m^{C,H}(\hat{G}(S),S)$.[13]
- As H is small, and the size of $\mathrm{Dist}_m^{C,H}$ is proportional to H, we will have that $\mathrm{Dist}_m^{C,H}$ is fooled by the seed-extending PRG $\hat{G}$, and in particular that the distribution $\mathrm{Dist}_m^{C,H}(\hat{G}(S),S)$ is computationally indistinguishable from $\mathrm{Dist}_m^{C,H}(Y,S)$.
- Finally, we will argue that with this small H, the distribution $\mathrm{Dist}_m^{C,H}(Y,S)$ is statistically close to the simulated distribution $\mathrm{Sim}^{C,H}(1^n)$.

Together, this will imply that there indeed exists a small set H such that for every message m, the real experiment $\mathrm{Tamper}_{\mathrm{C}}(m)$ is computationally indistinguishable from the simulated distribution $\mathrm{Sim}^{C,H}(1^n)$, as required.

[13] In fact, in the formal proof, this step will be more complicated, and we will defer the definition of H to the experiment defined in the next item, and will use a sepoarate argument to argue that the same H is also good in the epxeriment described in this item. Nevertheless, for the purpose of explaining the proof strategy, it is more natural to think as if H is already defined at this stage.

The forma proof. We now implement this strategy. We start by defining the circuit $\mathrm{Dist}_m^{C,H}$. This definition is specified in Figure 5.

Figure 5: The circuit Dist

Definition: We define a circuit $\mathrm{Dist}_m^{C,H}$, for a message m, a set H and tampering function C:

Parameters: A set $H \subseteq \{0,1\}^{\ell+d_{\mathrm{Hash}}}$ and a tampering function $C\colon \{0,1\}^n \to \{0,1\}^n$ and a message $m \in \{0,1\}^n$.

Non-uniform advice: The "set-bounded-PRG" function $\hat{G}^H$. (See Definition 3.2).

Input: A string $(w_{\mathrm{PRG}}, w_{\mathrm{Hash}}, s) \in \{0,1\}^{n_{\mathrm{data}}+\ell_{\mathrm{Hash}}} \times \{0,1\}^{\ell}$.

Output: A "message" $\bar{m} \in \{0,1\}^{n_{\mathrm{data}}} \cup \{fail, \mathrm{same}\}$.

Operations of Dist**:**

Sample randomness and compute the control and data part:

- $r \leftarrow \{0,1\}^{d_{\mathrm{ssnm}}}$,
- $\tau = \mathrm{Hash}_{w_{\mathrm{Hash}}}(m)$,
- $z_{\mathrm{ctrl}} = \mathrm{Enc}_{\mathrm{ssnm}}((s,\tau); r)$,
- $z_{\mathrm{data}} = m \oplus w_{\mathrm{PRG}}$.

Run the tampering function:

- $v = (v_{\mathrm{data}}, v_{\mathrm{ctrl}}) = C(z_{\mathrm{data}}, z_{\mathrm{ctrl}})$.

Attempt to decode: Compute $(\bar{s}, \bar{\tau}) = \mathrm{Dec}_{\mathrm{ssnm}}(v_{\mathrm{ctrl}})$, if $(\bar{s}, \bar{\tau}) = fail$ output $fail$ else continue:

1. If $(\bar{s}, \bar{\tau}) = (s, \tau)$
   - set $\bar{m} = v_{\mathrm{data}} \oplus w_{\mathrm{PRG}}$.
   - If $\tau = \mathrm{Hash}_{w_{\mathrm{Hash}}}(\bar{m})$ output $\bar{m}$,
   - else, output $fail$.
2. If $(\bar{s}, \bar{\tau}) \in H$, compute $\hat{G}^H(\bar{s}) = (\bar{w}_{\mathrm{PRG}}, \bar{w}_{\mathrm{Hash}})$
   - and set $\bar{m} = v_{\mathrm{data}} \oplus \bar{w}_{\mathrm{PRG}}$.
   - If $\bar{\tau} = \mathrm{Hash}_{\bar{w}_{\mathrm{Hash}}}(\bar{m})$, output $\bar{m}$,
   - else, output $fail$.
3. Else, output $fail$.

We also define the following:

- Let $\mathrm{Full} = \{0,1\}^{\ell+d_{\mathrm{Hash}}}$.
- Recall that for a set $H \in \{0,1\}^{\ell+d_{\mathrm{Hash}}}$, $\hat{G}^H$ is the "set-bounded-PRG" given as a non-uniform advice (see Definition 3.2).
- $S \leftarrow U_\ell$
- $Y = (Y_{\mathrm{PRG}}, Y_{\mathrm{Hash}}) \leftarrow U_{n_{\mathrm{data}}+\ell_{\mathrm{Hash}}}$

We now observe that for every small set H, $\mathrm{Dist}_m^{C,H}$ can be implemented by a small circuit. This will be useful, as we have set the PRG $\hat{G}$ to fool circuits of that size, and this will allow us to conclude that the two aforementioned output distributions of $\mathrm{Dist}_m^{C,H}$ are computationally indistinguishable (for every choice of small set H).

Claim 3.5. *Let $c_{\mathrm{Dist}} = t_{\mathrm{ssnm}} + t_{\mathrm{Hash}} + a'_{\mathrm{ssnm}} + c + c_0/2$ (and note that c_{Dist} depends on c, but not on c'). For every set H of size $n^{a'_{\mathrm{ssnm}}}$, there exists a non-uniform (randomized) circuit of size at most $n^{c_{\mathrm{Dist}}}$ that computes $\mathrm{Dist}_m^{C,H}$. Moreover, for every input (w,s) the values of the variables $(\bar{s},\bar{\tau})$ and $\bar{m}$ in the instantiation of $\mathrm{Dist}_m^{C,H}(w,s)$ can also be computed in the same size.*

Proof of Claim 3.5. The claim follows by construction since the function $\hat{G}^H$ can be given as a non-uniform advice of size $n^{a'_{\mathrm{ssnm}}+1}$, $(\mathrm{Enc}_{\mathrm{ssnm}}, \mathrm{Dec}_{\mathrm{ssnm}})$ runs in time $n^{t_{\mathrm{ssnm}}}$, Hash runs in time $n^{t_{\mathrm{Hash}}}$ and C runs in n^c. Overall, since $c_{\mathrm{Dist}} = t_{\mathrm{ssnm}} + t_{\mathrm{Hash}} + a'_{\mathrm{ssnm}} + c + c_0/2$ by choosing c_0 to be sufficiently large it follows that $\mathrm{Dist}_m^{C,H}$ can be computed by a size $n^{c_{\mathrm{Dist}}}$ randomized circuit. Recall that the constants $t_{\mathrm{ssnm}}, t_{\mathrm{Hash}}$ and a'_{ssnm} are chosen as a function of c and c_0 is a universal constant, and so c_{Dist} depends on c but not on c'. □

The proof of Lemma 3.4 will follow from the four claims below (which are the formal implementation of the aforementioned strategy). More specifically, Claim 3.8 relates the experiment $\mathrm{Tamper}_{\mathrm{C}}(m)$ to the instantiation of $\mathrm{Dist}_m^{C,\mathrm{Full}}(\hat{G}(S),S)$. Claim 3.8 (that is proven in the next section) shows that there exists a small set H such that the latter distribution is statistically close to the instantiation of $\mathrm{Dist}_m^{C,H}(\hat{G}(S),S)$ (this is the same experiment, except that this time a small set H is used, rather than the set of all strings). Claim 3.7 uses the pseudorandomness of G to show that the latter distribution is computationally indistinguishable from $\mathrm{Dist}_m^{C,H}(Y,S)$. Finally, Claim 3.6 (that is proven in the next section) shows that the latter distribution is statistically close to the simulation experiment.

Claim 3.6. *For every set $H \subseteq \{0,1\}^{\ell+d_{\mathrm{Hash}}}$ and message $m \in \{0,1\}^{n_{\mathrm{data}}}$ it holds that:*

$$\mathrm{Dist}_m^{C,H}(Y,S) \approx^s_{\epsilon_1} \mathrm{Copy}(m, \mathrm{Sim}^{C,H}(1^n)).$$

where $\epsilon_1 = 1/n^{c_{\mathrm{Hash}}}$.

Claim 3.7. *For every set $H \subseteq \{0,1\}^{\ell+d_{\mathrm{Hash}}}$ of size $n^{a'_{\mathrm{ssnm}}}$ and every message $m \in \{0,1\}^{n_{\mathrm{data}}}$:*

$$\mathrm{Dist}_m^{C,H}(\hat{G}(S),S) \approx^{\mathcal{C}}_{\epsilon_2} \mathrm{Dist}_m^{C,H}(Y,S).$$

where $\mathcal{C}$ is the class of randomized circuits of size at most $n^{c'}$ and $\epsilon_2 = 1/n^{c_{\mathrm{PRG}}}$.

Proof of Claim 3.7. By Claim 3.5, $\mathrm{Dist}_m^{C,H}$ can be computed by a circuit of size $n^{c_{\mathrm{Dist}}}$ where $c_{\mathrm{Dist}} = t_{\mathrm{ssnm}} + t_{\mathrm{Hash}} + a'_{\mathrm{ssnm}} + c + c_0/2$. Recall that $\hat{G}(S)$ is a seed extending PRG for circuits of size $n^{c_{\mathrm{PRG}}}$ and by our choice of parameters $c_{\mathrm{PRG}} \geq c_{\mathrm{Dist}} + c' + c_0/2$. Assume towards contradiction that there exists a circuit $E \in \mathcal{C}$ such that $|\Pr[E(\mathrm{Dist}_m^{C,H}(\hat{G}(S),S)) = 1] - \Pr[E(\mathrm{Dist}_m^{C,H}(Y,S)) = 1]| \leq 1/\epsilon_2$, it follows that for a sufficiently large c_0, the circuit E' of size $n^{c_{\mathrm{PRG}}}$ that on every input first applies $\mathrm{Dist}_m^{C,H}$ and then applies E is able to break $\hat{G}$ contradiction the fact that $\hat{G}(S)$ is a seed extending PRG. □

Claim 3.8. *There exists a set $H \subseteq \{0,1\}^{\ell+d_{\mathrm{Hash}}}$ of size $n^{a'_{\mathrm{ssnm}}}$ such that for every message $m \in \{0,1\}^{n_{\mathrm{data}}}$:*

$$\mathrm{Dist}_m^{C,\mathrm{Full}}(\hat{G}(S),S) \approx^s_{\epsilon} \mathrm{Dist}_m^{C,H}(\hat{G}(S),S).$$

where $\epsilon_3 = 1/n^{c_{\mathrm{PRG}}} + 1/n^{c_{\mathrm{ssnm}}}$.

Claim 3.9.

$$\mathrm{Tamper}_{\mathrm{C}}(m) \equiv \mathrm{Dist}_m^{C,\mathrm{Full}}(\hat{G}(S),S).$$

Proof of Claim 3.9. Follows immediately from construction. □

The proof of Claims 3.6 and 3.8 is given in Section 3.3.1. We now prove Lemma 3.4 follows from the four claims above.

Proof of Lemma 3.4.

Proof of Lemma 3.4. Let H be the set guaranteed by Claim 3.8 of size $n^{a'_{\text{ssnm}}}$ and recall that the constant a'_{ssnm} depends only on c but not on c'. Let $\mathcal{C}$ by the class of randomized circuits of size at most $n^{c'}$. By Claims 3.9,3.8, 3.7 and 3.6 it follows that,

$$\begin{aligned}\text{Tamper}_{\text{C}}(m) &\equiv \text{Dist}_m^{C,\text{Full}}(\hat{G}(S), S)\\ &\approx^s_{\epsilon_3} \text{Dist}_m^{C,H}(\hat{G}(S), S)\\ &\approx^{\mathcal{C}}_{\epsilon_2} \text{Dist}_m^{C,H}(Y, S)\\ &\approx^s_{\epsilon_1} \text{Copy}(m, \text{Sim}^{C,H}(1^n))\end{aligned}$$

where $\epsilon_1 = 1/n^{c_{\text{Hash}}}$, $\epsilon_2 = 1/n^{c_{\text{PRG}}}$ and $\epsilon_3 = 1/n^{c_{\text{PRG}}} + 1/n^{c_{\text{ssnm}}}$. This proves that $\text{Tamper}_{\text{C}}(m) \approx^{\mathcal{C}}_{\epsilon} \text{Copy}(m, \text{Sim}^{C,H}(1^n))$ since $\epsilon = \epsilon_1 + \epsilon_2 + \epsilon_3$ as desired. To conclude, note that the simulator $\text{Sim}^{C,H}$ can be implemented by a size $n^{t_{\text{Sim}}}$ randomized circuit for $t_{\text{Sim}} = t_{\text{ssnm}} + t_{\text{Hash}} + a'_{\text{ssnm}} + c + c_0/2$ where t_{ssnm} and a'_{ssnm} are constants that depend on c but not on c' and t_{Hash} and c_0 are universal constants. This holds since the function $\hat{G}^H$ can be given as a non-uniform advice of size $n^{a'_{\text{ssnm}}+1}$, $(\text{Enc}_{\text{ssnm}}, \text{Dec}_{\text{ssnm}})$ runs in time $n^{t_{\text{ssnm}}}$, Hash runs in time $n^{t_{\text{Hash}}}$ and C runs in n^c. □

3.3.1 Proof of Claims 3.6 and 3.8

In this section we give proofs for the two claims from the previous section that we have not yet proven. We start by proving Claim 3.6 and then prove Claim 3.8.

Proof of Claim 3.6

Recall that $S \leftarrow U_\ell$ and $Y = (Y_{\text{PRG}}, Y_{\text{Hash}}) \leftarrow U_{n_{\text{data}}+\ell_{\text{Hash}}}$. We will make use of the following definitions and Claims.

Definition 3.10. *We define the random variables corresponding to the experiment in* $\text{Dist}_m^{C,H}(Y, S)$ *as follows:*

- $Z^m_{\text{data}} = m \oplus Y_{\text{PRG}}$.
- $\text{T}^m = \text{Hash}_{Y_{\text{Hash}}}(m)$.
- $Z^m_{\text{ctrl}} = \text{Enc}_{\text{ssnm}}(S, \text{T}^m; R)$ *(where R is an independent random sting).*
- $V^m = (V^m_{\text{data}}, V^m_{\text{ctrl}}) = C(Z^m_{\text{data}}, Z^m_{\text{ctrl}})$
- $(\bar{S}^m, \bar{\text{T}}^m) = \text{Dec}_{\text{ssnm}}(V^m)$
- $\bar{O}^m$ *that denotes the final output in* $\text{Dist}_m^{C,H}(Y, S)$.

We omit the superscript m when it is clear from the context.

We will make use of following claims.

Claim 3.11. *For every message m, $(Z^m_{\text{data}}, \text{T}^m, S)$ are independent uniformly distributed random variables. (In particular, $(Z^m_{\text{data}}, \text{T}^m, S)$ is independent of m).*

Proof of Claim 3.11. Recall $Z^m_{\text{data}} = m \oplus Y_{\text{PRG}}$ and $\text{T}^m = \text{Hash}_{Y_{\text{Hash}}}(m)$. Since $Y = (Y_{\text{PRG}}, Y_{\text{Hash}})$ is a uniformly random string that is independent from S, the claim holds by construction and by the properties of the hash function (see Definition 2.18). □

In particular, the above claim implies that the distribution of $(Z^m_{\text{data}}, \mathrm{T}^m, S)$ does not depend on the message m. The following claim shows that if the channel C does not change the "control part" in the codeword, then the decoding algorithm is able to detect if the channel changed the "data part" in the codeword. This intuitively holds, because the channel is not able to find a valid different message that hashes to the same value.

Claim 3.12. *For every m, let $X^m = V^m_{\text{data}} \oplus Y_{\text{PRG}}$. It holds that*

$$\Pr[\text{Hash}_{Y_{\text{Hash}}}(X^m) = \mathrm{T}^m \cap X^m \neq m] \leq 1/2^{d_{\text{Hash}}}.$$

Proof of Claim 3.12. We first fix some message m and remove it from the superscript in the random variables to reduce clutter. Let R denote the randomness (of size at most d_{ssnm}) used by the SS-non malleable encoding and let R_C denote the randomness (of size at most n^c) used by the channel C. Let $Q = (Y_{\text{PRG}}, S, R, R_c)$ and note that Q is independent of Y_{Hash}. We will show that for every $q = (y_{\text{PRG}}, s, r, r_c) \in \{0,1\}^{n_{\text{data}}+\ell+d_{\text{ssnm}}+n^c}$, it holds that

$$\Pr[\text{Hash}_{Y_{\text{Hash}}}(X) = \mathrm{T} \cap X \neq m \mid Q = q] \leq 1/2^{d_{\text{Hash}}}. \tag{1}$$

Note that by averaging, proving Equation 1 concludes the proof. The rest of the proof is dedicated to proving Equation 1.

Let $X^q = (X \mid Q = q)$ and note that T is independent of Q. Writing Equation 1 in this notation, we want to prove that

$$\Pr[\text{Hash}_{Y_{\text{Hash}}}(X^q) = \mathrm{T} \cap X^q \neq m] \leq 1/2^{d_{\text{Hash}}}.$$

Let $Z^q_{\text{data}} = (Z_{\text{data}} \mid Q = q)$ and $Z^q_{\text{ctrl}} = (Z_{\text{ctrl}} \mid Q = q)$ and note that $Z^q_{\text{ctrl}} = \text{Enc}_{\text{ssnm}}(s, T, r)$ and $Z^q_{\text{data}} = m \oplus y_{\text{PRG}}$. Let $V^q_{\text{data}} = (V_{\text{data}} \mid Q = q)$ and $V^q_{\text{ctrl}} = (V_{\text{ctrl}} \mid Q = q)$ and let C^{r_C} denote the circuit C that has has r_C fixed as its randomness. By construction:

$$\begin{aligned}(V^q_{\text{data}}, V^q_{\text{ctrl}}) &= C^{r_C}(Z^q_{\text{data}}, Z^q_{\text{ctrl}}) \\ &= C^{r_C}(m \oplus y_{\text{PRG}}, \text{Enc}_{\text{ssnm}}(s, T, r)).\end{aligned}$$

Since $X^q = V^q_{\text{data}} \oplus y_{\text{PRG}}$, and having fixed all the randomness (in the experiment of $\text{Dist}^{C,H}_m(Y, S)$) except Y_{Hash}, by construction the value of X^q depends only on T. For every $\tau \in \{0,1\}^{d_{\text{Hash}}}$ let $x^{q,\tau} \in \{0,1\}^{n_{\text{data}}}$ denote the constant value for which

$$x^{q,\tau} = X^q \mid \mathrm{T} = \tau.$$

It holds that,

$$\begin{aligned}
&\Pr[\mathrm{Hash}_{Y_{\mathrm{Hash}}}(X^q) = \mathrm{T} \cap X^q \neq m] \\
&\leq \sum_{\tau \in \{0,1\}^{d_{\mathrm{Hash}}}} \Pr[\mathrm{Hash}_{Y_{\mathrm{Hash}}}(X^q) = \mathrm{T} \cap X^q \neq m \cap \mathrm{T} = \tau] \\
&= \sum_{\tau \in \{0,1\}^{d_{\mathrm{Hash}}}} \Pr[\mathrm{Hash}_{Y_{\mathrm{Hash}}}(x^{q,\tau}) = \tau \cap x^{q,\tau} \neq m \cap \mathrm{T} = \tau] \\
&= \sum_{\tau \in \{0,1\}^{d_{\mathrm{Hash}}}} \Pr[\mathrm{Hash}_{Y_{\mathrm{Hash}}}(x^{q,\tau}) = \tau \cap x^{q,\tau} \neq m \cap \mathrm{Hash}_{Y_{\mathrm{Hash}}}(m) = \tau] \\
&= \sum_{\tau \in \{0,1\}^{d_{\mathrm{Hash}}} : x^{q,\tau} \neq m} \Pr[\mathrm{Hash}_{Y_{\mathrm{Hash}}}(x^{q,\tau}) = \tau \cap \mathrm{Hash}_{Y_{\mathrm{Hash}}}(m) = \tau] \\
&\leq 2^{d_{\mathrm{Hash}}} \cdot 1/2^{2d_{\mathrm{Hash}}} \\
&= 1/2^{d_{\mathrm{Hash}}}
\end{aligned}$$

Where the last inequality follows by the hash function properties, see Definition 2.18. □

We are finally ready to prove Claim 3.6:

Proof of Claim 3.6. Consider the hybrid distribution $\widetilde{\mathrm{Dist}}_m^{C,H}$ that is defined to act exactly as $\mathrm{Dist}_m^{C,H}$, with the exception that at item 1 in the "Attempt to decode" stage it instead does the following:

1. If $(\bar{s}, \bar{\tau}) = (s, \tau)$
   - if $z_{\mathrm{data}} = v_{\mathrm{data}}$ output m.
   - else, output $fail$.

This change is made so that $\widetilde{\mathrm{Dist}}_m^{C,H}$ mimics the behavior of the simulator $\mathrm{Sim}^{C,H}$ (but instead of outputting "same", $\widetilde{\mathrm{Dist}}_m^{C,H}$ directly outputs m). The proof of the claim, follows by proving the following two items:

1. $\widetilde{\mathrm{Dist}}_m^{C,H}(Y, S) \equiv \mathrm{Copy}(m, \mathrm{Sim}^{C,H}(1^n))$.
2. $\mathrm{Dist}_m^{C,H}(Y, S) \approx_{\epsilon_1}^{s} \widetilde{\mathrm{Dist}}_m^{C,H}(Y, S)$.

Recall that, Z_{data}^m, T^m, S and $\bar{O}^m$ denote the random variables in the experiment $\mathrm{Dist}_m^{C,H}(Y, S)$. By construction, all random variables in $\widetilde{\mathrm{Dist}}_m^{C,H}(Y, S)$ are identical to $\mathrm{Dist}_m^{C,H}(Y, S)$ with the exception of the value of $\bar{m}$ and the final output $\bar{O}^m$. Let $\tilde{O}^m$ denote final output in the experiment $\widetilde{\mathrm{Dist}}_m^{C,H}(Y, S)$ (that is, $\tilde{O}^m = \widetilde{\mathrm{Dist}}_m^{C,H}(Y, S)$).

To prove Item 1, note that by Claim 3.11, $(Z_{\mathrm{data}}^m, \mathrm{T}^m, S)$ are independent random variables. This is exactly the same setup as the experiment $\mathrm{Sim}^{C,H}(1^n)$ where the values of $(z_{\mathrm{data}}, \tau, s)$ are chosen independently from a uniform distribution. Moreover, after setting the values of $(Z_{\mathrm{data}}^m, \mathrm{T}^m, S)$ all remaining operations in $\mathrm{Copy}(m, \mathrm{Sim}^{C,H}(1^n))$ and $\widetilde{\mathrm{Dist}}_m^{C,H}(Y, S)$ are functionally identically, implying that $\mathrm{Copy}(m, \mathrm{Sim}^{C,H}(1^n)) \equiv \widetilde{\mathrm{Dist}}_m^{C,H}(Y, S)$.

It remains to prove Item 2. For the remainder of this proof we drop the superscript m from the random variables for ease of notation. Define

$$E = \left\{(\bar{S}, \bar{\mathrm{T}}) = (S, \mathrm{T})\right\} \wedge \{V_{\mathrm{data}} \neq Z_{\mathrm{data}}\} \wedge \left\{\bar{\mathrm{T}} = \mathrm{Hash}_{Y_{\mathrm{Hash}}}(V_{\mathrm{data}} \oplus Y_{\mathrm{PRG}})\right\} \tag{2}$$

and let $\bar{E}$ be the complement event of E (note that this event is the same for both experiments since $(Z, V, \mathrm{T}, S, \bar{\mathrm{T}}, \bar{S}, Y)$ are the same random variables in both experiments). Since $\widetilde{\mathrm{Dist}}_m^{C,H}(Y, S)$ and $\mathrm{Dist}_m^{C,H}(Y, S)$ only differ in Item 1 in the "Attempt to decode" stage, it follows by construction that

$$\bar{O} \mid \bar{E} \equiv \tilde{O} \mid \bar{E} \tag{3}$$

Thus to conclude the proof it suffices to show that $\Pr[E] \leq \epsilon_1 = 1/n^{c_{\mathrm{Hash}}}$.

Let $X = V_{\mathrm{data}} \oplus Y_{\mathrm{PRG}}$. Since $Z_{\mathrm{data}} = m \oplus Y_{\mathrm{PRG}}$ it follows that:

$$E \implies \{X \neq m\}.$$

Moreover, since $E \implies \left\{\mathrm{T} = \bar{\mathrm{T}}\right\}$, it also holds that

$$E \implies \{\mathrm{Hash}_{Y_{\mathrm{Hash}}}(X) = T\}.$$

This implies that

$$\begin{aligned}\Pr[E] &\leq \Pr[\mathrm{Hash}_{Y_{\mathrm{Hash}}}(X) = \mathrm{T} \cap X \neq m] \\ &\leq 1/2^{d_{\mathrm{Hash}}} = 1/n^{c_{\mathrm{Hash}}}.\end{aligned}$$

Where the last inequality follows by Claim 3.12, and the equality follows since we chose $d_{\mathrm{Hash}} = c_{\mathrm{Hash}} \cdot \log(n)$. This concludes the proof of Claim 3.6. □

Proof of Claim 3.8

We make use of the following claims and definition in our proof of Claim 3.8. The following claim shows that by applying the SS-non-malleable code on the "control part" (that is, (S, T)), we are able to limit the number the possible "corrupt control part" $(\bar{S}, \bar{\mathrm{T}})$ that might arise when decoding a tampered codeword. Specifically, there exists a small set H such that $(\bar{S}, \bar{\mathrm{T}})$ is contains in H when decoding a corrupt codeword in the experiment $\mathrm{Dist}_m^{C,H}(Y, S)$. We stress that, the set H depends only on the channel C, but not on the message m, this intuitively holds since the data part that contained the message m is masked by the uniform and independent string Y_{data}.

Claim 3.13. *There exists a set H of size $n^{a'_{\mathrm{ssnm}}}$ such that for every message $m \in \{0,1\}^{n_{\mathrm{data}}}$:*

$$\Pr[(\bar{S}^m, \bar{\mathrm{T}}^m) \notin H \cup \{(S, \mathrm{T}^m), fail\}] \leq 1/n^{c_{\mathrm{ssnm}}}$$

Proof of Claim 3.13. Define $C_{\mathrm{ctrl}} \colon \{0,1\}^n \to \{0,1\}^{n_{\mathrm{ctrl}}}$ as follows, for every input $z \in \{0,1\}^n$ run $C(z)$ and output the last n_{ctrl} bits from $C(z)$. This definition is made so that $V_{\mathrm{ctrl}}^m = C_{\mathrm{ctrl}}(Z_{\mathrm{data}}^m, Z_{\mathrm{ctrl}}^m)$.

Recall also that by Claim 3.11 for every message m, $(Z_{\mathrm{data}}^m, \mathrm{T}^m, S)$ are independent uniformly distributed random variables. This implies that Z_{data}^m and Z_{ctrl}^m have the same distribution for every message m, and that Z_{data}^m and Z_{ctrl}^m are independent variables. Thus, for $C' \colon \{0,1\}^{n_{\mathrm{ctrl}}} \to \{0,1\}^{n_{\mathrm{ctrl}}}$ defined by $C'(x) = C_{\mathrm{ctrl}}(U_{n_{\mathrm{data}}}, x)$, it holds that $V_{\mathrm{ctrl}}^m \equiv C'(Z_{\mathrm{ctrl}}^m)$.

Recall that $c_{\text{ssnm}} = c + c_0$, and note that by taking c_0 to be a sufficiently large universal constant, it holds that C' is a randomized circuit of size at most $n^{c_{\text{ssnm}}}$. Thus, by the SS-non-malleability property applied against C', it follows that there exists a set H of size $n^{a'_{\text{ssnm}}}$ such that:

$$\begin{aligned}
&\Pr\left[\text{Dec}_{\text{ssnm}}\left(C'(Z^m_{\text{ctrl}}\right) \notin H \cup \{(S, \text{T}^m), fail\}\right] \\
&= \Pr\left[\text{Dec}_{\text{ssnm}}(V^m_{\text{ctrl}}) \notin H \cup \{(S, \text{T}^m), fail\}\right] \\
&= \Pr\left[(\bar{S}^m, \bar{\text{T}}^m) \notin H \cup \{(S, \text{T}^m), fail\}\right] \\
&\leq 1/n^{c_{\text{ssnm}}}.
\end{aligned}$$

This completes the proof. □

Similarly to Definition 3.10, we define the notation for the experiment $\text{Dist}^{C,H}_m(\hat{G}(S), S)$.

Definition 3.14. *We define the random variables corresponding to the experiment* $\text{Dist}^{C,H}_m(\hat{G}(S), S)$ *as follows:*

- $Z^{m,G}_{\text{data}} = m \oplus G(S_{\text{PRG}})$.
- $\text{T}^{m,G} = \text{Hash}_{G(S_{\text{Hash}})}(m)$.
- $Z^{m,G}_{\text{ctrl}} = \text{Enc}_{\text{ssnm}}(S, \text{T}^{m,G}; R)$ *(where R is an independent random string).*
- $V^{m,G} = (V^{m,G}_{\text{data}}, V^{m,G}_{\text{ctrl}}) = C(Z^{m,G}_{\text{data}}, Z^{m,G}_{\text{ctrl}})$
- $(\bar{S}^{m,G}, \bar{\text{T}}^{m,G}) = \text{Dec}_{\text{ssnm}}(V^{m,G})$
- $\bar{O}^{m,H,G}$ *that denotes the final output in* $\text{Dist}^{C,H}_m(\hat{G}(S), S)$ *for the set H and the message m.*

We omit the superscript m when it is clear from the context.

The following claim, essentially shows that the set H guarantied from Claim 3.13 and originally defined for the experiment $\text{Dist}^{C,H}_m(Y, S)$, gives an equivalent guarantee in the experiment $\text{Dist}^{C,H}_m(\hat{G}(S), S)$.

Claim 3.15. *There exists a set H of size $n^{a'_{\text{ssnm}}}$ such that for every message $m \in \{0,1\}^{n_{\text{data}}}$:*

$$\Pr[(\bar{S}^{m,G}, \bar{\text{T}}^{m,G}) \notin H \cup \left\{(S, \text{T}^{m,G}), fail\right\}] \leq 1/n^{c_{\text{ssnm}}} + \epsilon_{\text{PRG}}$$

Proof of Claim 3.15. Let H be the set guaranteed by Claim 3.13, we will prove the claim for this set. By Claim 3.5 there exists a non-uniform circuit D of size $n^{c_{\text{Dist}}+1} < n^{c_{\text{PRG}}}$ than on input (w, s) computes the values of $(\bar{s}, \bar{\tau})$ and outputs 1 iff $(\bar{s}, \bar{\tau}) \in H \cup \{(s, \tau), fail\}$. Since $(Y, S) \approx^{\mathcal{C}}_{\epsilon_{\text{PRG}}} (G(S), S)$ where $\mathcal{C}$ is the class of (possibly randomized) circuits of size at most $n^{c_{\text{PRG}}}$ and $\epsilon_{\text{PRG}} = 1/n^{c_{\text{PRG}}}$ it follows by Claims 3.5 and 3.13 that

$$\Pr[(\bar{S}^{m,G}, \bar{\text{T}}^{m,G}) \notin H \cup \left\{(S, \text{T}^{m,G}), fail\right\}] \leq 1/n^{c_{\text{ssnm}}} + 1/n^{c_{\text{PRG}}}$$

as otherwise, D can distinguish between (Y, S) and $(G(S), S)$ with advantage greater than ϵ_{PRG}. □

The proof of Claim 3.8 now follows from the above:

Proof of Claim 3.8. Let H be the set guaranteed by Claim 3.15. Recall that for any fixed m, by definition the random variables $S, \text{T}^G, \bar{S}^G, \bar{\text{T}}^G$ are identical in both experiments $\text{Dist}^{C,\text{Full}}_m(\hat{G}(S), S)$ and $\text{Dist}^{C,H}_m(\hat{G}(S), S)$. The only random variable that differ are

$$\bar{O}^{\text{Full},G} = \text{Dist}^{C,\text{Full}}_m(\hat{G}(S), S), \qquad \bar{O}^{H,G} = \text{Dist}^{C,H}_m(\hat{G}(S), S).$$

Let $Q = \left\{(\bar{S}^G, \bar{\mathrm{T}}^G) \in H \cup \left\{(S, \mathrm{T}^G), fail\right\}\right\}$ and note that by construction

$$\bar{O}^{H,G} \mid Q \equiv \bar{O}^{\mathrm{Full},G} \mid Q.$$

Since by Claim 3.15, $\Pr[Q] \geq 1 - (1/n^{c_{\mathrm{ssnm}}} + 1/n^{c_{\mathrm{PRG}}})$, it follows that $\bar{O}^{H,G} \approx^s_{\epsilon_3} \bar{O}^{\mathrm{Full},G}$ for $\epsilon_3 = 1/n^{c_{\mathrm{ssnm}}} + 1/n^{c_{\mathrm{PRG}}}$ concluding the proof.

□

4 Composing non-malleable codes with codes that correct from errors

In this section we show how to compose the non-malleable code of Theorem 1.2 with codes that recover from errors, and achieve the best of both worlds. This prove Corollaries 1.4 and 1.5. The formal restatements of Corollaries 1.4 and 1.5 appears in Corollaries 4.3 and 4.4 below. We start by formally defining the composition of coding schemes.

Definition 4.1 (Composing coding schemes)**.** *Let k_1, n_1, n_2 be integers.*

- *Let $(\mathrm{Enc}_1, \mathrm{Dec}_1)$ be a coding scheme with block length n_1 and message length k_1.*
- *Let $(\mathrm{Enc}_2, \mathrm{Dec}_2)$ be a coding scheme with block length n_2 and message length n_1.*

We define a randomized function $\mathrm{Enc} : \{0,1\}^{k_1} \to \{0,1\}^{n_2}$ and a deterministic function $\mathrm{Dec} : \{0,1\}^{n_2} \to \{0,1\}^{k_1} \cup \{fail\}$ as follows:

$$\mathrm{Enc}(m) = \mathrm{Enc}_2(\mathrm{Enc}_1(m)).$$

On input $v \in \{0,1\}^{n_2}$, if $\mathrm{Dec}_2(v) = fail$ then $\mathrm{Dec}(v)$ is defined to be $fail$. Otherwise, $\mathrm{Dec}(v) = \mathrm{Dec}_2(\mathrm{Dec}_1(v))$.

We now show that when composing two coding schemes, where the first is a code that recovers from errors, and the second is a non-malleable code, we obtain a composed scheme that inherits the best of both worlds, and has rate that is the product of the two rates. In the formulation below, we distinguish between two cases: decoding from errors (as defined in Definition 2.11) and Decoding from errors introduced by poly-size circuits (as defined in Definition 2.12). The first case is used for Corollary 1.4 and the second for Corollary 1.5.

Lemma 4.2. *Let k_1, n_1, n_2 be integers.*

- *Let $(\mathrm{Enc}_1, \mathrm{Dec}_1)$ be a coding scheme with block length n_1 and message length k_1.*
- *Let $(\mathrm{Enc}_2, \mathrm{Dec}_2)$ be a coding scheme with block length n_2 and message length n_1.*

Let $(\mathrm{Enc}, \mathrm{Dec})$ be the functions defined in Definition 4.1.

Rate. *$(\mathrm{Enc}, \mathrm{Dec})$ is a coding scheme with block length $n = n_2$ and message length $k = k_1$. In particular, if the two initial coding schemes have rates R_1 and R_2, then the composed coding scheme has rate $R_1 \cdot R_2$.*

Non-malleability for poly-size circuits. *If the following three conditions hold for some integers s_1, s_2, s':*

- *Enc_2 can be computed by a randomized circuit of size s_2.*
- *Dec_2 can be computed by a circuit of size s_2.*

- $(\mathrm{Enc}_1, \mathrm{Dec}_1)$ *is $\mathcal{C}$-computationally ϵ-non-malleable for the class of randomized circuits of size s_1, where $\mathcal{C}'$ is the class of circuits of size s'.*

Then, $(\mathrm{Enc}, \mathrm{Dec})$ is $\mathcal{C}$-computationally ϵ-non-malleable for the class of randomized circuits of size $s_1 - 2 \cdot s_2$. Furthermore, if $(\mathrm{Enc}_1, \mathrm{Dec}_1)$ is simulatable by a size s circuit, then $(\mathrm{Enc}, \mathrm{Dec})$ is simulatable by a size $s + 2 \cdot s_2$ circuit.

Decoding from errors. *If $(\mathrm{Enc}_2, \mathrm{Dec}_2)$ is a deterministic coding scheme that decodes from a p fraction of errors, then for every fixing of the coin tosses of Enc_1, the composition of the two coding schemes (which is a deterministic coding scheme) decodes from a p fraction of errors.*

Decoding from errors induced by poly-size circuits. *If $(\mathrm{Enc}_2, \mathrm{Dec}_2)$ is a coding scheme that is decodable with success probability $1 - \nu$ from some class $\mathcal{C} \subseteq \mathcal{F}_n^{\mathrm{rand}}$, then $(\mathrm{Enc}, \mathrm{Dec})$ is a coding scheme with success probability $1 - \nu$ from $\mathcal{C}$.*

We prove Lemma 4.2 below, but first we formally restate and prove the corollaries from the introduction.

Corollary 4.3 (Formal restatement of Corollary 1.4)**.** *If for every $0 \leq p < 1/4$ and infinitely many n, there exists a deterministic coding scheme $(\mathrm{Enc}_2, \mathrm{Dec}_2)$ that recovers from a p-fraction of errors, with rate $R_1(p)$ and moreover, Enc_2 and Dec_2 can be computed in time n^t for some universal constant t. Then, if E is hard for exponential size nondeterministic circuits then for every constants $c > 1$, $c' > 1$ and $0 \leq p < 1/4$, for infinitely many n, there exists a coding scheme $(\mathrm{Enc}, \mathrm{Dec})$ that is $\mathcal{C}$-computational $1/n^c$-non-malleable for circuits of size n^c, where $\mathcal{C}$ is the class of randomized circuits of size $n^{c'}$, and has a simulator that can be implemented by a randomized circuit of size $n^{t_{\mathrm{Sim}}}$ for some universal constant t_{Sim} that depends on t and c but not on c'. Furthermore, $(\mathrm{Enc}, \mathrm{Dec})$ recovers from a p-fraction of error, has a rate $R(p) = R_1(p) - o(1)$ and Enc and Dec can be computed in time n^d for a universal constant d that depends on t, c and c'.*

Proof of Corollary 4.3. Follows directly from Theorem 3.1, Lemma 4.2 and the assumptions made in the corollary. Specifically, if E is hard for exponential size nondeterministic circuits then applying Theorem 3.1 for every constants c_1 and c_1', and every sufficiently large n, there exists a coding scheme $(\mathrm{Enc}_1, \mathrm{Dec}_1)$ that is $\mathcal{C}_1$-computational $1/n^{c_1}$-non-malleable for circuits of size n^{c_1}, where $\mathcal{C}$ is the class of randomized circuits of size $n^{c_1'}$, has rate $R_1 = 1 - o(1)$ and has a simulator that can be implemented by a randomized circuit of size $n^{t_{\mathrm{Sim}_1}}$ for some universal constant t_{Sim_1} that depends on c_1 but not on c_1'.

For a given $0 \leq p < 1/4$, and for infinitely many n, let $(\mathrm{Enc}_2, \mathrm{Dec}_2)$ be the deterministic coding scheme given in the assumption that recovers from a p-fraction of error and has rate $R_2(p)$ and let t be the universal constant such that Enc_2 and Dec_2 run in time n^t.

Taking $c_1 = c + 2t$ and $c_1' = c'$, by Lemma 4.2 for infinitely many n, the coding scheme $(\mathrm{Enc}, \mathrm{Dec})$ that is composed form $(\mathrm{Enc}_1, \mathrm{Dec}_1)$ and $(\mathrm{Enc}_2, \mathrm{Dec}_2)$ has rate $R = R_1 \cdot R_2(p) = R_2(p) - o(1)$, is $\mathcal{C}$-computationally $1/n^c$-non-malleable for circuits of size n^c, where $\mathcal{C}$ is the class of randomized circuits of size $n^{c'}$. Moreover, $(\mathrm{Enc}, \mathrm{Dec})$ has a simulator that can be implemented by a randomized circuit of size $n^{t_{\mathrm{Sim}}}$ where $t_{\mathrm{Sim}} = t_{\mathrm{Sim}_1} + 2t$ and as such t_{Sim} is a constant that depends on t on c but not on c'. Finally, by construction $(\mathrm{Enc}, \mathrm{Dec})$ runs in time n^d where d is a constant that depends on t, c and c'. □

We will make use of Theorem 2.14, to get the following corollary.

Corollary 4.4 (Formal restatement of Corollary 1.5)**.** *If E is hard for exponential size nondeterministic circuits then for every constants $c > 1$, $c' > 1$ and $0 \leq p < 1/4$, every sufficiently small constant $\epsilon > 0$, and for infinitely many n, there exists a code $(\mathrm{Enc}, \mathrm{Dec})$ that is $\mathcal{C}$-computational $1/n^c$-non-malleable for circuits of size n^c, where $\mathcal{C}$ is the class of randomized circuits of size $n^{c'}$ and has a simulator that can be*

implemented by a randomized circuit of size $n^{t_{\mathrm{Sim}}}$ for some universal constant t_{Sim} that depends on c but not on c'. Furthermore, $(\mathrm{Enc}, \mathrm{Dec})$ recovers from a p-fraction of error induced by circuits of size at most n^c with success probability $1 - 1/n^c$, has a rate $R'(p) = 1 - H(p) - \epsilon$, and Enc and Dec can be computed in time n^d for a universal constant d that depends on c and c'.

Proof of Corollary 4.4. Follows directly from Theorems 3.1 and 1.2 and Lemma 4.2. Specifically, if E is hard for exponential size nondeterministic circuits then both of the following items holds:

- By Theorem 3.1, for every constant c_1 and c_1', and every sufficiently large n, there exists a coding scheme $(\mathrm{Enc}_1, \mathrm{Dec}_1)$ that is $\mathcal{C}_1$-computational $1/n^{c_1}$-non-malleable for circuits of size n^{c_1}, where $\mathcal{C}$ is the class of randomized circuits of size $n^{c_1'}$, has rate $R_1 = 1 - o(1)$ and has a simulator that can be implemented by a randomized circuit of size $n^{t_{\mathrm{Sim}_1}}$ for some universal constant t_{Sim_1} that depends on c_1 but not on c_1'
- By Theorem 1.2, for every constants $c > 1, 0 \leq p < 1/4$, every sufficiently small constant $\epsilon_1 > 0$, and for infinitely many n, there exists a code $(\mathrm{Enc}_1, \mathrm{Dec}_1)$ that recovers from a p-fraction of error induced by circuits of size at most n^c with success probability $1 - 1/n^c$, has a rate $R_2(p) = 1 - H(p) - \epsilon_1$, and Enc and Dec can be computed in time n_1^t for a universal constant t_1 that depends on c.

Taking $c_1 = c + 2 \cdot 2t_1$, $c_1' = c'$ and $\epsilon_1 = \epsilon/2$, by Lemma 4.2, for infinitely many n, the coding scheme $(\mathrm{Enc}, \mathrm{Dec})$ that is composed form $(\mathrm{Enc}_1, \mathrm{Dec}_1)$ and $(\mathrm{Enc}_2, \mathrm{Dec}_2)$ has rate $R = R_1 \cdot R_2(p) = (1 - o(1)) \cdot (1 - H(p) - \epsilon_1) \geq 1 - H(p) - \epsilon$, is $\mathcal{C}$-computationally $1/n^c$-non-malleable for circuits of size n^c, where $\mathcal{C}$ is the class of randomized circuits of size $n^{c'}$. Moreover, $(\mathrm{Enc}, \mathrm{Dec})$ has a simulator that can be implemented by a randomized circuit of size $n^{t_{\mathrm{Sim}}}$ where $t_{\mathrm{Sim}} = t_{\mathrm{Sim}_1} + 2t_1$ and as such t_{Sim} is a constant that depends on t_1 and c but not on c' (and recall that t_1 depends on c). Finally, by construction $(\mathrm{Enc}, \mathrm{Dec})$ runs in time n^d where d is a universal constant that depends on t, c and c'. □

4.1 Proof of Lemma 4.2.

This section is dedicated to proving Lemma 4.2. Let k_1, n_1, n_2 be integers and let $(\mathrm{Enc}_1, \mathrm{Dec}_1)$ be a coding scheme with block length n_1 and message length k_1 and let $(\mathrm{Enc}_2, \mathrm{Dec}_2)$ be a coding scheme with block length n_2 and message length n_1. Let $(\mathrm{Enc}, \mathrm{Dec})$ be their composition as defined in Definition 4.1. It is immediate from the construction that $(\mathrm{Enc}, \mathrm{Dec})$ is a coding scheme with block length n and message length k (where $n = n_2$ and $k = k_1$). We now prove the properties in Lemma 4.2:

Rate. Follows immediately by construction. More concretely, recall that Enc_1 maps k_1 bits to n_1 bits and Enc_2 maps n_1 bits to n_2, which implies that $R_1 = \frac{k_1}{n_1}$ and $R_2 = \frac{n_1}{n_2}$. Thus, Enc that maps k_1 bit to n_2 bits has a rate of $R = \frac{k_1}{n_2} = \frac{k_1}{n_1} \cdot \frac{n_1}{n_2} = R_1 \cdot R_2$.

Non-malleability for poly-size circuits. Recall, that by assumption $(\mathrm{Enc}_1, \mathrm{Dec}_1)$ is $\mathcal{C}$-computationally ϵ-non-malleable for the class of randomized circuits of size s_1, where $\mathcal{C}$ is the class of circuits of size s'. This implies that for every randomized channel $C_1 \colon \{0,1\}^{n_1} \to \{0,1\}^{n_1}$ of size at most s_1, there exists a simulator $\mathrm{Sim}_1^{C_1}$ that can be implemented by a randomized circuit if size s such that for $D_{C_1} \leftarrow \mathrm{Sim}_1^{C_1}$,

$$D_{C_1} \approx_\epsilon^{\mathcal{C}} \mathrm{Dec}_1(C_1(\mathrm{Enc}_1(m))). \tag{4}$$

To prove the non-malleability property for $(\mathrm{Enc}, \mathrm{Dec})$, we want to show that for every randomized channel $C\colon \{0,1\}^n \to \{0,1\}^n$ of size at most $s_1 - 2 \cdot s_2$, there exists a simulator Sim^C that can be implemented by a circuit of size s such that for $D_C \leftarrow \mathrm{Sim}^C$,

$$D_C \approx^{\mathcal{C}}_{\epsilon} \mathrm{Dec}(C(\mathrm{Enc}(m))).$$

We now define the the simulator Sim^C. For every randomized channel $C\colon \{0,1\}^n \to \{0,1\}^n$ of size at most $s_1 - 2 \cdot s_2$, we define the channel $C_1\colon \{0,1\}^{n_1} \to \{0,1\}^{n_1}$ as follows: On input $x \in \{0,1\}^{n_1}$, output $\mathrm{Dec}_2(C(\mathrm{Enc}_2(x)))$. Note that by definition C_1 can be implemented by a circuit of size s_1 (this, is because C can be implemented by a circuit of size $s_1 - 2 \cdot s_2$, and both Enc_1 and Dec_2 can be implemented by circuits of size s_2). This implies that Equation 4 holds for C_1.

The simulator Sim^C is define to be $\mathrm{Sim}_1^{C_1}$. That is, $D_C \leftarrow \mathrm{Sim}^C$ and $D_C \equiv D_{C_1}$ where $D_{C_1} \leftarrow \mathrm{Sim}_1^{C_1}$. This yields that

$$\begin{aligned} D_C &\equiv D_{C_1} \\ &\approx^{\mathcal{C}}_{\epsilon} \mathrm{Dec}_1(C_1(\mathrm{Enc}_1(m))) \\ &= \mathrm{Dec}_1(\mathrm{Dec}_2(C(\mathrm{Enc}_2(\mathrm{Enc}_1(m))))) \\ &= \mathrm{Dec}(C(\mathrm{Enc}(m))) \end{aligned}$$

Where the first equation follows by the definition of the simulator Sim^C, the second equation follows by the non-malleability property of $(\mathrm{Enc}_1, \mathrm{Dec}_1)$ the third equation follows by the definition of C_1 and the last equation follows by the definition of $(\mathrm{Enc}, \mathrm{Dec})$. To conclude, note that if $\mathrm{Sim}_1^{C_1}$ can be implemented by a size s randomized circuit, then Sim^C can be also be implemented by a size s circuit since Sim^C is defined to be $\mathrm{Sim}_1^{C_1}$.

Decoding from errors. Assume that $(\mathrm{Enc}_2, \mathrm{Dec}_2)$ is a deterministic coding scheme that decodes from a p-fraction of errors. We need to show that for every fixing of the coin tosses of Enc_1, the composition of the two coding schemes (which is a deterministic coding scheme) decodes from a p-fraction of errors. Recall that by definition for every $x \in \{0,1\}^{n_1}$ and every $v \in \{0,1\}^{n_2}$ if $\delta(\mathrm{Enc}_2(x), v) \leq p$ then $\mathrm{Dec}_2(v) = x$.

Given a message $m \in \{0,1\}^{k_1}$, let $\mathrm{Enc}_1(m) = x$ (for some fixing of the coin tosses of Enc_1) and recall that by definition $\mathrm{Enc}(m) = \mathrm{Enc}_2(\mathrm{Enc}_1(m)) = \mathrm{Enc}_2(x)$. Also note that since $(\mathrm{Enc}_1, \mathrm{Dec}_1)$ is a coding scheme, it holds that $\Pr[\mathrm{Dec}(\mathrm{Enc}_1(m)) = m] = 1$. Which implies that for for every fixing of the randomness of Enc_1, $\mathrm{Dec}_1(\mathrm{Enc}_1(m)) = m$.

It follows that for every $v \in \{0,1\}^{n_2}$ if $\delta(\mathrm{Enc}(m), v)) \leq p$ then $\mathrm{Dec}(v) = m$. This holds since, by the above, $\mathrm{Dec}(v) = \mathrm{Dec}_1(\mathrm{Dec}_2(v)) = \mathrm{Dec}_1(x) = m$.

Decoding from errors induced by poly-size circuits. Assume that $(\mathrm{Enc}_2, \mathrm{Dec}_2)$ is a coding scheme that is decodable with success probability $1 - \nu$ from a class $\mathcal{C} \subseteq \mathcal{F}_n^{\mathrm{rand}}$. We need to show that $(\mathrm{Enc}, \mathrm{Dec})$ is a coding scheme with success probability $1 - \nu$ from $\mathcal{C}$.

By assumption for every $x \in \{0,1\}^{n_1}$, and every $C \in \mathcal{C}$ it holds that:

$$\Pr[\mathrm{Dec}_2(C(\mathrm{Enc}_2(x))) = x] \geq 1 - \nu. \tag{5}$$

Since $(\mathrm{Enc}_1, \mathrm{Enc}_1)$ is a coding scheme, it holds that for every $m \in \{0,1\}^{k_1}$, $\Pr[\mathrm{Dec}_1(C(\mathrm{Enc}_2(m) =$

$m] = 1$. Thus, it holds that for every $m \in \{0,1\}^{k_1}$ and every $C \in \mathcal{C}$

$$\begin{aligned}\Pr[\mathrm{Dec}(C(\mathrm{Enc}(m)) = m] &= \Pr[\mathrm{Dec}_1(\mathrm{Dec}_2(C(\mathrm{Enc}_2(\mathrm{Enc}_1(m))) = m]. \\ &\geq \Pr_{x \leftarrow \mathrm{Enc}_1(m)}[\mathrm{Dec}_2(C(\mathrm{Enc}_2(x)) = x \cap \mathrm{Dec}_1(x) = m] \\ &\geq \Pr_{x \leftarrow \mathrm{Enc}_1(m)}[\mathrm{Dec}_2(C(\mathrm{Enc}_2(x)) = x] \geq 1 - \nu.\end{aligned}$$

Where the last inequality follows by Equation 5.

5 Statistically secure non-malleable codes from HTS

In this section we state and prove Theorem 1.3. We will show that given an "excellent" HTS function we are able to construct a statistically secure non-malleable code from our computationally secure non-malleable code given at Theorem 3.1.

5.1 Hard to sample functions (HTS) and a the formal statement of Theorem 1.3

Viola [Vio12] suggests to systematically study functions $f : \{0,1\}^n \to \{0,1\}^{n'}$ that are not only *hard to compute*, but also *hard to sample*, in the sense that no low-complexity sampling algorithm can sample a pair $(X, f(X))$ where X is uniform over $\{0,1\}^n$.

Following Viola [Vio12], Shaltiel and Silbak [SS23] introduce a notion of functions that are "hard to sample" (by poly-size circuits) on any distribution X. Intuitively, one would want to require that for every poly-size circuit A that samples a pair (X, Y), the probability that $Y = f(X)$ is small. However, it is obvious that such functions do not exist as a nonuniform circuit can be hardwired with certain pairs $(x, f(x))$, allowing it to sample a distribution $(X, f(X))$ in the case that X is fixed (or in the case that X has small support).

Shaltiel and Silbak [SS23] deal with this by incorporating the notion of a "small set" (as in small set non-malleable codes). More precisely, it is required that for every sampling circuit A, there is small set H of inputs, such that A is unlikely to sample a pair (X, Y) such that $X \notin H$, and $Y = f(X)$. A formal definition is given below.

Definition 5.1 (Hard To Sample (HTS))**.** *For a function $A : \{0,1\}^r \to \{0,1\}^n$, we use $Z \leftarrow A$ to denote the experiment in which $W \leftarrow U_r$, and $Z = A(W)$.*

*A function $f : \{0,1\}^n \to \{0,1\}^{n'}$ is an (h, ρ)-*HTS *for a class $\mathcal{C}$ of functions, if for every $A \in \mathcal{C}$ that samples a distribution $Z = (X, Y)$ over $\{0,1\}^n \times \{0,1\}^{n'}$, there exists a set $H \subseteq \{0,1\}^n$ of size at most h, such that:*

$$\Pr_{(X,Y) \leftarrow A}[X \notin H \text{ and } Y = f(X)] \leq \rho.$$

Shaltiel and Silbak [SS23] gave two explicit constructions of HTS under the assumption that E is hard for exponential size nondeterministic circuits. The first construction achieves $h = \mathrm{poly}(n)$, but has $n' > n$. (In fact, the SS-non-malleable codes of Theorem 2.9 were achieved by composing the non-malleable codes of [BDL22] with this HTS). The second construction has $n' \ll n$ but instead of $h = \mathrm{poly}(n)$, it only achieves $h = 2^{o(n)}$.

Theorem 1.3 from the introduction, states that we can use an HTS that has both $n' \ll n$, and $h = \mathrm{poly}(n)$ to achieve statistical security in Theorem 1.2. This is stated more formally, in the next theorem and corollary. More specifically, the theorem states general conditions under which an HTS can be used

to convert the security of a non-malleable code from computational to statistical, and the corollary is the restatement of Theorem 1.3, which follows as the code of Theorem 3.1 meets this general condition.

Theorem 5.2 (Statistically secure non-malleable code using HTS functions)**.** *Assume that,*

1. *For every constants $\alpha > 0$ and $c_f > 1$ there exists constants c_h and t_f such that for every sufficiently large n, there exists a function $f\colon \{0,1\}^n \to \{0,1\}^{\alpha \cdot n}$ such that f is an (n^{c_h}, n^{-c_f})-HTS for circuits of size n^{c_f}. Moreover, f can be computed in time n^{t_f}.*
2. *For every constant $c_{\mathrm{nm}} > 1$ there exists a constants $t_{\mathrm{Sim}} > c_{\mathrm{nm}}$ and $R > 0$ such that for every constant $c'_{\mathrm{nm}} > 0$ and every sufficiently large n, there exists a coding scheme $(\mathrm{Enc}, \mathrm{Dec})$ from k bits to n bits that is $\mathcal{C}$-computationally $1/n^{c_{\mathrm{nm}}}$-non-malleable for the class of randomized circuits of $n^{c_{\mathrm{nm}}}$, where $\mathcal{C}$ is the class of randomized circuits of size at most $n^{c'_{\mathrm{nm}}}$. Moreover, $(\mathrm{Enc}, \mathrm{Dec})$ are simulatable by a size $n^{t_{\mathrm{Sim}}}$ randomized circuit, has rate $R > \epsilon$ for some universal constant $\epsilon > 0$, and the encoding and decoding run in time $n^{t_{\mathrm{nm}}}$ where t_{nm} is a constant that depends on c_{nm} and c'_{nm}.*

Then for every constants $\alpha > 0$ and $c > 1$ and every sufficiently large n, there exists a **statistically** *secure $1/n^c$-non-malleable code for the class of randomized circuits of size at most n^c, that runs in time n^d where d is a constant that depends on c and α. Moreover, the rate of the code is $R \cdot \frac{1}{1+\alpha}$.*

Corollary 5.3. *If E is hard for exponential size nondeterministic circuits, and Item 1 in Theorem 5.2 holds. Then, for every constants $1 > \alpha > 0$ and $c > 1$ and every sufficiently large n, there exists a statistically secure $1/n^c$-non-malleable code for the class or randomized circuits of size at most n^c, that runs in time n^d where d is a constant that depends on c and α. Moreover, the rate of the code is $1 - \alpha$.*

Proof of corollary 5.3. The proof follows from Theorems 3.1 and 5.2. Note that assuming that E is hard for exponential size nondeterministic circuits by Theorem 3.1, for every sufficiently large n, there exists a computationally secure non-malleable code with rate $R = 1 - 1/n^{0.5}$ with the same requirements as stated in Item 2 in Theorem 5.2. Thus, by Theorem 5.2 for every constants $\alpha > 0$ and $c > 1$ and every sufficiently large n, there exists a statistically secure $1/n^c$-non-malleable code for the class of randomized circuits of size at most n^c, with rate $(1 - 1/n^{0.5}) \cdot \frac{1}{1+\alpha} \geq 1 - \alpha$. □

We prove Theorem 5.2 in the following section.

5.2 Proof of Theorem 5.2.

Shaltiel and Silbak [SS23] used an HTS to enhance the security of a given code. The high level idea is that given a pair of functions $(\mathrm{Enc}, \mathrm{Dec})$ and an HTS f, one can consider an enhanced code defined as follows: When one wants to encode a message x, then one encodes the message $x \circ f(x)$, and the decoding procedure fails if the decoded pair is not of this form. This is captured (in the case of randomized encoding function) in the definition below.

Definition 5.4. *Let n, k, v, d be parameters. Given:*

- *A function $f : \{0,1\}^k \to \{0,1\}^v$.*
- *Functions $\mathrm{Enc} : \{0,1\}^{k+v} \times \{0,1\}^d \to \{0,1\}^n$, and $\mathrm{Dec} : \{0,1\}^n \to \{0,1\}^{k+v} \cup \{fail\}$*

We define a pair of functions $\mathrm{Enc}_f : \{0,1\}^k \times \{0,1\}^d \to \{0,1\}^n$ and $\mathrm{Dec}_f : \{0,1\}^n \to \{0,1\}^k$ as follows:

- $\mathrm{Enc}_f(x, s) = \mathrm{Enc}(x \circ f(x), s)$.

- $\mathrm{Dec}_f(z)$ *works by applying* $\mathrm{Dec}(z)$. *If* $\mathrm{Dec}(z)$ *does not fail, then it outputs* $x \circ y \in \{0,1\}^{k+m'}$ *and* Dec_f *outputs* x *if* $y = f(x)$*, and fails otherwise.*

Loosely speaking, the cost of this enhancement is that the rate of the code deteriorates, but if the given HTS has output length which is much shorter than the input length, then this deterioration is small. The advantage of this enhancement is that it makes it more difficult for an adversary to break the non-malleability of the code. Loosely speaking, this is because after enhancement, the adversary loses the ability to lead a decoded message which is not of the form $x \circ f(x)$. A bit more formally, we can think of the simulator for the original code as a potential adversary for the HTS. By setting the parameters carefully, and using the relationship between the real experiment and the simulated experiment, this limits the behavior of the real adversary to the non-malleable code. Specifically, this argument shows that an adversary to the non-malleable code is unlikely to lead the decoding to a message that is not in the small set H.

This intuition was used by Shaltiel and Silbak [SS23] in their construction of SS-non-malleable codes. Here we use it in a different way in order to show that if a computationally secure non-malleable code has some additional properties (listed in the lemma below) then the enhanced code actually has *statistical security*. We remind the reader that an overview of this ideas appears in Section 1.3.6.

Lemma 5.5. *Let* k, d, n, v *be parameters. There exists a sufficiently large universal constant* $c_0 > 0$ *such that assuming that for every* $c > 0$ *there exists constants* $c_{\mathrm{nm}}, t_{\mathrm{Sim}}, c'_{\mathrm{nm}}$ *and* c_f, c_h, t_f *such that,* $c_{\mathrm{nm}} > c + c_0$*,* $c_f > t_{\mathrm{Sim}} + c_0$ *and* $c'_{\mathrm{nm}} > t_f + c_h + c_0$*, and for every sufficiently large* n*:*

1. *There exists functions* $\mathrm{Enc} : \{0,1\}^{k+v} \times \{0,1\}^d \to \{0,1\}^n$*, and* $\mathrm{Dec} : \{0,1\}^n \to \{0,1\}^{k+v} \cup \{fail\}$ *such that* $(\mathrm{Enc}, \mathrm{Dec})$ *is a* $\mathcal{C}$*-computationally* $1/n^{c_{\mathrm{nm}}}$*-non-malleable for the class of randomized circuits of size at most* $n^{c_{\mathrm{nm}}}$*, where* $\mathcal{C}$ *is the class of randomized circuits of size at most* $n^{c'_{\mathrm{nm}}}$*. Moreover,* $(\mathrm{Enc}, \mathrm{Dec})$ *is simulatable by a size* $n^{t_{\mathrm{Sim}}}$ *randomized circuit, and* Enc *and* Dec *can be computed in time* $n^{t_{\mathrm{nm}}}$*, where* t_{nm} *is a universal constant that depends on* c*.*
2. *There exists a function* $f \colon \{0,1\}^k \to \{0,1\}^v$ *such that* f *is an* (n^{c_h}, n^{-c_f})*-HTS for circuits of size* n^{c_f}*. Moreover,* f *can be computed in time* n^{t_f}*.*

Then, $(\mathrm{Enc}_f, \mathrm{Dec}_f)$ *(as defined in Definition 5.4 for* $(\mathrm{Enc}, \mathrm{Dec})$ *and* f*) is a* **statistically** *secure* $1/n^c$*-non-malleable code for the class of randomized circuits of size at most* n^c *that runs in time* n^t *where* $t > 1$ *is a constant that depends on* c*. Moreover, the rate of the code is* k/n*.*

We prove Lemma 5.5 in the next subsection, but first we prove Theorem 5.2.

Proving Theorem 5.2.

Proof of Theorem 5.2. Let c_0 be the sufficiently large constant guarantied in Lemma 5.5. For every $\alpha > 0$ and $c > 1$ the following holds:

- By the assumption stated in Item 2 in Theorem 5.2 we can take $c_{\mathrm{nm}} > c + c_0$ and get a constant t_{Sim} and a rate $R > \epsilon$ for some universal constant ϵ, and can chose any constant c'_{nm} (which we specify below) such that there exists functions $\mathrm{Enc} : \{0,1\}^{k+v} \times \{0,1\}^d \to \{0,1\}^n$, and $\mathrm{Dec} : \{0,1\}^n \to \{0,1\}^{k+v} \cup \{fail\}$ and $(\mathrm{Enc}, \mathrm{Dec})$ is a $\mathcal{C}$-computationally $1/n^{c_{\mathrm{nm}}}$-non-malleable for the class of randomized circuits of size at most $n^{c_{\mathrm{nm}}}$, where $\mathcal{C}$ is the class of randomized circuits of size at most $n^{c'_{\mathrm{nm}}}$. Moreover, $(\mathrm{Enc}, \mathrm{Dec})$ is simulatable by a size $n^{t_{\mathrm{Sim}}}$ circuit, has rate R and runs in time $n^{t_{\mathrm{nm}}}$, where t_{nm} is a universal constant that depends on c.

- Let $k = R \cdot \frac{1}{1+\alpha} \cdot n$ and $v = \alpha \cdot k$. Since $k = O(n)$, by the assumption stated in Item 1 in Theorem 5.2 we can take $c_f > t_{\mathrm{Sim}} + c_0$ and get constant c_h and t_f and a function $f\colon \{0,1\}^k \to \{0,1\}^v$ such that f is an (n^{c_h}, n^{-c_f})-HTS for circuits of size n^{c_f}. Moreover, f can be computed in time n^{t_f}. We can now fully specify c'_{nm} from above, and take $c'_{\mathrm{nm}} > t_f + c_h + c_0$.

Note that the above choice of constants was specifically made so that $(\mathrm{Enc}, \mathrm{Dec})$ and f meet the conditions in Lemma 5.5. Thus, by Lemma 5.5, the code $(\mathrm{Enc}_f, \mathrm{Dec}_f)$ is statistically secure $1/n^c$-non-malleable for the class randomized circuits of size at most n^c, and runs in time n^t where $t > 1$ is a constant that depends on c. Moreover, the rate of the code is $k/n = R \cdot \frac{1}{1+\alpha}$ as desired. This concludes the proof of Theorem 5.2. □

5.3 Proof of Lemma 5.5.

In this subsection we prove Lemma 5.5. An overview of this proof appears in Section 1.3.6

Given a constant c, let $c_{\mathrm{nm}}, t_{\mathrm{Sim}}, c'_{\mathrm{nm}}$ and c_f, c_h, t_f be the constants, coding scheme $(\mathrm{Enc}, \mathrm{Dec})$ and function f, guaranteed by the assumptions made at Lemma 5.5. Recall that by assumption $c_{\mathrm{nm}} > c + c_0$, $c_f > t_{\mathrm{Sim}} + c_0$ and $c'_{\mathrm{nm}} > t_f + c_h + c_0$. And that for a every sufficiently large n, $(\mathrm{Enc}, \mathrm{Dec})$ is a coding scheme that maps $k + v$ bits to n bits and is $\mathcal{C}$-computationally $1/n^{c_{\mathrm{nm}}}$-non-malleable for the class of randomized circuits of size at most $n^{c_{\mathrm{nm}}}$, where $\mathcal{C}$ is the class of randomized circuits of size at most $n^{c'_{\mathrm{nm}}}$, and is simulatable by a size $n^{t_{\mathrm{Sim}}}$ randomized circuit. And $f\colon \{0,1\}^k \to \{0,1\}^v$ is an (n^{c_h}, n^{-c_f})-HTS for circuits of size n^{c_f}, and can be computed in time n^{t_f}.

First note that by the non-malleability of $(\mathrm{Enc}, \mathrm{Dec})$, for every randomized circuit C of size at most n^c, there exists a simulating circuit Sim^C of size $n^{t_{\mathrm{Sim}}}$ such that for every $m' \in \{0,1\}^{k+v}$ it holds that

$$\mathrm{Dec}(C(\mathrm{Enc}(m'))) \approx^{\mathcal{C}'}_{\epsilon'} \mathrm{Copy}(m', \mathrm{Sim}^C(1^n)) \tag{6}$$

Where $\mathcal{C}$ is the class of randomized circuits of size at most $n^{c'_{\mathrm{nm}}}$ and $\epsilon' = 1/n^{c_{\mathrm{nm}}}$.

We now define a new (statistical) simulator Sim^C_f for the code $(\mathrm{Enc}_f, \mathrm{Dec}_f)$ using f and above (computational) simulator Sim^C.

Definition 5.6 (The simulator Sim^C_f)**.** *Given the circuit Sim^C and the function f let Sim^C_f be defined as follows: On input 1^n, sample $(m, y) \leftarrow \mathrm{Sim}^C(1^n)$. If $(m, y) \neq fail$, check if $y = f(m)$ if so output m, otherwise output $fail$.*

We will prove that for every randomized channel C of size at most n^c and every message $m \in \{0,1\}^k$:

$$\mathrm{Dec}_f(C(\mathrm{Enc}_f(m))) \approx^s_\epsilon \mathrm{Copy}(m, \mathrm{Sim}^C_f(1^n)) \tag{7}$$

for $\epsilon = 1/n^c$. Note that proving Equation 7 concludes the proof for 5.5. In the following we fix a circuit C and a message m, denote $m' = (m, f(m))$ and consider the probability space:

- $(M_R, Y_R) = \mathrm{Dec}(C(\mathrm{Enc}(m')))$.
- $M^f_R = \mathrm{Dec}_f(C(\mathrm{Enc}_f(m)))$.
- $(M_I, Y_I) = \mathrm{Copy}(m', \mathrm{Sim}^C(1^n))$.
- $M^f_I = \mathrm{Copy}(m, \mathrm{Sim}^C_f(1^n))$.

Stating Equation 7 in this notation, we want to prove that $M^f_R \approx^s_\epsilon M^f_I$. But first we prove that

Claim 5.7. *Let $c'' = c_h + c_0/2$, $\epsilon' = 1/n^{c_{\mathrm{nm}}}$ and let $\mathcal{C}''$ be class of randomized circuits of size at most $n^{c''}$. It holds that*

$$M_R^f \approx_{\epsilon'}^{\mathcal{C}''} M_I^f.$$

Proof of Claim 5.7. Recall that $c'_{\mathrm{nm}} > t_f + c_h + c_0$. Equation 6 states that

$$(M_R, Y_R) \approx_{\epsilon'}^{\mathcal{C}} (M_I, Y_I) \tag{8}$$

where $\mathcal{C}$ is the class of randomized circuits of size at most $n^{c'_{\mathrm{nm}}}$ and $\epsilon' = 1/n^{c_{\mathrm{nm}}}$. Let D^f be the function defined as follows: On input (m, y), if $(m, y) \neq fail$, check if $y = f(m)$ if so output m, otherwise output $fail$. By definition it holds that

$$M_R^f \equiv D^f(M_R, Y_R) \qquad\qquad M_I^f \equiv D^f(M_I, Y_I) \tag{9}$$

Moreover, D^f can be implemented by a circuit of size n^{c_D} where $c_D = t_f + c_0/4$ for some sufficiently large universal constant c_0. Since $c'_{\mathrm{nm}} > c'' + c_D + c_0/4$ the claim holds as otherwise there exists a circuit of size less than $n^{c'_{\mathrm{nm}}}$ that contradicts Equation 8. □

By making use of the HTS properties of f and the above claim, we show that:

Claim 5.8. *There exists a set $H \subset \{0,1\}^k$ of size at most n^{c_h} such that:*

$$\Pr[M_I^f \notin H \cup \{fail\}] \leq \rho$$

and

$$\Pr[M_R^f \notin H \cup \{fail\}] \leq \rho + \epsilon'$$

where $\rho = 1/n^{c_f}$ and $\epsilon' = 1/n^{c_{\mathrm{nm}}}$.

Proof of Claim 5.8. Recall that for a message $m \in \{0,1\}^k$, $m' = (m, f(m))$. and that by assumption $\mathrm{Sim}^C(1^n)$ can be implemented by a size $n^{t_{\mathrm{Sim}}}$ randomized circuit. Let A_m be the randomized circuit that computes $\mathrm{Copy}(m', \mathrm{Sim}^C(1^n))$ (that is, $(M_I, Y_I) \leftarrow A_m$). By the above, A_m can be implemented by a size $n^{t_{\mathrm{Sim}}+c_0/4}$ randomized circuit, for some sufficiently large universal constant c_0. Recall that $c_f > t_{\mathrm{Sim}} + c_0$, thus by the HTS property of f against the circuit A_m, we have that there exists a set $H \subseteq \{0,1\}^k$ of size at most n^{c_h}, such that:

$$\Pr[M_I \notin H \text{ and } Y_I = f(M_I)] \leq \rho.$$

By the definition of Sim_f^C, this implies that:

$$\Pr[M_I^f \notin H \cup \{fail\}] \leq \rho,$$

proving the first part in the claim. To see why the second part holds, consider the distinguisher D^H that on every input $x \in \{0,1\}^k$ it outputs 1 iff $x \in H \cup \{fail\}$. Note that D^H can be implemented by a circuit of size $n^{c_h+c_0/2}$, for a sufficiently large universal constant c_0. Thus by Claim 5.7 it follows that

$$\Pr[M_R^f \notin H \cup \{fail\}] \leq \rho + \epsilon',$$

as otherwise D^H can distinguish between M_I^f and M_R^f with advantage greater than ϵ'. This concludes the proof. □

Proving Lemma 5.5. We are finally, ready to prove Lemma 5.5. The key idea in the proof, is that if the simulated experiment M_I^f and the real experiment M_R^f are statistically far, then there exists an efficient circuit D' that can distinguish M_I^f and M_R^f with roughly the same distinguishing advantage as the statistical distance, contradicting Claim 5.7.

Loosely speaking, this holds since by Claim 5.8, with all but a small probability M_I^f and M_R^f are distributed over a small domain (that is, the set H guaranteed in the claim). This implies that there exists an adversary that is "restricted" to the set H, and is still able to distinguish between M_I^f and M_R^f. Since the set H is small, it follows that there exists an efficient circuit D' (that has H as a non-uniform advice) that is able to implement the adversary.

Proof Lemma 5.5. Assume for contradiction that Equation 7 does not hold. That is, the statistical distance between M_R^f and M_I^f is larger than $\epsilon = 1/n^c$. By the definition of the statistical distance, there exists a set $T \subseteq \{0,1\}^k \cup \{fail\}$ such that:

$$|\Pr[M_R^f \in T] - \Pr[M_I^f \in T]| > \epsilon \tag{10}$$

Let $H \subseteq \{0,1\}^k$ be the set of size n^{c_h} guaranteed by Claim 5.8, and let $H' = T \cap (H \cup \{fail\})$. Consider the distinguisher D' defined as follows: on input x, if $x \in H'$ output 1, otherwise output 0. We will show that

$$|\Pr[D'(M_R^f) = 1] - \Pr[D'(M_I^f) = 1]| > \epsilon' = 1/n^{c_{\text{nm}}}. \tag{11}$$

Note that proving Equation 11 concludes the proof of the lemma. This is because, D' can be implemented by a circuit of size $n^{c_h + c_0/2}$ (as the distinguisher D' can hold the set H' as a non-uniform advice), contradicting Claim 5.7. We now prove Equation 11, for this we define:

$$E_R = \left\{M_R^f \in H \cup \{fail\}\right\} \qquad E_I = \left\{M_I^f \in H \cup \{fail\}\right\} \tag{12}$$

and let $\bar{E}_R$ and $\bar{E}_I$ be the complementary events of E_R and E_I respectively. By Claim 5.8, it holds that for $\rho = 1/n^{c_f}$,

$$\Pr[\bar{E}_I] \leq \rho \qquad \Pr[\bar{E}_R] \leq \rho + \epsilon' \tag{13}$$

Recall that $D'(M_I^f) = 1$ iff $M_I^f \in T$ and $M_I^f \in H$. Writing the total probability,

$$\begin{aligned} \Pr[M_I^f \in T] &= \Pr[M_I^f \in T \cap M_I^f \in H] + \Pr[M_I^f \in T \cap M_I^f \notin H] \\ &= \Pr[D'(M_I^f) = 1] + \Pr[M_I^f \in T \cap M_I^f \notin H] \\ &= \Pr[D'(M_I^f) = 1] + \Pr[M_I^f \in T \cap \bar{E}_I] \end{aligned} \tag{14}$$

Similarly,

$$\Pr[M_R^f \in T] = \Pr[D'(M_R^f) = 1] + \Pr[M_R^f \in T \cap \bar{E}_R] \tag{15}$$

By Equations 14 and 15, and the triangle inequality:

$$\begin{aligned} &|\Pr[M_R^f \in T] - \Pr[M_I^f \in T]| \\ &\leq |\Pr[D'(M_R^f) = 1] - \Pr[D'(M_I^f) = 1]| + |\Pr[M_R^f \in T \cap \bar{E}_R]| + |\Pr[M_I^f \in T \cap \bar{E}_I]| \\ &\leq |\Pr[D'(M_R^f) = 1] - \Pr[D'(M_I^f) = 1]| + \rho + \rho + \epsilon' \end{aligned}$$

Where the last inequality follows from Equation 13. Since by assumption $1/n^c = \epsilon < |\Pr[M_R^f \in T] - \Pr[M_I^f \in T]|$, $\epsilon' = 1/n^{c_{\rm nm}}$ and $\rho = 1/n^{c_f}$ it follows that

$$|\Pr[D'(M_R^f) = 1] \Pr[D'(M_I^f) = 1]| > 1/n^c - 1/n^{c_{\rm nm}} - 2/n^{c_f}.$$

Since $c_{\rm nm} > c + c_0$ and $c_f > c_{\rm nm}$, for some sufficiently large universal constant c_0, it holds that $1/n^c - 1/n^{c_{\rm nm}} - 2/n^{c_f} > 1/n^{c_{\rm nm}}$ proving Equation 11 and concluding the proof as desired. □

Acknowledgement

We are grateful to Banny Applebaum and Iftach Haitner for helpful discussions.

References

[AAG+16] Divesh Aggarwal, Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. Optimal computational split-state non-malleable codes. In Eyal Kushilevitz and Tal Malkin, editors, *Theory of Cryptography - 13th International Conference, TCC 2016-A, Tel Aviv, Israel, January 10-13, 2016, Proceedings, Part II*, volume 9563 of *Lecture Notes in Computer Science*, pages 393–417. Springer, 2016.

[AASY15] B. Applebaum, S. Artemenko, R. Shaltiel, and G. Yang. Incompressible functions, relative-error extractors, and the power of nondeterministic reductions. In *30th Conference on Computational Complexity*, pages 582–600, 2015.

[ADL14] Divesh Aggarwal, Yevgeniy Dodis, and Shachar Lovett. Non-malleable codes from additive combinatorics. In David B. Shmoys, editor, *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 774–783. ACM, 2014.

[AGM+15] Shashank Agrawal, Divya Gupta, Hemanta K. Maji, Omkant Pandey, and Manoj Prabhakaran. A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In Yevgeniy Dodis and Jesper Buus Nielsen, editors, *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, volume 9014 of *Lecture Notes in Computer Science*, pages 375–397. Springer, 2015.

[AIKS16] S. Artemenko, R. Impagliazzo, V. Kabanets, and R. Shaltiel. Pseudorandomness when the odds are against you. In *31st Conference on Computational Complexity, CCC*, volume 50, pages 9:1–9:35, 2016.

[AKO+22] Divesh Aggarwal, Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, Maciej Obremski, and Sruthi Sekar. Rate one-third non-malleable codes. In Stefano Leonardi and Anupam Gupta, editors, *STOC '22: 54th Annual ACM SIGACT Symposium on Theory of Computing, Rome, Italy, June 20 - 24, 2022*, pages 1364–1377. ACM, 2022.

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

[BCL+20] Marshall Ball, Eshan Chattopadhyay, Jyun-Jie Liao, Tal Malkin, and Li-Yang Tan. Non-malleability against polynomial tampering. In Daniele Micciancio and Thomas Ristenpart, editors, *Advances in Cryptology - CRYPTO 2020 - 40th Annual International Cryptology Conference, CRYPTO 2020, Santa Barbara, CA, USA, August 17-21, 2020, Proceedings, Part III*, volume 12172 of *Lecture Notes in Computer Science*, pages 97–126. Springer, 2020.

[BD22] G. Blanc and D. Doron. New near-linear time decodable codes closer to the GV bound. *Electron. Colloquium Comput. Complex.*, TR22-027, 2022.

[BDG+18] Marshall Ball, Dana Dachman-Soled, Siyao Guo, Tal Malkin, and Li-Yang Tan. Non-malleable codes for small-depth circuits. In Mikkel Thorup, editor, *59th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2018, Paris, France, October 7-9, 2018*, pages 826–837. IEEE Computer Society, 2018.

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

[BDKM16] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes for bounded depth, bounded fan-in circuits. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology - EUROCRYPT 2016 - 35th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Vienna, Austria, May 8-12, 2016, Proceedings, Part II*, volume 9666 of *Lecture Notes in Computer Science*, pages 881–908. Springer, 2016.

[BDKM18] Marshall Ball, Dana Dachman-Soled, Mukul Kulkarni, and Tal Malkin. Non-malleable codes from average-case hardness: $${\mathsf {A}}{\mathsf {C}}^0$$, decision trees, and streaming space-bounded tampering. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018 Proceedings, Part III*, volume 10822 of *Lecture Notes in Computer Science*, pages 618–650. Springer, 2018.

[BDL22] M. Ball, D. Dachman-Soled, and J. Loss. (nondeterministic) hardness vs. non-malleability. In *Advances in Cryptology - CRYPTO 2022 - 42nd Annual International Cryptology Conference*, volume 13507, pages 148–177, 2022.

[BGW19] Marshall Ball, Siyao Guo, and Daniel Wichs. Non-malleable codes for decision trees. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 413–434. Springer, 2019.

[BKS+10] B. Barak, G. Kindler, R. Shaltiel, B. Sudakov, and A. Wigderson. Simulating independence: New constructions of condensers, ramsey graphs, dispersers, and extractors. *J. ACM*, 57(4), 2010.

[BOV07] B. Barak, S. J. Ong, and S. P. Vadhan. Derandomization in cryptography. *SIAM J. Comput.*, 37(2):380–400, 2007.

[BV17] N. Bitansky and V. Vaikuntanathan. A note on perfect correctness by derandomization. In *Advances in Cryptology - EUROCRYPT 2017 - 36th Annual International Conference on the Theory and Applications of Cryptographic Techniques*, volume 10211, pages 592–606, 2017.

[CG14] Mahdi Cheraghchi and Venkatesan Guruswami. Non-malleable coding against bit-wise and split-state tampering. In Yehuda Lindell, editor, *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, volume 8349 of *Lecture Notes in Computer Science*, pages 440–464. Springer, 2014.

[CG16] M. Cheraghchi and V. Guruswami. Capacity of non-malleable codes. *IEEE Trans. Inf. Theory*, 62(3):1097–1118, 2016.

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

[CL17] Eshan Chattopadhyay and Xin Li. Non-malleable codes and extractors for small-depth circuits, and affine functions. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1171–1184. ACM, 2017.

[CL20] Eshan Chattopadhyay and Xin Li. Non-malleable codes, extractors and secret sharing for interleaved tampering and composition of tampering. In Rafael Pass and Krzysztof Pietrzak, editors, *Theory of Cryptography - 18th International Conference, TCC 2020, Durham, NC, USA, November 16-19, 2020, Proceedings, Part III*, volume 12552 of *Lecture Notes in Computer Science*, pages 584–613. Springer, 2020.

[CT22] L. Chen and R. Tell. When arthur has neither random coins nor time to spare: Superfast derandomization of proof systems. *Electron. Colloquium Comput. Complex.*, TR22-057, 2022.

[DKO13] Stefan Dziembowski, Tomasz Kazana, and Maciej Obremski. Non-malleable codes from two-source extractors. In Ran Canetti and Juan A. Garay, editors, *Advances in Cryptology - CRYPTO 2013 - 33rd Annual Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2013. Proceedings, Part II*, volume 8043 of *Lecture Notes in Computer Science*, pages 239–257. Springer, 2013.

[DKP21] D. Dachman-Soled, I. Komargodski, and R. Pass. Non-malleable codes for bounded parallel-time tampering. In *Advances in Cryptology - CRYPTO 2021 - 41st Annual International Cryptology Conference, CRYPTO 2021*, volume 12827 of *Lecture Notes in Computer Science*, pages 535–565. Springer, 2021.

[DMOZ22] D. Doron, D. Moshkovitz, J. Oh, and D. Zuckerman. Nearly optimal pseudorandomness from hardness. *J. ACM*, 69(6):43:1–43:55, 2022.

[DPW18] S. Dziembowski, K. Pietrzak, and D. Wichs. Non-malleable codes. *J. ACM*, 65(4):20:1–20:32, 2018.

[Dru13] Andrew Drucker. Nondeterministic direct product reductions and the success probability of SAT solvers. In *54th Annual IEEE Symposium on Foundations of Computer Science, FOCS*, pages 736–745, 2013.

[FMVW16] S. Faust, P. Mukherjee, D. Venturi, and D. Wichs. Efficient non-malleable codes and key derivation for poly-size tampering circuits. *IEEE Trans. Inf. Theory*, 62(12):7179–7194, 2016.

[GMW17] Divya Gupta, Hemanta K. Maji, and Mingyuan Wang. Constant-rate non-malleable codes in the split-state model. *IACR Cryptol. ePrint Arch.*, page 1048, 2017.

[GMW19] Divya Gupta, Hemanta K. Maji, and Mingyuan Wang. Explicit rate-1 non-malleable codes for local tampering. In Alexandra Boldyreva and Daniele Micciancio, editors, *Advances in Cryptology - CRYPTO 2019 - 39th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 18-22, 2019, Proceedings, Part I*, volume 11692 of *Lecture Notes in Computer Science*, pages 435–466. Springer, 2019.

[GS16] V. Guruswami and A. Smith. Optimal rate code constructions for computationally simple channels. *Journal of the ACM (JACM)*, 63(4):35, 2016.

[GST03] Dan Gutfreund, Ronen Shaltiel, and Amnon Ta-Shma. Uniform hardness versus randomness tradeoffs for arthur-merlin games. *Computational Complexity*, 12(3-4):85–130, 2003.

[GSV18] Aryeh Grinberg, Ronen Shaltiel, and Emanuele Viola. Indistinguishability by adaptive procedures with advice, and lower bounds on hardness amplification proofs. In *59th IEEE Annual Symposium on Foundations of Computer Science*, pages 956–966, 2018.

[GW02] O. Goldreich and A. Wigderson. Derandomization that is rarely wrong from short advice that is typically good. In *APPROX-RANDOM*, pages 209–223, 2002.

[HNY17] P. Hubácek, M. Naor, and E. Yogev. The journey from NP to TFNP hardness. In *8th Innovations in Theoretical Computer Science Conference, ITCS*, volume 67, pages 60:1–60:21, 2017.

[IW97] R. Impagliazzo and A. Wigderson. $P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma. In *STOC*, pages 220–229, 1997.

[JST21] F. G. Jeronimo, S. Srivastava, and M. Tulsiani. Near-linear time decoding of ta-shma's codes via splittable regularity. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1527–1536, 2021.

[KOS17] Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Four-state non-malleable codes with explicit constant rate. In Yael Kalai and Leonid Reyzin, editors, *Theory of Cryptography - 15th International Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, Proceedings, Part II*, volume 10678 of *Lecture Notes in Computer Science*, pages 344–375. Springer, 2017.

[KOS18] Bhavana Kanukurthi, Sai Lakshmi Bhavana Obbattu, and Sruthi Sekar. Non-malleable randomness encoders and their applications. In Jesper Buus Nielsen and Vincent Rijmen, editors, *Advances in Cryptology - EUROCRYPT 2018 - 37th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tel Aviv, Israel, April 29 - May 3, 2018*

Proceedings, Part III, volume 10822 of *Lecture Notes in Computer Science*, pages 589–617. Springer, 2018.

[KSS19] S. Kopparty, R. Shaltiel, and J. Silbak. Quasilinear time list-decodable codes for space bounded channels. *To appear in the 60th Annual Symposium on Foundations of Computer Science (FOCS)*, 2019.

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

[Li17] Xin Li. Improved non-malleable extractors, non-malleable codes and independent source extractors. In Hamed Hatami, Pierre McKenzie, and Valerie King, editors, *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, pages 1144–1156. ACM, 2017.

[Li19] Xin Li. Non-malleable extractors and non-malleable codes: Partially optimal constructions. In Amir Shpilka, editor, *34th Computational Complexity Conference, CCC 2019, July 18-20, 2019, New Brunswick, NJ, USA*, volume 137 of *LIPIcs*, pages 28:1–28:49. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019.

[Li23] Xin Li. Two source extractors for asymptotically optimal entropy, and (many) more. *Electron. Colloquium Comput. Complex.*, TR23-023, 2023.

[Lip94] R. J. Lipton. A new approach to information theory. In *11th Annual Symposium on Theoretical Aspects of Computer Science*, pages 699–708, 1994.

[LL12] Feng-Hao Liu and Anna Lysyanskaya. Tamper and leakage resilience in the split-state model. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, volume 7417 of *Lecture Notes in Computer Science*, pages 517–532. Springer, 2012.

[MV05] P. Bro Miltersen and N. V. Vinodchandran. Derandomizing arthur-merlin games using hitting sets. *Computational Complexity*, 14(3):256–279, 2005.

[NW94] N. Nisan and A. Wigderson. Hardness vs. randomness. *JCSS: Journal of Computer and System Sciences*, 49, 1994.

[SS21a] R. Shaltiel and J. Silbak. Explicit list-decodable codes with optimal rate for computationally bounded channels. *Comput. Complex.*, 30(1):3, 2021.

[SS21b] R. Shaltiel and J. Silbak. Explicit uniquely decodable codes for space bounded channels that achieve list-decoding capacity. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing*, pages 1516–1526, 2021.

[SS22] R. Shaltiel and J. Silbak. Error correcting codes that achieve BSC capacity against channels that are poly-size circuits. In *63rd IEEE Annual Symposium on Foundations of Computer Science, FOCS*, pages 13–23, 2022.

[SS23] R. Shaltiel and J. Silbak. Explicit codes for poly-size circuits and functions that are hard to sample on low entropy distributions. *Electronic Colloquium on Computational Complexity (ECCC)*, (149), 2023.

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

[SU06] R. Shaltiel and C. Umans. Pseudorandomness for approximate counting and sampling. *Computational Complexity*, 15(4):298–341, 2006.

[SU09] R. Shaltiel and C. Umans. Low-end uniform hardness versus randomness tradeoffs for am. *SIAM J. Comput.*, 39(3):1006–1037, 2009.

[TS17] A. Ta-Shma. Explicit, almost optimal, epsilon-balanced codes. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC*, pages 238–251, 2017.

[TV00] L. Trevisan and S. P. Vadhan. Extracting randomness from samplable distributions. In *41st Annual Symposium on Foundations of Computer Science*, pages 32–42, 2000.

[Vio12] E. Viola. The complexity of distributions. *SIAM J. Comput.*, 41(1):191–218, 2012.

[Zuc07] David Zuckerman. Linear degree extractors and the inapproximability of max clique and chromatic number. *Theory Comput.*, 3(1):103–128, 2007.