On Resilience to Computable Tampering

Maynard Marshall Ball Jr.

Submitted in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy
under the Executive Committee
of the Graduate School of Arts and Sciences

COLUMBIA UNIVERSITY

2021

Abstract

On Resilience to Computable Tampering

Maynard Marshall Ball Jr.

Non-malleable codes, introduced by Dziembowski, Pietrzak, and Wichs (ICS 2010), provide a means of encoding information such that if the encoding is tampered with, the result encodes something either identical or completely unrelated. Unlike error-correcting codes (for which the result of tampering must always be identical), non-malleable codes give guarantees even when tampering functions are allowed to change every symbol of a codeword.

In this thesis, we will provide constructions of non-malleable codes secure against a variety tampering classes with natural computational semantics:

- **Bounded-Communication:** Functions corresponding to 2-party protocols where each party receives half the input (respectively) and then may communicate $< n/4$ bits before returning their (respective) half of the tampered output.
- **Local Functions (Juntas):** each tampered output bit is only a function of $n^{1-\delta}$ inputs bits, where $\delta > 0$ is any constant (the efficiency of our code depends on δ). This class includes NC^0.
- **Decision Trees:** each tampered output bit is a function of $n^{1/4-o(1)}$ adaptively chosen bits.
- **Small-Depth Circuits:** each tampered output bit is produced by a $c \log(n)/\log\log(n)$-depth circuit of polynomial size, for some constant c. This class includes AC^0.
- **Low Degree Polynomials:** each tampered output field element is produced by a low-degree (relative to the field size) polynomial.
- **Polynomial-Size Circuit Tampering:** each tampered codeword is produced by circuit of size n^c where c is any constant (the efficiency of our code depends on c). This result

assumes that E is hard for exponential size nondeterministic circuits (all other results are unconditional).

We stress that our constructions are efficient (encoding and decoding can be performed in uniform polynomial time) and (with the exception of the last result, which assumes strong circuit lower bounds) enjoy unconditional, statistical security guarantees. We also illuminate some potential barriers to constructing codes for more complex computational classes from simpler assumptions.

Table of Contents

Acknowledgements

I am deeply indebted to my advisor, Tal Malkin, whose unwaivering encouragement, friendship, and advice has been invaluable over the years. Tal's perspectives and insights on matters both technical and non-technical have had a profound impact on me. My life has undergone a dramatic transformation since Tal entered it, and I cannot imagine the past years without her support and engagement.

Not so long after meeting Tal, she sent me off to the Simons Institute's Cryptography Semester as a ward of her brilliant former student, Dana Dachman-Soled. This semester had a tremendous impact on me and I am grateful to the organizers and participants for this, but particularly Tal Rabin. I entered knowing virtually nothing of cryptography, and three months later I left still knowing nothing, but an incredible toolkit, perspective, and connections made there have had an immeasurable impact on my life since.

It was there, with Dana and Tal and Dana's former student Mukul Kulkarni, that I began exploring the topic of this thesis after spirited talk by Yevgeniy Dodis. Since then, I have continued to have the immense pleasure of working with the three of them on these topics. Almost all of the work in this thesis is joint with Dana, an incredibly creative, thoughtful, and humble researcher who has been a great role model and mentor to me.

Also at Simons I had the incredible fortune to prepare cocktails with Elette Boyle and meet with Alon Rosen. The two of them, with Tal Moran, invited me to visit them in the FACT Center at IDC Herzilya for eight months. I returned to the states two and half years later with my partner, Hallel, and our son, Neta. Through and alongside each of them I have grown in different ways: learning about new areas, navigating professional situations, and much much more. But I am especially grateful for their shared dynamic (that also infected their many visitors): nearly every day was exciting and hilarious. Their friendship has been integral to my development.

I would also like to thank my other incredible collaborators. Those who coauthored the work in this thesis: Siyao Guo, Daniel Wichs, Eshan Chattopadhyay, Jyun-Jie , and Julian Loss. And those who have collaboratored on other projects: Ran Cohen, Tianren Liu, Manuel Sabin, Pierre Meyer, Tim Randolph, Apoorvaa Deshpande, Justin Holmgren, Brent Carmer, Nichole Schimanski, Akshay Degwekar, and Vinod Vaikuntanthan.

I would like express my particular gratitude to Prashant Vasudevan and Lisa Kohl. I look forward to our continued conversations.

I have had no shortage of great mentors. In addition to those above I would like to particularly thank Hoetech Wee, Yuval Ishai, Mike Rosulek, Oded Goldreich, Yevgeniy Dodis, and Li-Yang Tan.

I would also like to thank the Columbia Theory group. I had the best office mates: Clément Canonne, Emmanouil Vasileios Vlatakis Gkaragkounis, and Chengyu Lin. I also would like to thank those professors professors who always took the time to answer my questions: Xi Chen, Omri Weinstein, Rocco Servedio, Alexandr Andoni, and Mihalis Yannakakis (whose class encouraged me to pursue TCS in the first place).

I would like to thank my coworkers at Google: especially Mariana Raykova, Tancrède Lepoint, Karn Seth, and Moti Yung. And my future mentors: Huijia Lin and Stefano Tessaro.

I am grateful for the support of my parents, my sisters, Adele and Claire, and my brother, David during these past years. I would like to thank all my friends, but especially Susan Thomases and Thomas Bettridge.

And finally and most significantly, I acknowledge Hallel and Neta. Neta, you light up every day for me. And Hallel, I would not be who I am today without you.

This work was supported in part by an IBM PhD Fellowship and a grant from the Check Point Institute for Information Security.

Dedication

For Hallel and Neta.

Publication History

The results of Section 3.4, Section 5.4, and Chapter 10 previously appeared in [1]. The results of Chapter 5 (with the exception of Section 5.4) previously appeared in [2]. The results of Chapters 4 and 6 previously appeared in [3]. The results of Chapter 7 previously appeared in [4]. The results of Chapter 8 appeared previously in [5]. The results of Chapter 9 are part of a larger unpublished manuscript, [6].

Chapter 1: Introduction

In this thesis, we investigate the feasibility of protecting memory against efficiently computable tampering attacks, for various notions of efficiency. Very roughly, memory is said to be tamper-resilient if no attack can 'mal' the memory contents into some related information. In particular, any attack either (a) does nothing and the original memory can be recovered, or (b) completely destroys the memory contents so that what remains bears no relation to the original contents. Importantly, whether case (a) or (b) occurs should additionally be independent of the protected information.

In particular, we show how to construct explicit non-malleable codes [7] against a variety of "natural" tampering classes that correspond to simple models of computation. In contrast to other domains of non-malleable cryptography [8], non-malleable codes must achieve their guarantees without relying on secret keys. This is important when the secret keys themselves are what need to be protected from tampering. [9, 10]

However, in this thesis our focus is not practical attacks, but theoretical feasibility. With this in mind, let us take a step back to and briefly contextualize tamper-resilience in the theory of computation, before describing non-malleable codes and our results.

1.1 Hardness vs. Tamper-Resilience?

The fundamental goal of the theory of computer science is to understand what is possible and impossible for different computational devices. The negative thrust of this work, computational lower bounds, seeks to find explicit tasks that these devices fail to carry out. An explicit worst-case lower bound specifies a function that no machine of a given type, can correctly evaluate on all inputs. A stronger form of such a result, an explicit average-case lower bound (or correlation

bound) specifies a function and a distribution such that no machine of a given type (from a given computational class) can accurately predict the value of the function, when given an input drawn from the distribution.

> *Explicit average-case circuit lower bounds.* There exists an efficiently computable function g and an efficiently samplable distribution X such that for all f in some computational class $\mathcal{F}$,
>
> $$f(X) \not\approx g(X)$$
>
> (We take $\approx$ to denote statistical closeness (small total variation distance between the two distributions.))

Over the past half century, many remarkable results of this form have been proven. Unfortunately, for many classes of interest we cannot prove such theorems, or even the weaker worst-case variants. In many cases, there seem to be sharp barriers beyond which our current technical machinery fails to say anything meaningful.[1]

In the present thesis, we do not try to expand the domains for which we know lower bounds, but instead we attempt to sharpen our existing hardness techniques to yield stronger and more useful guarantees. In particular, we focus on the task of tamper-resilience or non-malleability.[2] Before we rigorously introduce the central focus of our study, non-malleable codes, we instead what it means for a *function* to be non-malleable with respect to a distribution which captures the core difficulty in tamper-resilience with syntax similar to that of lower bounds.[3] Roughly, a function, g, is non-malleable for a tampering class, $\mathcal{F}$, with respect to a distribution, X, if the output of the function on a sample tampered by some machine in the class, is effectively *uncorrelated* with

[1] There are numerous examples in the literature, but one considered in this thesis is that of small-depth circuits. We can prove very strong average-case lower bounds against super-polynomial size circuits of depth $c \log n / \log \log n$ with unbounded fan-in for some specific constant c. If we could prove such lower bounds for arbitrary c we could prove lower bounds against NC^1, the class of bounded fan-in polynomial size circuits with depth $O(\log n)$. This class is remarkably rich, corresponding to very efficient parallel computation. Determining the limitations of this class is a long standing open problem.

[2] This is by no-means the only "harder" dimension to consider. To name just a few alternative dimensions (for which we know some connection to lower bounds): learning a class, fooling a class with a pseudorandomness, hardness of compression for a class.

[3] A more stronger variant of the notion we are describing was defined in [11].

the output of the function on the untampered sample. In this simplified presentation, we ensure feasibility by requiring the tampering function to always change something about X (otherwise the tampered output of g will be identical to the untampered output and clearly correlated). To make this task non-trivial, we additionally require $g(X)$ to have some entropy (otherwise, any constant function g would be non-malleable).

> *Explicit non-malleability.* There exists an efficiently computable function g and an efficiently samplable distribution X such that for all f in some computational class $\mathcal{F}$ such that f has no fixed points ($\forall x, f(x) \neq x$)
>
> $$g(X), g(f(X)) \approx g(X'), g(f(X)),$$
>
> where X' is identically distributed to and independent of X, and $f(X) = f(X')$ with probability bounded away from 1.

To illustrate why we believe this task is more difficult, consider the example of Parity and the class of constant depth circuits of polynomial size. Now classic work shows that it is hard for any polynomial size circuit of constant depth to predict the Parity of a uniform random string with reasonable advantage. [12] However, if we take g to be Parity and X to be uniform, we have not achieved non-malleability. Consider the n-bit input, n-bit output circuit, f, that flips the first bit. Obviously this circuit has constant depth and polynomial size. It also clearly violates non-malleability. For any string x, $g(x) = g(f(x)) \oplus 1$. On the other hand, $g(X'), g(f(X))$ is simply two uniform independent bits. So, intuitively it seems non-malleability is at the very least distinct from average-case hardness. Looking ahead, in Section 5.5 we will show classes for which non-malleability is strictly impossible, but average-case hardness is trivial.

To complement this observation, we also observe that for many natural classes explicit non-malleability implies explicit average-case lower bounds. In particular, if g is non-malleable for $\mathcal{F}$ with respect to X, then g must be hard-on-average for $\mathcal{F}$ with respect to X. To see this suppose not, then there is some $f \in \mathcal{F}$ which on input X can predict $g(X)$ with some advantage. Define f'

to simply take f's guess for $g(X)$, b, and output some $\tilde{x}$ such that $g(x) = 1 - b$. If f can predict g well, then $g(X) = 1 - g(f(X))$ with reasonably high probability. Whereas, $g(X')$ is independent of $g(f(X))$, and required to have some entropy of its own. We go through this argument in more detail in Section 3.3 for the case of non-malleable codes.

In particular, this tells us that if we want to prove non-malleability against a class, we have to prove average-case circuit lower bounds against the class. Given that we only know how to prove such lower bounds for specific classes, this gives us some guidelines as to where we might reasonably hope to prove non-malleability. Alternatively, in the absence of proving a circuit lower bound unconditionally, we must assume such lower bounds. In this thesis, we pursue both paths. Ultimately, our investigations leave the impression that proving non-malleability is, in general, strictly harder than proving circuit lower bounds, though this is by no means conclusive. Whether there is an alternative (natural) notional that indeed minimally captures non-malleability remains elusive.

That said, our methods often rely heavily on those used to prove circuit lower bounds (See Chapter 7 in particular). We interpret this as attesting to the strength of these tools, and not evidence of an equivalence.

1.2 Non-Malleable Codes

Motivated by applications in tamper-resilient cryptography, non-malleable codes were first introduced by Dziembowski, Pietrzak, and Wichs [7] as an extension of error-correcting codes that give meaningful guarantees even when every bit of a codeword may be altered. To define the non-malleability of an encoding scheme $(\mathrm{Enc}, \mathrm{Dec})$ for a class of tampering functions $\mathcal{F}$, consider the following experiment for any $f \in \mathcal{F}$: (1) encode a message x via Enc, (2) tamper the resulting codeword with f, and (3) decode the tampered codeword with Dec. Roughly, $(\mathrm{Enc}, \mathrm{Dec})$ is non-malleable if $\tilde{x} = \mathrm{Dec}(f(\mathrm{Enc}(x)))$ is either completely unrelated to the original message x or identical to x, for any x. In particular, the outcome of the experiment should be simulatable without any knowledge of x. Of course if the adversary does nothing the experiment will output x, which

the simulator does not know. This is handled by allowing the simulator to output a special symbol "same" to indicate that the message was left unchanged.

> *Non-Malleable Codes.* (See Def. 2) A pair of (randomized) algorithms (Enc, Dec) is said to be non-malleable against a tampering class $\mathcal{F}$ if for every message x and every tampering function $f \in \mathcal{F}$ there exists a distribution over messages and the special symbol `same`, D_f (the simulator for f), such that
>
> 1. **(Correctness)** $\mathrm{Dec}(\mathrm{Enc}(x)) = x$.
> 2. **(Tamper-resilience)**

$$\mathrm{Dec}(f(\mathrm{Enc}(x))) \approx \left\{ \begin{array}{c} y \leftarrow D_f \\ \text{if } y = \texttt{same}, \text{ output } x; \text{ otherwise, output } y \} \end{array} \right\}$$

The initial work of Dziembowski et al. [7] observed that achieving non-malleability against arbitrary tampering is strictly impossible.[4] On the other hand, they showed that relatively mild restrictions on the size of class of tampering attacks rendered non-malleability feasible. The goal, as in many coding tasks, is to construct an explicit code against as large a class of tampering functions (channels) as possible with best information rate and strongest security guarantees.

Since 2010, a flurry of work has done just that, giving explicit constructions of statistically secure non-malleable codes for specific families of tampering functions and the efficiency of these constructions (in both information rate and computational complexity).

1.3 Our Results: How to Construct Non-Malleable Codes

In this thesis, we will focus on tampering attacks that can be performed in simple computational models. In particular, we construct efficient non-malleable codes for a variety of classes[5].

[4] Consider the generic attack that decodes, tampers, and re-encodes the tampered message.

[5] As these classes are typically defined with respect to a single output, we say a tampering attack, $\boldsymbol{f} : c \mapsto \tilde{c}$, is computable in a class, $\mathcal{F}$, if each tampered symbol, $\tilde{c}_i$ is computable by some $g_i \in \mathcal{F}$ (namely, $\tilde{c}_i = g_i(c)$ for all $i \in [n]$).

We begin with a number of observations that give greater context to our task. We give a simple argument that efficient codes exist for small tampering families and show that non-malleability for polynomial size tampering can be decided in AM. We also flesh out the necessity of circuit lower bounds and observe that for the specific case of tampering functions that are only restricted by the number of bits they can change, there is no benefit to non-malleability over error correction, provided one wants to encode more than one bit.

Bounded communication tampering. This class, also known as leakage resilient split-state, consists of functions that can be represented by two party protocols of the following form. One party, Alice, has left half the codeword, and another, Bob, has the right half. Alice and Bob, upon receiving their inputs, follow a deterministic protocol where they communicate a bounded number of bits. After communicating, Alice outputs a tampered left half-codeword and Bob outputs a right half-codeword.

In Chapter 4, we construct show how to compile any split-state code into a code for bounded-communication tampering where Alice and Bob can communicate up to $1/4$th of the length of their inputs, with just a constant factor loss in rate (codewords grow by a constant factor), and small (additive inverse exponential) loss in security.

Prior to our work, constructions for this tampering class were known [13, 14], but they could not handle as much communication. Additionally, our construction is simpler, generic, and yields better rate and security.

Local tampering A function is said to have locality ℓ if each output bit can be written as a function of at most ℓ input variables.

In Chapter 5, we construct explicit non-malleable codes for tampering functions with locality n^{δ}, where δ is any constant strictly less than 1. Our methods additionally yield inefficiently computable non-malleable codes for any locality locality $o(n/\log n)$. (The sets of relevant variables and the function evaluated on them can be distinct for each output.)

Our code deviates from the original definition of non-malleability in that we allow D to be ran-

domized. It is not known if the two notions are equivalent. Subsequent to our work, Cheraghchi and Li [15] constructed non-malleable codes with deterministic decoding that are resilient to tampering with locality $o(\sqrt{n})$. Another code with deterministic decoding for locality $o(\sqrt[3]{n})$ is given in the published version of this Chapter [2, 4].

Another follow up work by Gupta, Maji, and Wang [16] showed how to compile our codes to get explicit rate 1[6] non-malleable codes for locality $\gamma \log n$, for any constant $\gamma < 1$.

Decision tree tampering. We say a tampering attack is computable by depth d decision trees if each tampered output bit is a function of at most d, adaptively-made, bit probes to the input. (The adaptive probe sequences and output decisions corresponding to each output bit can be distinct.)

In Chapter 6, we construct explicit non-malleable codes for tampering by decision trees of depth $n^{1/4-o(1)}$.

Small-depth circuit tampering. We say a tampering attack is computed by a depth d circuit of size s if each tampered output bit is produced by a standard basis (unbounded fan-in AND/OR/NOT) circuit of depth d and size s. (The circuits corresponding to each output can be distinct.)

In Chapter 7, we construct explicit non-malleable codes for tampering by circuits of depth $d \leq c_1 \log n / \log\log n$ and size $\exp(n^{c_2/d})$ where $c_1, c_2 \in (0, 1)$ are constants. Constructing an explicit non-malleable code for circuits of even just depth $d = O(\log n / \log\log n)$ would imply a separation of P and NC^1, a long standing open problem.

Prior to our work, Cheraghchi and Li [15] constructed non-malleable codes for circuits of constant depth and polynomial size, but their code yielded codewords were subexponentially long in the message length (i.e. $n = 2^{\Omega(\sqrt{k})}$, where k is the message length and n is codeword length).

Low-degree polynomial tampering (in large fields). In contrast to prior classes which considered a binary alphabet, here we assume our alphabet corresponds to some large prime field $\mathbb{F}_q$. An attack over such an alphabet is computable by degree-d polynomials if each tampered output

[6] A code has rate 1, $\lim_{n\to\infty} k/n(k) = 1$ where k is the message length and $n(k)$ is the length of codewords for messages of length k.

symbol is the output of some polynomial of degree d. (The polynomials corresponding to each output can be distinct.)

In Chapter 8, we construct explicit non-malleable codes for tampering by polynomials of degree $d = q^{1/4-o(1)}$. Our result immediately implies a non-malleable code against tampering by very small arithmetic circuits. Along the way we show how to construct seedless non-malleable extractor (See Section 2.3) for these tampering classes.

Prior to our work, Cheraghchi and Li [15] constructed non-malleable codes for degree 1 (affine) tampering. Additionally, in the published version of this Chapter [5] it additionally shown how to combine our extractor with a technique of Lin et al. [17] to construct a non-malleable secret sharing scheme resilient to low-degree tampering. Very roughly, a non-malleable secret sharing scheme is a threshold secret sharing scheme with the guarantee that if the shares are (jointly) tampered by functions in the class, then any reconstruction recovers either the original secret or something independent of the secret.

Polynomial-size circuit tampering. For any constant c, we say a tampering attack is computed by a size n^c circuit, if the tampered output is produced by some circuit with n inputs and n outputs and at most n^c fan-in 2 gates from the standard basis (AND/OR/NOT).

In Chapter 9, we show that for any constant c, there exists an explicit non-malleable code for tampering by n^c circuits that is secure assuming lower bounds on nondeterminsitic circuits. Our assumption is taken from the derandomization literature.

Prior to our work, no fully explicit codes for these classes were known. Constructions were known assuming a common random string. [18, 19] Additionally, computationally-secure constructions were known under strong cryptographic assumptions [20, 21]. Both of these latter works required assumptions that we currently can only provably instantiate with a random oracle. Thus, they only yield explicit constructions under heuristic assumptions about concrete hash functions.

We conclude this thesis by making the general intuition that non-malleability is harder than circuit lower bounds a bit more formal: we rule out the possibility of constructing non-malleable

codes from minimal average-case hardness bounds in a black-box manner. We also give an unconditional separation between the two notions: ruling out the possibility of non-malleable codes against $(n-1)$-local functions (which gives an explicit separation between lower bounds and non-malleability because these functions fail to compute parities with any advantage). However, we note that under cryptographic assumptions a class of such implications is known in the computational security regime.[7] [22, 20]

1.3.1 Relations between the tampering classes considered.

As far as we know, the classes considered above are incomparable for the parameter regimes in which we can prove security. However, relationships are known between other parameter regimes (which we will exploit). We briefly discuss what is known about the relations between these classes.

Bounded-communication vs local functions. Bounded-communication tampering is capable of arbitrary attacks that are independent on the left and the right. Such an attack can have locality up to $n/2$, which is beyond what our results can handle. The locality 1 attack that simply copies the left hand codeword to the right hand side and the right hand side to the left requires n communication. For similar reasons bounded-communication tampering is incomparable to all other classes we consider.

Local functions vs decision trees. Both local functions and decision trees (or rather, forests) have outputs that depend on only a few input variables. However, for local functions these variables are chosen statically, independent of the actual input, whereas decision trees may choose the variables they query adaptively based on the values returned. As such, every t-local function is a t-depth decision tree. On the other hand, by considering all possible queries in advance, every t-depth decision tree is a 2^t-local function. However, since $t \leq n$ (the input length), this is only

[7] Computational security means real tampering experiment can be simulated in a manner that is indistinguishable to *polynomial time* distinguishers, but not necessarily indistinguishable to arbitrary (inefficient) statistical tests.

meaningful for $t \leq \log n$.

We can efficiently handle locality n^δ for any constant $\delta < 1$ and decision tree depth $n^{1/4-o(1)}$. These classes are incomparable for $\delta \geq 1/4$.

Small-depth circuits vs decision trees. Next, we note that every depth t decision tree can be expressed as disjunction of clauses, each containing at most t variables (corresponding to each accepting path). So, every decision tree can be expressed as depth-2 circuit of size at most $t^2\binom{n}{t} \leq t^2(en/t)^t$ (for $t \leq n/2$).

On the other hand, decision trees of depth $d < n$ fail to compute even a single conjunction of all variables. However, in some sense, Håstad's switching lemma tells us that, on average, small-depth circuits exhibit local behavior like that of low-depth decision trees. [12]

Low-degree polynomials vs small-depth circuits. There is a syntactic mismatch due to the different alphabets of these classes. That said, Razborov and Smolensky showed that small-depth circuits, while they can be approximated by low-degree polynomials, fail to perform even elementary arithmetic. [23, 24, 25] Conversely, even a single conjunction has too many 0s to be computed by a low-degree polynomial. [26]

Polynomial size circuit tampering vs everything else. The parity function is hard for all the classes above, with the exception of bounded-communication tampering. Yet parity can be computed by linear size circuits. Bounded communication fails to compute inner products, which can in turn be computed by linear size circuits.

On the other hand, the class of n^c circuits contains at most $2^{3cn^c \log n)}$ functions. On the other hand, our other tampering classes are doubly exponential in size ($2^{2^{n^{\Omega(1)}}}$). So by counting arguments, this class fails to compute certain functions in these classes, regardless of the choice of c.

1.3.2 Related work

We discuss a variety of related work on non-malleable codes and their relatives. Our discussion of work on non-malleable codes is broken into two parts: the first focuses on fully explicit codes with *statistical* security guarantees, the second focuses on relaxed definitions.

Non-malleable codes. Non-malleable codes were introduced by Dziembowski, Pietrzak and Wichs [7] Various subsequent works re-formulated the definition [13], or considered extensions of the notion [27, 28, 29, 30]. The original work of [7] presented a construction of non-malleable codes against bit-wise tampering, and used the probabilistic method to prove the existence of non-malleable codes against tampering classes $\mathcal{F}$ of bounded size (this result gives rise to constructions for the same tampering classes $\mathcal{F}$ in the random oracle model). A sequence of works starting from the work of Liu and Lysyanskaya [31] presented constructions of non-malleable codes secure against split-state tampering. The original work and some subsequent works [32, 33] required an untamperable common reference string (CRS) and/or computational assumptions. Other works removed these restrictions and achieved information-theoretic non-malleable codes against split-state tampering with no CRS [34, 13, 35]. Recently, Aggarwal and Obremski [36] constructed a code rate of $\Omega(1)$ for two states. Constructions requiring more than two split-states, and which achieve constant rate, were also given in [37, 38].

Agrawal et al. [39] presented constructions of non-malleable codes against tampering functions that are permutations.

Chattopadhyay and Li [15] gave constructions of information-theoretic, non-malleable codes against affine functions over $\mathbb{F}_2$ and (separately) AC^0 tampering functions. However in contrast to the present work, their AC^0 non-malleable code results in a codeword that is super-polynomial in the message length and so is inefficient.

Finally, Ball et al. [22] constructed non-malleable codes secure against small-space streaming tampering attacks.

Non-malleable codes in other models. A variety of non-malleable codes against complexity-based tampering classes have been constructed in other models. These constructions require either common randomness (CRS), access to a public random oracle (RO), and/or computational/cryptographic assumptions.

In addition to those already mentioned, Faust et.al [19] presented an efficient non-malleable code, in the CRS model, against tampering function families $\mathcal{F}$ of bounded size, improving upon the original work of [7]. Since the size of the CRS grows with the size of the function family, this approach cannot be used to obtain efficient constructions of non-malleable codes against tampering classes that contain circuits of unbounded polynomial size (e.g., AC^0).

Cheraghchi and Guruswami [40] in an independent work showed the existence of information theoretically secure non-malleable codes (with no CRS) against tampering families $\mathcal{F}$ of bounded size (via a randomized construction). However their construction is inefficient for negligible error.

The work of [22] constructs a generic framework for converting correlation bounds into non-malleable codes. They instantiate their framework to construct non-malleable codes both for decision trees and large small-depth circuits. However, this work (a) requires a Common Reference String, and (b) is secure in a game-based model against efficient adversaries, assuming unproven cryptographic assumptions. In fact, one of the assumptions, public key encryption with decryption in AC^0, necessarily limits their error term to be at most $\exp(-\log^{O(1)}(n))$.

A follow-up work, [20], improves upon [22] showing how to remove the Common Reference String, but still does not achieve unconditional or statistical guarantees. A very recent follow up to the follow up improves upon this, but still falls short of statistical guarantees [21]. Both of these works make also rely on assumptions that we only know how to provably instantiate from random oracles.

Faust et.al [41] gave constructions of (a weaker notion of) non-malleable codes against space bounded tampering in the random oracle model.

Chapter 2: Preliminaries

In this chapter, we present a series of definitions and lemmas that will be useful in later chapters. We advise the reader treat to initially skip this chapter and use it as a reference for later chapters, looking up the relevant definitions and propositions as needed.

2.1 Notation

Define $e(x) = e^{2\pi i x}$, where $i = \sqrt{-1}$.

For $n \in \mathbb{N}$, we use $[n]$ to denote the set $\{1, 2, \ldots, n\}$. For non-negative integer k, we use $\binom{[n]}{k}$ denote the set of all subsets of $[n]$ of size k. Let Σ be a set of symbols. For sequence $X = (x_1, \ldots, x_n) \in \Sigma^n$ and $S = \{i_1, \ldots, i_k\} \subseteq [n]$ such that $i_1 < i_2 < \ldots < i_k$, we use X_S to denote the sequence $(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$. For the particular case of $x, y \in \{0,1\}^n$, if they disagree on at least $\varepsilon \cdot n$ indices, we say they are ε-far, otherwise, they are ε-close to each other.

For non-negative integers $\lambda_1, \ldots, \lambda_n$ that sum to 1, and arbitrary distributions $D_1, \ldots, D_n$, we use $\sum_i \lambda_i D_i$ to denote the distribution that places weight $\sum_i \lambda_i D_i(x)$ at the point x.

For any distribution D, let $D(x)$ denote $\Pr[D = x]$, and let $\mathrm{Supp}(D)$ denote the support of D.

For a set Σ, we use Σ^Σ to denote the set of all functions from Σ to Σ. Given a distribution $\mathcal{D}$, $z \leftarrow \mathcal{D}$ denotes sample z according to $\mathcal{D}$. For a set S, we take $z \xleftarrow{u} S$ to denote sampling z uniformly from S.

For two distributions $\mathcal{D}_1$,$\mathcal{D}_2$ over Σ, their statistical distance is defined as $\Delta(\mathcal{D}_1; \mathcal{D}_2) := \frac{1}{2}\sum_{z\in\Sigma} |\mathcal{D}_1(z) - \mathcal{D}_2(z)|$. We sometimes write $\mathcal{D}_1 \approx_\epsilon \mathcal{D}_2$ to denote $\Delta(\mathcal{D}_1; \mathcal{D}_2) \leq \epsilon$. If $\Delta(\mathcal{D}_1; \mathcal{D}_2) \leq \epsilon$ (resp. $\Delta(\mathcal{D}_1; \mathcal{D}_2) > \epsilon$), we say $\mathcal{D}_1$ and $\mathcal{D}_2$ are ϵ-close (resp. ϵ-far).

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

Definition 1 (Restriction)**.** For a vector $v \in \{0, 1, *\}^n$ and a Boolean function $f : \{0,1\}^n \to \{0,1\}^n$

the restriction of f to v, $f \restriction_v$ is defined as $f \restriction_v (x) = f(z)$ where,

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

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

As throughout we are considering multi-output generalizations of functions with a single output (either bit or field element), we say that a multiple output function $f = (f_1, \ldots, f_m)$ is in C if $f_i \in C$ for any $i \in [m]$.

2.2 Non-Malleable Codes

Non-malleable codes were first defined in [7].

Definition 2 (Coding Scheme [42])**.** Let Σ be a finite alphabet set. A *coding scheme*, (E, D), consists of a (possibly randomized) encoding function $\mathsf{E} : \{0,1\}^k \to \Sigma^n$ and a deterministic decoding function $\mathsf{D} : \Sigma^n \to \{0,1\}^k \cup \{\perp\}$ such that $\forall m \in \{0,1\}^k, \Pr[\mathsf{D}(\mathsf{E}(m)) = m] = 1$ (over randomness of E).

We say (E, D) has block length n and message length k.

Our result on Local tampering (Chapter 5) will additionally rely on a randomized decoding function. We say a (E, D) is an encoding scheme with randomized decoding if D is randomized. We note that unlike as is the case for encryption, we do not know how to generically derandomize decoding for non-malleable codes and so these definitions may not be equivalent.

We define the distance of a *randomized* coding scheme by considering the minimum distance of all codes formed as follows: for each message $x \in \{0,1\}^k$ choosing an arbitrary codeword corresponding to that message, $c_x \in \{\mathsf{E}(x;r)\}_{r \in \{0,1\}^*}$. We take the distance of the randomized encoding scheme to be the maximum of all such minimum distances (i.e. the distance of the best sub-code).

Definition 3 (Distance of a Coding Scheme)**.** The *distance* of a (randomized) coding scheme, (E, D), is

$$\max_{\substack{S \subset \{c = \mathsf{E}(x;r) : x \in \{0,1\}^k, r \in \{0,1\}^*\}: \\ \forall x \in \{0,1\}^k, \exists c_x \in S : \Pr[\mathsf{D}(c_x) = x] > 1/2}} \min_{\substack{c_x, c_y \in S: \\ x \neq y \in \{0,1\}^k}} \|c_x - c_y\|_0$$

Note that this definition can be extended to arbitrary alphabets. Moreover, it is clear that the minimum distance of any coding scheme with K messages and codeword space Σ^n is upper bounded by the maximum of the traditional notion of minimum distance in (non-randomized) codes with the same parameters: the minimum distance between codewords from a code (over Σ^n with K distinct code words).

We define a function that will be useful in defining non-malleable codes:

$$\mathrm{copy}(x, y) = \begin{cases} x & \text{if } x \neq \texttt{same} \\ y & \text{if } x = \texttt{same}. \end{cases}$$

Definition 4 (Non-malleable codes)**.** Let Σ be a finite alphabet set. A coding scheme $(\mathrm{Enc}, \mathrm{Dec})$ on alphabet Σ with block length n and message length k is a non-malleable code with respect to a tampering family $\mathcal{G}$ and error ϵ if for every $g \in \mathcal{G}$ there is a random variable D_g supported on $\{0,1\}^k \cup \{\texttt{same}\}$ that is independent of the randomness in Enc, and any message $z \in \{0,1\}^k$, we have

$$\Delta(\mathrm{Dec}(f(\mathrm{Enc}(z))); \mathrm{copy}(D_g, z)) \leq \epsilon$$

We define the rate of a non-malleable code C to be the quantity $\frac{k}{n \log(|\Sigma|)}$.

Dziembowski et al. [43] provided the following convenient characterization of non-malleability for single-bit messages

Lemma 1 (Lemma 2 [43])**.** *Let* (E, D) *be a coding scheme with* $\mathsf{E} : \{0,1\} \to \mathcal{X}$ *and* $\mathsf{D} : \mathcal{X} \to \{0,1\}$*. Let* $\mathcal{F}$ *be a set of functions* $f : \mathcal{X} \to \mathcal{X}$*. Then* (E, D) *is* ε*-non-malleable with respect to* $\mathcal{F}$

if and only if for every $f \in \mathcal{F}$*,*

$$\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1 - b] \leq \frac{1}{2} + \varepsilon,$$

where the probability is over the uniform choice of b *and the randomness of* E*.*

Because this yields such a clean characterization of non-malleability, we will will say that a one-bit encoding scheme is ε-malleable w.r.t. a tampering class $\mathcal{F}$ if there exists $f \in \mathcal{F}$ such that $\Pr_b[\mathrm{D}(f(\mathrm{E}(b))) = 1 - b] \geq 1/2 + \varepsilon$.

We sometimes prefer to use a simpler, but equivalent, definition based on the following notion of non-malleable reduction by Aggarwal et al. [44].

A non-malleable reduction from $\mathcal{F}$ to $\mathcal{G}$ is simply an encoding scheme, $(\mathrm{Enc}, \mathrm{Dec})$ such that for any $f \in \mathcal{F}$ and any message x, the value $\mathrm{Dec}(f(\mathrm{Enc}(x)))$ is statistically close to $g(x)$ for g chose from some distribution over functions in $\mathcal{G}$. In particular a non-malleable code for $\mathcal{F}$ is a non-malleable reduction from $\mathcal{F}$ to the family $\mathcal{G}$ consisting of the identity function and constant functions.

Definition 5 (Non-Malleable Reduction [44])**.** Let $\mathcal{F} \subset A^A$ and $\mathcal{G} \subset B^B$ be some classes of functions. We say $\mathcal{F}$ *reduces to* $\mathcal{G}$, $(\mathcal{F} \Rightarrow \mathcal{G}, \varepsilon)$, if there exists an efficient (randomized) encoding function $\mathrm{Enc} : B \to A$, and an efficient decoding function $\mathrm{Dec} : A \to B$, such that

1. $\forall x \in B, \Pr[\mathrm{Dec}(\mathrm{Enc}(x)) = x] = 1$ (over the randomness of E).

2. $\forall f \in \mathcal{F}, \exists G$ s.t. $\forall x \in B$, $\Delta(\mathrm{Dec}(f(\mathrm{Enc}(x))); G(x)) \leq \varepsilon$, where G is a distribution over $\mathcal{G}$ and $G(x)$ denotes the distribution $g(x)$, where $g \leftarrow G$.

If the above holds, then $(\mathrm{Enc}, \mathrm{Dec})$ is an $(\mathcal{F}, \mathcal{G}, \varepsilon)$*-non-malleable reduction*.

Definition 6 (Non-Malleable Code [44])**.** Let $\mathrm{NM}_k \subseteq \{f : \{0,1\}^k \to \{0,1\}^k\}$ denote the set of *trivial manipulation functions* on k-bit strings, consisting of the identity function $\mathrm{id}(x) = x$ and all constant functions $f_c(x) = c$, where $c \in \{0,1\}^k$. $(\mathrm{Enc}, \mathrm{Dec})$ defines an $(\mathcal{F}_{n(k)}, k, \varepsilon)$*-non-malleable*

code, if it defines an $(\mathcal{F}_{n(k)}, \mathrm{NM}_k, \varepsilon)$-non-malleable reduction. Moreover, the rate of such a code is taken to be $k/n(k)$.

The following useful theorem allows us to compose non-malleable reductions.

Theorem 1 (Composition [44])**.** *If* $(\mathcal{F} \Rightarrow \mathcal{G}, \varepsilon_1)$ *and* $(\mathcal{G} \Rightarrow \mathcal{H}, \varepsilon_2)$*, then* $(\mathcal{F} \Rightarrow \mathcal{H}, \varepsilon_1 + \varepsilon_2)$.

Definition 7 (Alternative-Non-Malleability [7])**.** Let $\mathcal{F}$ be a family of tampering functions. We say that a coding scheme $(\mathrm{Enc}, \mathrm{Dec})$ is ϵ-alternative-non-malleable with respect to $\mathcal{F}$ if for any $m_0, m_1 \in \{0,1\}^k$ and any $f \in \mathcal{F}$, we have:

$$\mathrm{AltNM}^{f,\mathrm{Enc},\mathrm{Dec}}_{m_0,m_1}(0) \approx_\epsilon \mathrm{AltNM}^{f,\mathrm{Enc},\mathrm{Dec}}_{m_0,m_1}(1)$$

where we define the two experiments by

$$\mathrm{AltNM}^{f,\mathrm{Enc},\mathrm{Dec}}_{m_0,m_1}(b) := \left\{ \begin{array}{c} c \leftarrow \mathrm{Enc}(m_b), \tilde{c} \leftarrow f(c), \tilde{m} = \mathrm{Dec}(\tilde{c}) \\ \text{Output } \texttt{same} \text{ if } \tilde{m} \in \{m_0, m_1\}, \text{ and } \tilde{m} \text{ otherwise.} \end{array} \right\}$$

Lemma 2 ([7])**.** *If* $(\mathrm{Enc}, \mathrm{Dec})$ *is* ϵ*-alternatively-non-malleable with respect to* $\mathcal{F}$ *for* k*-bit inputs, then* $(\mathrm{Enc}, \mathrm{Dec})$ *is* $(\epsilon + 2^{-k})$*-non-malleable with respect to* $\mathcal{F}$.

If $(\mathrm{Enc}, \mathrm{Dec})$ *is* ϵ*-non-malleable with respect to* $\mathcal{F}$ *for* k*-bit inputs, then* $(\mathrm{Enc}, \mathrm{Dec})$ *is* 2ϵ*-alternatively-non-malleable with respect to* $\mathcal{F}$.

2.3 Seedless Non-Malleable Extractors

We introduce and discuss a useful ingredient for building non-malleable codes (and more).

Informally, a *randomness extractor* is a deterministic algorithm that produces nearly uniform bits of randomness from defective sources of randomness. Before defining a randomness extractor formally, we first define the notion of min-entropy that is typically used as a measure of the quality of a source:

Definition 8 (Min-entropy and (n, k)-sources)**.** Let X be a distribution on $\{0,1\}^n$. The min-entropy of X, denoted by $H_\infty(X)$, is defined as $\min_x(\log(1/\Pr[X = x]))$.
An (n, k)-source is a distribution on $\{0,1\}^n$ with min-entropy at least k.

We say such a source has entropy deficiency $n - k$.

Definition 9 (Extractor)**.** Let $\mathcal{X}$ be a family of sources on $\{0,1\}^n$. A function $\mathsf{Ext} : \{0,1\}^n \to \{0,1\}^m$ is called an extractor for the family $\mathcal{X}$ with error ϵ if for any $X \in \mathcal{X}$,

$$\Delta(\mathsf{Ext}(X); \mathrm{U}_m) \leq \epsilon,$$

where U_m is the uniform distribution over $\{0,1\}^m$.

It turns out that there cannot exist an extractor that works for the family of distributions on $\{0,1\}^n$ with min-entropy even $n - 1$. To circumvent this difficulty, a long line of work has focused on extracting from a weak source X assuming access to a short independent seed Y. Such extractors are called *seeded extractors* (See Def. 17) [45] and we now have almost optimal constructions of such extractors [46, 47]. Another successful line of research focused on extracting random bits assuming more structure on the source X. Such extractors are called as *seedless extractors*. Examples include assuming that the weak source consists of multiple independent sources [48, 49, 50, 51] (see Def. 18), assuming that the source is supported on an affine subspace [52, 53] or an algebraic variety [54], or even simply assuming that there are some unknown coordinates of the source that are uniform and independent [55]. Explicit constructions of seeded and seedless extractors have found numerous applications in complexity theory [56], coding theory [57] and cryptography [58, 59].

Recently, several works studied a a more robust notion of a randomness extractor called *non-malleable extractor*. The main motivations for studying this stronger variant is from applications in cryptography. Surprisingly, explicit constructions of non-malleable extractors have led to improved constructions of standard extractors. As in the case of standard extractors, there are *seeded non-malleable extractors* and *seedless non-malleable extractors*. The seeded variant was intro-

duced by Dodis and Wichs [60] with applications to the problem of privacy application [58]. The seedless variant of non-malleable extractors was introduced by Cheraghchi and Guruswami [11] with applications to constructions of non-malleable codes.

We focus on the seedless variant of non-malleable extractors. We begin with a simplified variant that only considers tampering functions without fixed points (i.e. $\forall x, f(x) \neq x$).

Definition 10 (Relaxed Seedless non-malleable extractor)**.** Let $\mathcal{X}$ be a family of sources on $\{0,1\}^n$ and $\mathcal{F}$ be a class of tampering functions acting on $\{0,1\}^n$. Further assume that all $f \in \mathcal{F}$ does not have any fixed points. A function $\mathrm{NMExt} : \{0,1\}^n \to \{0,1\}^m$ is defined to be a relaxed non-malleable extractor with respect to $\mathcal{X}$ and $\mathcal{F}$ with error ϵ if the following hold: for any $X \in \mathcal{X}$ and $f \in \mathcal{F}$, we have

$$\Delta(\mathrm{NMExt}(X), \mathrm{NMExt}(f(X)); \mathrm{U}_m, \mathrm{NMExt}(f(X))) \leq \epsilon.$$

An informal way of interpreting the above definition is as follows. Let X be a source from the family $\mathcal{X}$. The distribution $X' = f(X)$ represents the tampered distribution, where $f \in \mathcal{F}$ (note that $X' \neq X$). The task of the non-malleable extractor NMExt is to remove the correlation between the random variables X and X' (which are clearly dependent). With this intuition we give the full definition:

Definition 11 (Seedless non-malleable extractors)**.** Let Σ be a finite alphabet set, $\mathcal{G}$ be a class of tampering functions $\Sigma^n \to \Sigma^n$ and $\mathcal{X}$ be a class of distributions over Σ^n. A function $\mathrm{NMExt} : \Sigma^n \to \{0,1\}^m$ is called a seedless non-malleable extractor that works for $\mathcal{X}$ with respect to $\mathcal{G}$ with error ϵ if for every distribution $X \in \mathcal{X}$ and every tampering function $g \in \mathcal{G}$, there exists a random variable D_g on $\{0,1\}^m \cup \{\mathtt{same}\}$ that is independent of X, such that

$$\Delta(\mathrm{NMExt}(X), \mathrm{NMExt}(g(X)); \mathrm{U}_m, \mathrm{copy}(D_g, \mathrm{U}_m)) \leq \epsilon.$$

Cheraghchi and Guruswami [11] showed that relaxed seedless non-malleable extractors imply

seedless non-malleable extractors with some loss in parameters for the specific case of independent sources and with independent tampering. Their transformation generalizes, but in general the class of sources may change as well (depending on the tampering class considered). We leave the details as an exercise for the reader if they are interested.

Chattopadhyay and Zuckerman [37] gave explicit constructions of seedless non-malleable extractors assuming X consists of 10 independent sources, and each source is arbitrarily tampered. This was improved by Chattopadhyay, Goyal and Li [61] to construct seedless non-malleable extractors for 2 independent sources and later improved upon by Li [35]. Chattopadhyay and Li [15] constructed a seedless non-malleable extractor against the class of affine functions. In another work, Chattopadhyay and Li [14] constructed seedless non-malleable extractors when the source X consists of 2 independent sources that are interleaved in an unknown way. They also consider some generalizations such as composition of linear tampering and partitioned tampering.

Non-malleable codes via seedless non-malleable extractors

Cheraghchi and Guruswami [11] established the following connection between non-malleable codes and seedless non-malleable extractors.

Theorem 2. *Let Σ be some finite alphabet set. Let* $\mathrm{NMExt} : \Sigma^n \to \{0,1\}^m$ *be a polynomial time computable seedless non-malleable extractor that works for uniform distribution with respect to a class of tampering functions $\mathcal{G}$ acting on Σ^n. Suppose there is a sampling algorithm* Samp *that on any input $z \in \{0,1\}^m$ runs in time* $\mathrm{poly}(n, \log|\Sigma|)$ *and samples from a distribution that is δ-close to uniform on the pre-image set* $\mathrm{NMExt}^{-1}(s)$.

Then there exists an efficient construction of a non-malleable code on alphabet Σ with block length n, relative rate $\frac{m}{n}$, error $2^m\epsilon + \delta$ with respect to the tampering family $\mathcal{G}$.

Given such an invertible non-malleable extractor, the non-malleable code for $\mathcal{G}$ is defined as follows: Any message $v \in \{0,1\}^m$ is encoded as $\mathrm{Samp}(v)$. The decoding of a codeword $c \in \Sigma^n$ is $\mathrm{NMExt}(c) \in \{0,1\}^m$.

2.4 Split-State Tampering

We will define each tampering class in the relevant section, however many of our results build on non-malleable codes for the following class.

2.4.1 Split-State Tampering

Most of our results build upon non-malleable codes for so-called "split-state" tampering.

Definition 12 (Split-State Model [7])**.** The *split-state model*, SS_k, denotes the set of all functions:

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

Split-State Tampering is perhaps the most extensively studied form of tampering in the literature, and many constructions of non-malleable codes are known for this class. We highlight three of them here.

Theorem 3 ([36])**.** *There is an explicit split-state non-malleable code rate* $\Omega(1)$ *and error* $\exp(-n^{\Omega(1)})$.

Theorem 4 ([35])**.** *There is an explicit split-state non-malleable code with rate* $\Omega(\log\log n/\log n)$ *and error* $\exp(-\Omega(n\log\log n/\log n))$.

Theorem 5 ([34, 62])**.** *There is an explicit split-state non-malleable code with rate* $\Omega(1/\mathrm{poly}(n))$ *and error* $\exp(-n^{\Omega(1)})$.

We will also use some results of Li on seedless non-malleable extractors. We follow the literature and refer to (relaxed) seedless non-malleable extractors for 2 independent sources and split-state tampering as non-malleable two-source extractors.

Theorem 6 ([35])**.** *There is a constant* γ *such that there exists explicit non-malleable two-source extractors for* $(n, (1-\gamma)n)$*-sources that outputs* $\Omega(n)$ *bits with error* $2^{-\Omega(n/\log n)}$.

A (relaxed) non-malleable two-source non-malleable extractor, $\text{NMExt} : \{0,1\}^{2n} \to \{0,1\}^m$ for (n,k)-sources is said to be strong, with error ϵ, if for every independent pair of (n,k)-sources X, Y and every pair of tampering functions $f : \{0,1\}^n \to \{0,1\}^n$ and $g : \{0,1\}^n \to \{0,1\}^n$,

$$X, \text{NMExt}(X,Y), \text{NMExt}(f(X), g(Y)) \approx_\epsilon X, \mathcal{U}_m, \text{NMExt}(f(X), g(Y)).$$

Theorem 7 ([35])**.** *If* $\text{NMExt} : \{0,1\}^{2n} \to \{0,1\}^m$ *is a two-source non-malleable extractor with error* ϵ *and* (n,k)*-sources, then for any* $k' \geq k$, NMExt *is a strong two-source non-malleable extractor for min-entropy* k' *with error* $2^{2m}(\epsilon + 2^{k+1-k'})$.

So, there is a constant γ *such that there exists explicit strong non-malleable two-source extractors for* $(n, (1-\gamma)n)$*-sources that outputs* $\Omega(n)$ *bits with error* $2^{-\Omega(n/\log n)}$.

2.5 Elements of Complexity Theory

We take E to denote the class of languages decidable in deterministic time $2^{O(n)}$.

We say a circuit has size s if it contains at most s gates. We say it has depth d if every path from an input wire to any output gate (where edges are directed out along gate outputs and in along gate inputs) is of length at most d. A circuit family, $\{C_n\}_{n\in\mathbb{N}}$, is a collection of circuits such that C_n takes inputs of length n.

NC^0denotes the class of circuit families comprised of fan-in 2 AND/OR gates and NOT gates of polynomial size (in the input length) and constant depth.[1] AC^0denotes the class of circuit families comprised of unbounded fan-in AND/OR gates and NOT gates of polynomial size and constant depth. NC^0denotes the class of circuit families comprised of fan-in 2 AND/OR gates and NOT gates of polynomial size and $O(\log n)$ depth. NC denotes the class of circuit families comprised of fan-in 2 AND/OR gates and NOT gates of polynomial size and polylogarithmic depth.

A *nondeterministic circuit* C is a circuit with additional "non-deterministic" inputs, in addition

[1] If one is defining fast parallel algorithms, one would like NC^0(and NC more generally) to be uniform consisting solely of circuit families that can be generated by a polynomial time (or, often, logarithmic space) deterministic turing machine (on input 1^n, the TM outputs a description of C_n). However, here $\mathsf{NC}^0$and NC will play an adversarial role, so we do not impose this restriction.

to the usual inputs. We say C evaluates to 1 on x if and only if the exists an assignment, w, the non-deterministic input wires such that the circuit, evaluated deterministically on input (x, w) outputs 1.

Promise problems generalize the concept of languages that give a better handle on semantic complexity classes. A promise problem, Π, consists of a set of Yes instances, Π_Y, and a disjoint set of No instances, Π_N. A machine is considered to decide Π if on input x *promised* to be in $\Pi_Y \cup \Pi_N$ it accepts x if and only if $x \in \Pi_Y$. In other words, the machine should accept Π_Y and reject Π_N, but can behave arbitrarily elsewhere.

Interactive Proofs (IP) correspond to interactive Turing Machines where a Prover and Verifier interact and the Prover (Merlin) is computationally unbounded while the Verifier (Arthur) is (randomized) and computationally efficient (in the input length). Typically, Arthur is required to be probabilistic polynomial time, but here we occasionally allow Arthur non-deterministic advice. An IP is said to (ϵ, δ)-recognize a promise problem, $\Pi = (\Pi_Y, \Pi_N)$, if it satisfies: (ϵ-Completeness) there exists a Prover such that for all $x \in \Pi_Y$ Arthur accepts with probability at least ϵ, (δ-Soundness) for any Prover and any $x \in \Pi_N$ Arthur accepts with probability at most δ. An IP is said to be constant round if Arthur and Merlin exchange a constant number of messages. The completeness/soundness gap of an (ϵ, δ)-IP is $\epsilon - \delta$.

An IP is said to be an AM-protocol (AM/poly) if the protocol follows the following format: (1) Arthur sends a uniformly random message, (2) Merlin responds, (3) Arthur accepts or rejects as a deterministic polynomial time function (with non-uniform advice) of the transcript. An IP is said to be an MA-protocol (MA) if the following format: (1) Merlin sends a proof, (3) Arthur accepts or rejects as a randomized polynomial time function of the transcript. (Note this class is only distinct from NP if Arthur is uniform.)

Lemma 3 ([63, 64, 65])**.** *For any polynomial $s(n)$, there exists a polynomial $s'(n)$ such that the following holds.*

For any $n \in \mathbb{N}$, if $\Pi = (\Pi_Y, \Pi_N)$ is a promise problem with a constant-round private-coin interactive proof where the verifier runs in deterministic time $s(n)$ with $s(n)$ bits of non-uniform

advice with a gap between soundness and completeness of $1/s(n)$ *on inputs of length n, then* Π *is decided by a nondeterministic circuit of size* $s'(n)$ *on inputs of length n.*

2.5.1 Distributional Problems

Definition 13 (Distributional Problem)**.** A *distributional problem* is a decision problem along with ensembles $(\Psi = \{\Psi_n\}_{n=1}^{\infty}, L = \{L_n\}_{n=1}^{\infty})$ for $n \in \mathbb{N}$, where Ψ_n is probability distribution over $\{0,1\}^n$. The decision problem is to decide whether $s \in L_n$ where, $s \leftarrow \Psi_n$. For a string $s \in \{0,1\}^n$, we define $L(s)$ to output 1, if and only if $s \in L_n$.

We say distributional problem $(\Psi = \{\Psi_n\}_{n=1}^{\infty}, L = \{L_n\}_{n=1}^{\infty})$ is in P if $L \in \mathsf{P}$. We say it is efficiently samplable if there is a randomized poly-time algorithm that on input 1^n samples Ψ_n.

Definition 14 ($\varepsilon(n)$-Hard Distributional Problem)**.** Let $(\Psi = \{\Psi_n\}_{n=1}^{\infty}, L = \{L_n\}_{n=1}^{\infty})$ be a distributional problem, we say that (Ψ, L) is $\varepsilon(n)$-hard for family of boolean circuits $C = \{C_n\}_{n=1}^{\infty}$, if and only if for every circuit $C_n \in C$,

$$\Pr_{x \leftarrow \Psi_n}[C_n(x) = L_n(x)] \leq \frac{1}{2} + \varepsilon(n)$$

If no distribution is specified it assumed to be uniform. In particular, we say a problem or language, L, is $\epsilon(n)$-hard for a class of C if $\forall C \in C, \Pr_{x \leftarrow \mathcal{U}_n}[C(x) = L(x)] \leq \epsilon(n)$ for almost all n.

2.5.2 Hard Cores

Definition 15 (ε-hard-core function)**.** Let $f : \{0,1\}^n \to \{0,1\}$ be a boolean function, and D_S be a distribution over $\{0,1\}^n$ induced by the characteristic function of set $S \subset \{0,1\}^n$[2]. We call f *ε-hard-core* on S for size s (where $n \leq s \leq \frac{2^n}{n}$), if for any boolean circuits C of size at most s

$$\Pr_{x \leftarrow D_S}[C(x) = f(x)] < \frac{1}{2} \cdot (1 + \varepsilon)$$

[2]Characteristic function of set S outputs 1 if the input to the function is in set S.

.

We also present the following theorem from [66]. Note that stronger variants of Impagliazzo's Hard-core set lemma exist in the literature(in fact, in [66]), however this simple variant suffices for us.

Theorem 8 (Theorem 1 [66])**.** *Let $f : \{0,1\}^n \to \{0,1\}$ be boolean function that δ-hard for size s. Also, let $\varepsilon > 0$. Then $\exists$ set $S \subseteq \{0,1\}^n$ and constant c, such that $|S| \geq \delta \cdot 2^n$ and f is ε-hard-core on S for circuits of size $s' \leq c \cdot \varepsilon^2 \cdot \delta^2 \cdot s$.*

Impagliazzo's theorem assumes an expressive computation class, however at the very low end of complexity is not always clear if such a proof will go through. The following property is a sufficient condition to apply the theorem above to a circuit class $\mathcal{F}$.

Definition 16 (Hard Core Set (HCS) Amenable)**.** We say that $\mathcal{F} = \{\mathcal{F}_n\}_{n=1}^{\infty}$ is HCS-Amenable if for any polynomial $p(\cdot)$, it holds that if $C_1, \ldots, C_{p(n)} \in \mathcal{F}_n$ then $\mathsf{MAJ}(C_1, \ldots, C_{p(n)}) \in \mathcal{F}_n$.

2.6 Pseudorandom Objects

2.6.1 Extractors

Definition 17 (Strong Extractors [45])**.** A function $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ is a (k, ϵ) extractor if for every (n, k) source X, Y, $\mathsf{Ext}(X, Y)$ is ϵ-close to Y, U_m where Y is uniformly distributed over $\{0,1\}^d$ and U_m is uniformly and independently distributed over $\{0,1\}^m$. An extractor is explicit if it is computable in polynomial time.

Theorem 9 (Explicit Strong Extractors [46])**.** *For every constant $\alpha > 0$, and all positive integers n, k and all $\epsilon > 0$, there is an explicit construction of a (k, ϵ) extractor $\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^d \to \{0,1\}^m$ with $d = O(\log n + \log(1/\epsilon))$ and $m \geq (1-\alpha)k$.*

Definition 18 (Two Source Extractors [48])**.** A function $2\mathsf{Ext} : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^m$ is a (k, ϵ) two source extractor if for independent source X and sources Y such that $H_\infty(X) + H_\infty(Y) \geq$

k, $(Y, 2\mathsf{Ext}(X, Y))$ is ϵ-close to (Y, U_m) and $(X, 2\mathsf{Ext}(X, Y))$ is ϵ-close to (X, U_m) where U_m is uniformly distributed over $\{0,1\}^m$. An extractor is explicit if it is computable in polynomial time.

Theorem 10 (Explicit Two Source Extractors [48])**.** *For all positive integers m, n such that n is a multiple of m and for all $\epsilon \geq 0$, there exists an efficient $(n + m + 2\log 1/\epsilon, \epsilon)$ 2-source extractor with n-bit sources and m-bit output.*[3]

2.6.2 Binary Ramp Secret Sharing Encoding Schemes, or "Reconstructable Probabilistic Encoding (RPE)"

Reconstructable Probabilistic Encoding (RPE) schemes were first introduced by Choi et al. [67, 68], extending a definition from [69, 70]. Informally, RPE is a combination of error correcting code and secret sharing, in particular, it is an error correcting code with an additional secrecy property and reconstruction property. Linear error correcting secret sharing schemes are a special case of RPEs.

Definition 19 (Binary Reconstructable Probabilistic Encoding [67, 68])**.** We say a triple (E, D, R) is a binary reconstructable probabilistic encoding scheme with parameters $(k, n, c^{\mathsf{err}}, c^{\mathsf{sec}})$, where $k, n \in \mathbb{N}$, $0 \leq c^{\mathsf{err}}, c^{\mathsf{sec}} < 1$, if it satisfies the following properties:

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

2. **Secrecy of partial views.** For all $x \in \{0,1\}^k$ and any non-empty set $S \subset [n]$ of size $\leq \lfloor c^{\mathsf{sec}} \cdot n \rfloor$, $E(x)_S$ is identically distributed to the uniform distribution over $\{0,1\}^{|S|}$.

3. **Reconstruction from partial views.** R is an efficient procedure that given any set $S \subset [n]$

[3] [48] implies this theorem and the parameters have been taken from [13].

of size $\leq \lfloor c^{\mathsf{sec}} \cdot n \rfloor$, any $\hat{c} \in \{0,1\}^n$, and any $x \in \{0,1\}^k$, samples from the distribution $\mathrm{E}(x)$ with the constraint $\mathrm{E}(x)_S = \hat{c}_S$.

For the special case of $c^{\mathsf{err}} = 0$, this is object is called binary ramp secret sharing scheme (with parameters (k, n, c^{sec})).

Lemma 4. *[69, 70, 67, 68] For any $k \in \mathbb{N}$, there exist constants $0 < c^{\mathsf{rate}}, c^{\mathsf{err}}, c^{\mathsf{sec}} < 1$ such that there is a binary RPE scheme with parameters $(k, c^{\mathsf{rate}}k, c^{\mathsf{err}}, c^{\mathsf{sec}})$.*

To achieve longer encoding lengths n, with the same $c^{\mathsf{err}}, c^{\mathsf{sec}}$ parameters, one can simply pad the message to an appropriate length.

We conclude this section, by observing that binary ramp secret sharing schemes (RPEs with parameters $(k, n, 0, c^{\mathsf{sec}})$) are implied by any linear error correcting code with parameters (k, n, d) where k is the message length, n is the codeword length, $d := c^{\mathsf{sec}} \cdot n + 1$ is the minimal distance.

Lemma 5. *Suppose there exists a binary linear error correcting code with parameters (k, n, d), then there is a binary RPE scheme with parameters $(k, n, 0, (d-1)/n)$.*

Proof. For a linear error correcting code with (k, n, d), let A denote its encoding matrix, H denote its parity check matrix. Let B be a matrix so that $BA = I$ where I is the $k \times k$ identity matrix (such B exists because A has rank k and can be found efficiently). By property of parity check matrix, $HA = \mathbf{0}$ and $Hs \neq 0$ for any $0 < ||s||_0 < d$ where $\mathbf{0}$ is the $(n-k) \times k$ all 0 matrix.

We define $(\mathrm{E}, \mathrm{D}, \mathrm{R})$ as follows: for $x \in \{0,1\}^k$ and randomness $r \in \{0,1\}^{n-k}$, $\mathrm{E}(x; r) := B^T x + H^T r$, for $c \in \{0,1\}^n$; $\mathrm{D}(c) := A^T c$; given $S \subset [n]$ of size $\leq d-1$, $\hat{c} \in \{0,1\}^n$, $x \in \{0,1\}^k$, R samples r uniformly from the set of solutions to $(H^T r)_S = (\hat{c} - B^T x)_S$ then outputs $\mathrm{E}(x; r)$.

(E, D) is an encoding scheme because $\mathrm{D} \circ \mathrm{E} = A^T B^T = I^T = I$. For secrecy property, note that for any non-empty $S \subseteq [n]$ of size at most $d-1$, $(Hr)_S$ is distributed uniformly over $\{0,1\}^{|S|}$, because for any $a \in \{0,1\}^{|S|}$,

$$\Pr_r[(H^T r)_S = a] = \mathrm{E}[\Pi_{i \in S} \frac{1 + (-1)^{(H^T r)_i + a_i}}{2}] = 2^{-|S|} \sum_{S' \subseteq S} \mathrm{E}[\Pi_{i \in S'} (-1)^{(H^T r)_i + a_i}] = 2^{-|S|},$$

where the last equality is because the only surviving term is $S' = \emptyset$ and for other S', $\sum_{i \in S'} H_i^T \neq 0$ so $\mathrm{E}[\Pi_{i \in S'}(-1)^{(H^T r)_i}] = 0$. It implies $\mathrm{E}(x)_S$ is also distributed uniformly over $\{0,1\}^S$. By definition, R satisfies reconstruction property. Hence $(\mathrm{E}, \mathrm{D}, \mathrm{R})$ is a binary RPE with parameters $(k, n, 0, (d-1)/n)$.

□

Lemma 6. *Suppose there exists a binary linear error correcting code with parameters (k, n, d), then there is a binary ramp secret sharing scheme with parameters $(k, n, (d-1)/n)$.*

Proof. For a linear error correcting code with (k, n, d), let A denote its encoding matrix, H denote its parity check matrix. Let B be a matrix so that $BA = I$ where I is the $k \times k$ identity matrix (such B exists because A has rank k and can be found efficiently). By property of parity check matrix, $HA = \mathbf{0}$ and $Hs \neq 0$ for any $0 < ||s||_0 < d$ where $\mathbf{0}$ is the $(n-k) \times k$ all 0 matrix.

We define $(\mathrm{Enc}, \mathrm{Dec})$ as follows: for $x \in \{0,1\}^k$ and randomness $r \in \{0,1\}^{n-k}$, $\mathrm{Enc}(x; r) := B^T x + H^T r$, for $c \in \{0,1\}^n$; $\mathrm{Dec}(c) := A^T c$.

$(\mathrm{Enc}, \mathrm{Dec})$ is an encoding scheme because $\mathrm{Dec} \circ \mathrm{Enc} = A^T B^T = I^T = I$. For secrecy property, note that for any non-empty $S \subseteq [n]$ of size at most $d-1$, $(Hr)_S$ is distributed uniformly over $\{0,1\}^{|S|}$, because for any $a \in \{0,1\}^{|S|}$,

$$\Pr_r[(H^T r)_S = a] = \mathrm{E}[\Pi_{i \in S} \frac{1 + (-1)^{(H^T r)_i + a_i}}{2}] = 2^{-|S|} \sum_{S' \subseteq S} \mathrm{E}[\Pi_{i \in S'}(-1)^{(H^T r)_i + a_i}] = 2^{-|S|},$$

where the last equality is because the only surviving term is $S' = \emptyset$ and for other S', $\sum_{i \in S'} H_i^T \neq 0$ so $\mathrm{E}[\Pi_{i \in S'}(-1)^{(H^T r)_i}] = 0$. It implies $\mathrm{Enc}(x)_S$ is also distributed uniformly over $\{0,1\}^S$. Hence $(\mathrm{Enc}, \mathrm{Dec})$ is a binary ramp secret sharing encoding scheme with parameters $(k, n, (d-1)/n)$.

□

To achieve longer encoding lengths n, with the same c^{sec} parameter, one can simply pad the message to an appropriate length.

2.6.3 Bounded Independence

We note some useful facts about bounded independence.

Definition 20. Random variables, $X_1, \ldots, X_n$ supported on a set S, are *t-wise independent* if for any $s_1, \ldots, s_t \in S$ and any distinct $i_1, \ldots, i_t \in [n]$,

$$\Pr[X_{i_1} = s_1 \wedge \cdots \wedge X_{i_t} = s_t] = \prod_{j=1}^{t} \Pr[X_{i_j} = s_j]$$

We say a hash family $\mathcal{H} = \{h : [n] \to [m]\}$ is said to be *t-wise independent* if the random variables $H(1), \ldots, H(n)$, where H is sampled uniformly from $\mathcal{H}$, are t wise independent.

We say function $G\colon \{0,1\}^s \to \{0,1\}^n$ is a σ-wise independent generator if $G(\zeta)$ is σ-wise independent where ζ is uniformly distributed over $\{0,1\}^s$.

Note that if $\mathcal{H}$ is a t-wise independent hash family of size 2^s, then $G(H) := H(1), \ldots, H(n)$ is a t-wise independent generator.

The simple Carter-Wegman hashing construction based on random polynomials suffices for our purposes.

Lemma 7. *[71] There exists an (explicit) t-wise independent generator: $G : \{0,1\}^{(t+1)\log n} \to \{0,1\}^n$, computable in time $\tilde{O}(tn)$.*

Moreover, $G : \{0,1\}^{(t+1)m} \to \{0,1\}^n$ can be constructed such that for a subset $S \subseteq [n]$ of size σ, $G(\zeta)_S \equiv X_1, \ldots, X_t$ (for uniformly chosen ζ) where (1) X_i's are independent Bernoullis with $\Pr[X_i = 1] = q/d$ for $q \in \{0, \ldots, d\}$, and (2) $m = \max\{\log n, \log d\}$.

Let $G_{\sigma,p}$ denote a σ-wise independent Carter-Wegman generator with bias p, and G_σ such a generator with $p = 1/2$

2.6.4 Characters sums over finite fields

Let q be a prime. The additive characters of $\mathbb{F}_q$ are of the form $\chi_j(x) = e(xj/q)$, for $j = 0, 1, \ldots, q-1$. χ_0 is called the trivial character, and the others are called as non-trivial characters

of $\mathbb{F}_q$. We now recall a deep result from algebraic geometry that has found various applications in pseudorandomness.

Theorem 11 (Weil bound [72])**.** *Let p be a non-constant univariate polynomial of degree $d < q$ over $\mathbb{F}_q$. For any non-trivial additive character χ of $\mathbb{F}_q$, we have*

$$\left|\sum_{y\in\mathbb{F}_q} \chi(p(y))\right| \leq d\sqrt{q}.$$

We record a couple of XOR lemmas that lets us translate bounds on expectations of characters under a distribution D, to the closeness of D in statistical distance to the uniform distribution.

Lemma 8 ([73])**.** *For every prime q, there exists an efficiently computable map $\sigma : \mathbb{F}_q \to \{0,1\}^m$ such that if Y is a distribution on $\mathbb{F}_q$ such that for every non-trivial additive character χ of $\mathbb{F}_q$,*

$$\mathbb{E}[\chi(Y)] \leq \delta,$$

then it is the case that

$$|\sigma(Y) - \mathrm{U}_m| \leq \epsilon,$$

where $\epsilon = \delta 2^{m/2} + O(2^m/q)$.

Lemma 9 ([73, 74])**.** *For every prime q, there exists an efficiently computable map $\sigma : \mathbb{F}_q \to \{0,1\}^m$ such that if (Y, Y') is a distribution on $\mathbb{F}_q \times \mathbb{F}_q$ where for all additive characters χ, ϕ of $\mathbb{F}_q$, where χ is non-trivial,*

$$\mathbb{E}[\chi(Y)\phi(Y')] \leq \delta,$$

then it is the case that

$$|(\sigma(Y), \sigma(Y')) - (\mathrm{U}_m, \sigma(Y'))| \leq \epsilon,$$

where $\epsilon = \delta 2^m + O(2^m/q)$.

2.6.5 Pseudorandom Generators for Space-Bounded Computation

Definition 21. [75] A generator $\mathsf{prg} : \{0,1\}^m \to (\{0,1\}^n)^k$ is a *pseudorandom generator for space(w) and block size n with parameter* ε if for every finite state machine, Q, of size 2^w over alphabet $\{0,1\}^n$ we have that

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

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

Theorem 12. *[75] There exists a fixed constant $c > 0$ such that for any $w, k \leq cn$ there exists an (explicit) pseudorandom generator* $\mathsf{prg} : \{0,1\}^{O(k)+n} \to \{0,1\}^{n2^k}$ *for space(w) with parameter* 2^{-cn}. *Moreover,* prg *can be computed in polynomial time (in $n, 2^k$).*

2.6.6 The Pseudorandom Switching Lemma of Trevisan and Xue

Definition 22. Fix $p \in (0,1)$. A string $s \in \{0,1\}^{n \times \log(1/p)}$ encodes a subset $L(s) \subseteq [n]$ as follows: for each $i \in [n]$,

$$i \in L(s) \iff s_{i,1} = \cdots = s_{i,\log(1/p)} = 1.$$

Definition 23. Let $\mathcal{D}$ be a distribution over $\{0,1\}^{n\log(1/p)} \times \{0,1\}^n$. This distribution defines a distribution $\mathcal{R}(\mathcal{D})$ over restrictions $\{0,1,*\}^n$, where a draw $\rho \leftarrow \mathcal{R}(\mathcal{D})$ is sampled as follows:

1. Sample $(s, y) \leftarrow \mathcal{R}(\mathcal{D})$, where $s \in \{0,1\}^{n\log(1/p)}$, $y \in \{0,1\}^n$.
2. Output ρ where

$$\rho_i := \begin{cases} y_i & \text{if } i \notin L(s) \\ * & \text{otherwise} \end{cases}$$

Theorem 13 (Polylogarithmic independence fools CNF formulas [76, 77])**.** *The class of M-clause CNF formulas is ϵ-fooled by $O((\log(M/\epsilon))^2)$-wise independence.*

Theorem 14 (A Pseudorandom version of Håstad's switching lemma [12, 78])**.** *Fix* $p, \delta \in (0,1)$ *and* $w, S, t \in \mathbb{N}$*. There exists a value* $r \in \mathbb{N}$*,*

$$r = \text{poly}(t, w, \log(S), \log(1/\delta), \log(1/p)),^{4}$$

such that the following holds. Let $\mathcal{D}$ *be any* r*-wise independent distribution over* $\{0,1\}^{n\times\log(1/p)} \times \{0,1\}^{n}$ [5] *. If* $F : \{0,1\}^n \to \{0,1\}$ *is a size-*S *depth-*2 *circuit with bottom fan-in* w*, then*

$$\Pr\left[\, \text{DT}(F \restriction \boldsymbol{\rho}) \geq t \,\right] \leq 2^{w+t+1}(5pw)^t + \delta,$$

where the probability is taken with respect to a pseudorandom restriction $\boldsymbol{\rho} \leftarrow \mathcal{R}(\mathcal{D})$*.*

Proof. By Lemma 7 of [78], any distribution $\mathcal{D}'$ over $\{0,1\}^{n\times\log(1/p)} \times \{0,1\}^n$ that ϵ-fools the class of all $(S \cdot 2^{w(\log(1/p)+1)})$-clause CNFs satisfies

$$\Pr\left[\, \text{DT}(F \restriction \boldsymbol{\rho}) \geq t \,\right] \leq 2^{w+t+1}(5pw)^t + \epsilon \cdot 2^{(t+1)(2w+\log S)},$$

where the probability is taken with respect to a pseudorandom restriction $\boldsymbol{\rho} \leftarrow \mathcal{R}(\mathcal{D}')$. By Theorem 13, the class of $M := (S \cdot 2^{w(\log(1/p)+1)})$-clause CNF formulas is

$$\epsilon := \delta \cdot 2^{-(t+1)(2w+\log S)}$$

fooled by r-wise independence where

$$r = O((\log(M/\epsilon))^2) = \text{poly}(t, w, \log(S), \log(1/\delta), \log(1/p)),$$

and the proof is complete. □

[4] The exponent of this polynomial is a fixed absolute constant independent of all other parameters.

[5] Recall that an r-wise independent distribution over $\{0,1\}^m$ can be generated with seed length $O(r \log m)$. In this case, $m = n \times \log(1/p) + n$ so this is $O(r \log(n \log(1/p)))$.

Taking a union bound we get the following corollary.

Corollary 1. *Fix $p, \delta \in (0,1)$ and $w, S, t \in \mathbb{N}$. There exists a value $r \in \mathbb{N}$,*

$$r = \mathrm{poly}(t, w, \log(S), \log(1/\delta), \log(1/p)),$$

such that the following holds. Let $\mathcal{D}$ be any r-wise independent distribution over $\{0,1\}^{n\times\log(1/p)} \times \{0,1\}^n$. Let $F_1, \ldots, F_M$ be M many size-S depth-2 circuits with bottom fan-in w. Then

$$\Pr_{\boldsymbol{\rho}\leftarrow\mathcal{R}_p}\left[\exists\, j \in [M] \text{ such that } \mathrm{DT}(F_j \restriction \boldsymbol{\rho}) \geq t\right] \leq M \cdot \left(2^{w+t+1}(5pw)^t + \delta\right). \tag{2.1}$$

2.7 Concentration Inequalities and Other Useful Lemmas

Theorem 15 (Generalized Chernoff Bound [79])**.** *Let $X_1, \ldots, X_n$ be Boolean random variables such that, for some $0 \leq \delta \leq 1$, we have that, for every subset $S \subseteq [n]$, $\mathrm{E}[\Pi_{i\in S} X_i] \leq \delta^{|S|}$. Then $\Pr[\sum_{i=1}^n X_i \geq 2\delta n] \leq \exp(-n\delta/3)$.*

The following useful theorems give Chernoff-type concentration bounds for t-wise independent distributions.

Theorem 16 ([80])**.** *Let $X_1, \ldots X_n$ be t-wise independent random variables supported on $[0,1]$ such that $\mu = \mathbb{E}[\sum_{i=1}^n X_i]$, then for $A > 0$*

$$\Pr\left[\left|\sum_{i=1}^n X_i - \mathbb{E}\left[\sum_{i=1}^n X_i\right]\right| \geq A\right] \leq 1.0004\left(\frac{tn}{A^2}\right)^{t/2}.$$

Theorem 17 ([81])**.** *If X is a sum of t-wise independent random indicator variables with $\mu = \mathbb{E}[X]$, then $\forall \epsilon : 0 < \varepsilon \leq 1, t \leq \varepsilon^2\mu e^{-1/3}$, $\Pr[|X-\mu| > \varepsilon\mu] < \exp(-\lfloor t/2 \rfloor)$.*

We recall a useful result from [82] (Lemma 4.2).

Lemma 10. *Let n, r, d, λ be arbitrary positive integers, and q be a prime. Let $p_1(x), \ldots, p_r(x)$ be non-constant n-variate polynomials of degree at most d over $\mathbb{F}_q$. Suppose that $d_i = deg(p_i)$.*

Define $c_i = \lambda(2dr+1) + \lambda i$. *Then, for all* $1 \le i < j \le r$, *we have*

$$|deg(p_i^{c_i}) - deg(p_j^{c_j})| = |c_i \cdot d_i - c_j \cdot d_j| \ge \lambda.$$

We also record the Schwartz-Zippel Lemma.

Lemma 11 ([26, 83])**.** *Let* $p(x) \in \mathrm{Poly}_{n,q,d}$ *be a non-zero polynomial. Then,*

$$\Pr_{x \in \mathbb{F}_q^n}[p(x) = 0] \le d/q.$$

Proposition 1 (Implicit in Lemma 3.8.1 in [84])**.** *Let* X_0, X_1 *be random variables such that* $\Delta(X_0; X_1) = \epsilon$. *Consider the following game:*

- *Arthur samples a coin* $b \leftarrow \mathcal{U}$ *and gives Merlin* $x \leftarrow X_b$.
- *Merlin responds with* b'. *If* $b' = b$, *Merlin wins. Otherwise, Merlin loses.*

Merlin wins with probability $\frac{1+\epsilon}{2}$ *by outputting* b' *such that* $\Pr[X_{b'} = x] \ge \Pr[X_{1-b'} = x]$. *Moreover, this strategy is optimal.*

We also use the following simple combinatorial propositions.

Proposition 2. *Let* $c > 1$. *Let* $(XY), (XZ)$ *be two joint random variables supported on any space* $\Sigma_X \times \Sigma$ *such that* $\Delta(X, Y; X, Z) \le \epsilon$, *then exists an event* $S \subseteq \Sigma_X$ *such that*

1. $\Pr[X \in S] \ge 1 - 1/c$
2. $\forall x \in S, \Delta(XY|X = x; XZ|X = x) \le c\epsilon$, *where* $(XY|X = x)$ *denotes the random variable* XY *conditioned on* $X = x$ *and, similarly,* $(XZ|X = x)$ *denotes the random variable* XZ *conditioned on* $X = x$.

Proof. Let A the variable distributed according to the procedure where $x \leftarrow X$ and then $\Delta(XY|X =$

$x; XZ|X = x)$ is output. By definition, $\mathbb{E}[A] \leq \epsilon$.[6] Thus, by Markov's inequality we have

$$\Pr[A \geq c\epsilon] \leq \frac{\mathbb{E}[A]}{c\epsilon} \leq \frac{1}{c}$$

It follows that there exists a set S with the desired properties. In particular, S is the set of x such that A conditioned on $X = x$ is *not* greater than $c\epsilon$. □

Proposition 3. *Let $\beta \in (0, 1)$ Let $(XY), (XZ)$ be two joint random variables supported on any space $\Sigma_X \times \Sigma$ such that $\Delta(X, Y; X, Z) > \epsilon$, then exists an event $S \subseteq \Sigma_X$ such that*

1. $\Pr[X \in S] \geq \epsilon - \beta$
2. $\forall x \in S, \Delta(XY|X = x; XZ|X = x) \geq \beta$, *where $(XY|X = x)$ denotes the random variable XY conditioned on $X = x$ and, similarly, $(XZ|X = x)$ denotes the random variable XZ conditioned on $X = x$.*

Proof. Let A the variable distributed according to the procedure where $x \leftarrow X$ and then $\Delta(XY|X = x; XZ|X = x)$ is output. Let $A' = 1 - A$. By definition, $\mathbb{E}[A'] < 1 - \epsilon$. Thus, by Markov's inequality we have,

$$\Pr[A' \geq 1 - \beta] < \frac{1 - \epsilon}{1 - \beta}.$$

So, it follows that $\Pr[A \leq \beta] < \frac{1-\epsilon}{1-\beta}$. Thus

$$\Pr[A > \beta] \geq 1 - \frac{1 - \epsilon}{1 - \beta} = \frac{\epsilon - \beta}{1 - \beta} \geq \epsilon - \beta.$$

It follows that there exists a set S with the desired properties. In particular, S is the set of x such that A conditioned on $X = x$ is greater than β. □

[6] $\mathbb{E}[A] = \sum_{x \in \Sigma_X} \Pr[X = x]\Delta(XY|X = x; XZ|X = x) = \Delta(XY; XZ)$.

Chapter 3: Warming Up

In this chapter we provide some elementary results that, hopefully, grant some contextual clarity to the rest of this work.

Efficient codes exist for small families. In Section 3.1, we extend an argument from [7] show that for any family of tampering functions of size 2^t, $\mathcal{F}$, if D is chosen to be drawn from a t'-wise independent hash family, $\{h : \{0,1\}^n \to \{0,1\}\}_{h \in \mathcal{H}}$, ($t' \approx t$) then with overwhelming probability (E, D) is a single-bit non-malleable code for $\mathcal{F}$, where E encodes b by sampling a random string, x, conditioned on $\mathrm{D}(x) = b$.[1]

In Section 3.2, we observe that this construction can be further derandomized for the specific case of polynomial size circuit tampering. This is because, for this class, the property of *not* being a non-malleable code for this class admits an MA proof (provided (E, D) can be implemented by polynomial size circuits).

Non-Malleability implies lower bounds. In section 3.3, we present the observation that explicit non-malleability against a tampering class $\mathcal{F}$ (extended in the usual way), implies explicit average-case hardness lower bounds against $\mathcal{F}$, for many natural families, $\mathcal{F}$. This can also be interpreted as showing a non-malleable code for $\mathcal{F}$ is in fact leakage resilient against a related class.

On non-malleable codes as a relaxation of error-correcting codes: tampering functions that change $d/2$ bits, where d is the distance of the code. It is common to present non-malleable codes as a strict relaxation of error correcting codes. Non-malleable only guarantee correctness of decoding in the absence of errors and consequently gain "security" against a wider range of

[1] In fact, E as described may not be efficient, but rejection sampling suffices (with appropriate fallback behavior after sufficiently many failures).

adversarial channels, in particular channels that can modify more bits of the codeword. However, note that in all known results, there is a tradeoff: Non-malleable codes allow for modifying more (or all) symbols of the codeword than error correcting codes but require that the computation of the tampering function is restricted in some way, while error correcting codes can tolerate modification of less codeword symbols, but do not place any computational restrictions on the tampering adversary.

In Section 3.4, we ask whether this is in fact necessary: Specifically, does the relaxed definition of non-malleable codes allow us to construct codes that allow for modifying more symbols of the codeword than error correcting codes *without* placing any computational restrictions on the tampering. Note that error correcting codes with distance d that allow for error correction after modification of at most $(d-1)/2$ symbols are known (e.g. Reed Solomon codes achieve this bound) and error correction is impossible under modification of $d/2$ symbols. Given the above, we resolve question negatively, showing that it is impossible to construct non-malleable codes with distance d against tampering functions that arbitrarily modify $d/2$ codeword symbols. This indicates that in order to obtain improved parameters beyond what is possible with error correcting codes, imposing computational restrictions on the tampering function is *necessary*.

3.1 An Efficient One-Bit NMC for Small Tampering Families Exists

Dziembowski Pietrzak and Wichs [7] showed that for any tampering family $\mathcal{F}$ that is not too much larger, if the decoding function D is simply chosen at random, and encoding, E, simply selects a random string that decodes to the correct value, this code is non-malleable against $\mathcal{F}$ with overwhelming probability.

We show that, for protecting a single bit, selecting a $O(\log|\mathcal{F}|)$-wise independent hash suffices for D. To encode, rejection sampling is enough.

Theorem 18. *For any $\mathcal{F}$ of size at most $2^{t'}$, the construction in Figure 3.1 with parameters $n > \log(12t/\epsilon^2)$ and $t \in [2\log(3|\mathcal{F}|/\beta), \epsilon^2/6 \cdot 2^n]$ is an ϵ-NMC for $\mathcal{F}$ with probability at least $1-\beta$.*

Proof. We follow the framework of Dziembowski, Pietrzak, and Wichs.

Let n be a parameter.
Let $\mathcal{H} = \{h : \{0,1\}^n \to \{0,1\}\}$ be a t-wise independent hash family.

Preprocessing Sample $h \leftarrow \mathcal{H}$. Output h. Let c_0, c_1 be strings such that $h(c_0) = 0$ and $h(c_1) = 1$.

Encode To encode a bit b:

Sample i.i.d. $x_1, \ldots, x_n \xleftarrow{u} \{0,1\}^n$. Output the first x_i such that $h(x_i) = b$, if one exists. Otherwise, output c_b.

Decode On input c:

Output $h(c)$.

Figure 3.1: A Non-Malleable Code for Small Families

Let $(\mathrm{E}_h, \mathrm{D}_h)$ denote the code defined by $h \in \mathcal{H}$.

Fix some $f \in \mathcal{F}$. Let G_f denote the digraph with the vertex set $V(G) = \{0,1\}^n$ and edge set $E(G) = \{(x, f(x)) : x \in \{0,1\}^n\}$. We will consider $\mathcal{H}$ as a set of colorings of G_f.

Note that the rejection sampling procedure fails with probability at most 2^{-n}. Conditioned on that event, S, not happening, $\mathrm{E}_h(b)$ outputs a uniformly random $x \in \{0,1\}^n$ such that $h(x) = b$.

Observe that if we conditioned on the success of the reduction sampling (i.e. S not happening), $\mathrm{E}_h(b)$ for uniform b outputs a uniformly random string $x \in \{0,1\}^n$. Additionally, note that because G_f has out-degree 1, so we can associate each edge $e = (u, v)$ with a unique vertex u.

So, conditioned on S not happening for any fixed h,

$$\begin{aligned}
\Pr_{b \xleftarrow{u} \{0,1\}}[\mathrm{D}(f(\mathrm{E}(b))) = 1 - b] &= \Pr_{c \xleftarrow{u} \{0,1\}^n}[\mathrm{D}(f(x)) = 1 - h(x)] \\
&= \Pr_{(u,v) \xleftarrow{u} E(G_f)}[h(u) \neq h(v)] \\
&= \frac{1}{2^n}\left(\sum_{(u,v) \in E(G_f)} \chi_{h(u) \neq h(v)}\right).
\end{aligned}$$

In other words, if S does not happen, f breaks ϵ-non-malleability with respect to $(\mathrm{E}_h, \mathrm{D}_h)$ if and only if

$$\sum_{e \in E(G_f)} \chi_{h(u) \neq h(v)} \geq (1/2 + \epsilon)2^n.$$

Proposition 4 ([7])**.** *For any directed graph $G = (V, E)$ where each vertex has out-degree one, there is a partition of E into two components U_1, U_2 such that neither component contains a non-trivial cycle and each component contains at least $1/3$ of the edges in G.*

Proof. Note that because each vertex has out-degree one, it can be on at most one cycle. Let $C_1, C_2, \ldots$ be an enumeration of non-trivial cycles. Partition the n_i vertices of C_i arbitrarily such that $\lfloor n_i/2 \rfloor$ are in U_1 and $\lceil n_i/2 \rceil$ are in U_2. Partition the remaining n^* vertices such that $\lfloor n^*/2 \rfloor$ are in U_2 and $\lceil n^*/2 \rceil$ are in U_1.

Neither U_1 nor U_2 contains a complete non-trivial cycle and because the smallest odd non-trivial cycle is of size 3, we have each of $|U_1|, |U_2| \geq |E|/3$. □

Given this claim, it suffices to prove that each of $G_1 = (V, U_1)$ and $G_2 = (V, U_2)$ are bounded with high probability, namely

$$\text{for } i = 1, 2: \qquad \Pr_h \left[\sum_{(u,v) \in U_1} \chi_{h(u) \neq h(v)} \geq (1/2 + \epsilon)|G_i| \right] \leq \frac{\beta}{3|\mathcal{F}|}.$$

The theorem follows by union bounding over all $f \in \mathcal{F}$, as well as the event S.

Fix G_i (for $i \in \{1, 2\}$) from the claim above.

Let $(X_e)_{e=(u,v) \in E(G_i)}$ denote the sequence of variables for each $e \in U_i$, $X_e = 1$ if and only if $h(u) \neq h(v)$, where $h \leftarrow \mathcal{H}$.

Because H is at least 2-wise independent, for any X_e, $\mathbb{E}[X_e] \leq 1/2$ (if $e = (u, u)$ for some u then X_e is constant 0).

Claim 1. The X_e's are t-wise independent.

Proof. Fix any set $S' \subset U_i$ of size at most t. Any self-loops, e, in S', corresponding to variables X_e that are identically equal to 0, independently of H and all other variables $X_{e'}$. So, it suffices to consider S the subset of edges in S' that are not self-loops.

Now, consider the connected components in G'_S, the subgraph of G_i induced by S, with all vertices lacking adjacent edges removed.

By Claim 4, there are no non-trivial cycles in G'_S. Therefore, G'_S is a directed acyclic graph with out-degree 1. Consider sampling $(X_e)_{e\in S}$ by coloring vertices according to H in reverse topological order. Because h is $2t$-wise indedendent and there are at most $2t$ vertices in G'_S, it is equivalent to randomly color each vertex one-by-one.

Namely, at step i, color the ith vertex in the reverse toplogical ordering, v_i, according to H, $H(v_i)$. At the outset i-th step of this procedure, the $(i-1)$th vertex, v_{i-1} has been colored by H. Recall, that because the edge $e_i = (v_i, v_{i-1})$ is the unique on-going edge from v_i, by the reverse toplogical ordering this means v_i has not yet been colored at the outset of step i.

Moreover, regardless of whether we condition on $h(v_{i-1}) = 0$ or $h(v_{i-1}) = 1$, the probability that $h(v_i) \neq h(v_{i-1})$ (and hence $\Pr[X_{e_i} = 1 | X_{e_{i-1},\ldots,X_{e_1}}]$) remains $1/2$ (i.e. $\Pr[X_{e_i} = 1]$). □

By linearity of expectation, $\mathbb{E}[\sum_{e\in E(G_i)} X_e] \leq 1/2|E(G_i)|$. Therefore by Theorem 16,

$$\Pr\left[\sum X_e - \mathbb{E}[\sum X_e] \geq \epsilon|E(G_i)|\right] \leq 1.0004\left(\frac{t|E(G_i)|}{\epsilon^2|E(G_i)|^2}\right)^{t/2}$$

Because $t \leq \epsilon^2/6 \cdot 2^n \leq \epsilon^2/2 \cdot |E(G_i)|$, we have

$$\begin{aligned}\Pr[\sum_{e\in E(G_i)} X_e \geq (1/2+\epsilon)|G_i|] &\leq 1.0004\left(\frac{t|E(G_i)|}{\epsilon^2|E(G_i)|^2}\right)^{t/2} \\ &\leq 1.0004 \cdot 2^{-t/2}\end{aligned}$$

So, the theorem follows so long as $t \geq 2\log(3|\mathcal{F}|/\beta)$. □

3.2 An Efficiently Enumerable Family Exists for Small Circuit Tampering (under strong derandomization assumptions)

In this section, we show that a PRG for MA protocols suffices to fool the test of non-malleability for small circuit sampling. Assuming a PRG against MA distinguishers indeed allows us to shrink the family from Section 3.1 to a polynomial number of candidate codes, most of which remain good. This, in particular, implies an efficient construction in the CRS model where the CRS is

smaller than size of the tampering functions.

We do not discuss this PRG assumption here, because in Chapter 9 we show how to get rid of the CRS entirely and encode more bits from the related assumption that E requires nondeterministic circuits of exponential size. In particular, this assumption implies PRGs for non-deterministic circuits which suffices for Theorem 19.

Theorem 19. *For any constant c, let* $\mathsf{SIZE}[n^c]$ *denote the class of n-bit to n-bit functions that can be represented by an* n^c *size circuit. Note that* $|\mathsf{SIZE}[n^c]| = 2^{2cn^c \log n)}$*, for almost all n.*

Let $t = 2\log(3) + 4cn^c \log n + n$ *and* $n/2 > \log(12t)$..

Let $c' > c + \log(t \log t \log^2 n) + 1$.

Let $\mathrm{E}_h, \mathrm{D}_h$ *be the encoding and decoding procedures from Figure 3.1 where h is a t-wise independent hash from the family* $\mathcal{H}$ *of Lemma 7.*

And let $G : \{0,1\}^s \to \{0,1\}^m$ *be a PRG such that no MA protocol with Arthur size at most* $m^{c'}$ *has advantage greater than* $m^{-c'}/3$*, where* $m = O(t \log n)$ *is the description length of the family* $\mathcal{H}$ *from Lemma 7.*

Then, for any $\epsilon > m^{-c'}$ *there is a coding scheme with a CRS of length s that is* ϵ*-non-malleable against* $\mathcal{F}$ *with probability at least* $m^{-c'}/\epsilon$*, over the randomness of the CRS.*

Let $G : \{0,1\}^s \to \{0,1\}^\ell$ be a PRG for MA protocols with Arthur a (randomized) circuit of size $n^{c'}$ with security α. Let $\mathcal{G}$ denote the pseudorandom distribution corresponding to the output of G on a uniform seed.

Now, suppose the construction in Figure 3.1 is not ϵ-non-malleable with probability at least δ when the CRS is drawn from the pseudorandom distribution $\mathcal{G}$ instead of a random seed. In particular, with probability at least δ (over $g \leftarrow \mathcal{G}$) there exists an f such that

$$\Pr_b[\mathrm{D}_g(f(\mathrm{E}_g(b))) = 1 - b] > 1/2 + \epsilon.$$

Then, consider the following randomized non-deterministic distinguisher. On input, x:

Merlin Send f of size n^c from above, if such an f exists.

Arthur

- Sample uniform bit b.
- Sample $b' = \mathrm{D}_x(f(\mathrm{E}_x(b)))$.
- If $b' \neq b$, output 1. Otherwise, output 0.

Note that Arthur is size $O(n^c + \ell^2)$

If x is drawn from $\mathcal{G}$, then with probability δ the distinguisher outputs 1 with probability $1/2+\epsilon$. On the other hand, a constant function c, always makes the distinguisher output 1 with probability at least $1/2$. So in total, the distinguisher on $\mathcal{G}$ outputs 1 with probability at least $1/2 + \delta \cdot \epsilon$.

If x is drawn from $\mathcal{U}$, then with probability at least $1 - \delta'$, the distinguisher outputs 1 with probability at most $1/2 + \epsilon'$. So, in total the distinguisher outputs 1 with probability at most $1/2 + \epsilon' + \delta'$.

So, in total the distinguisher has advantage $\delta\epsilon - \epsilon' - \delta'$. And in conclusion, we must have $\delta\epsilon < \epsilon' + \delta' + \alpha$.

Thus, if $\alpha, \delta', \epsilon' < n^{-c'}/3$, then it must be that $\delta, \epsilon < n^{-c'}$. In other words, the derandomized code must be ϵ non-malleable with probability at least $1 - n^{-c'}/\epsilon$.

3.3 Non-Malleability against circuits implies circuit lower bounds

In this section we assume a binary alphabet. Let C be a class of boolean functions with the following closure property (*): if $c \in C$, then for $a, b \in \{0, 1\}$ the function $c_{a,b} \in C$ where $c_{a,b}(x) = a$ if $c(x) = 0$ and $c_{a,b}(x) = b$ if $c(x) = 1$.

Now consider tampering class C as we consider in this paper: namely the ith tampered output corresponds to some $c \in C$, for all i.

Proposition 5. *If C has the closure property (*), then an explicit non-malleable code, even for a single bit,* $(\mathrm{Enc}, \mathrm{Dec})$ *implies an ϵ-hard distributional problem for C, with a samplable distribution. In particular,* $(\mathrm{Enc}(\mathcal{U})), \mathrm{Dec})$ *is ϵ-hard.*

Suppose for the sake of contradiction there exists some $c \in C$ such that

$$\Pr_{x \leftarrow \mathrm{Enc}(\mathcal{U})}[c(x) = \mathrm{Dec}(x)] > 1/2 + \epsilon.$$

Then, (Enc, Dec) is not ϵ-non-malleable with respect to C. In particular, consider the following attack. Fix any encoding of 1, x^1, and any encoding of 0, x^0. Now consider the tampering function such that the i tampered output is computed by $c_{x_i^0, x_i^1}$. Because c correctly decodes with probaility $> 1/2 + \epsilon$ this attack flips the encoded bit with probability $> 1/2 + \epsilon$. By the characterization of one-bit NMC, Lemma 1, (Enc, Dec) is not ϵ-non-malleable.

3.4 Non-malleability is no better than error-correction for many arbitrary errors

Let (E, D) be a coding scheme with distance d. Define the class of functions $\mathcal{F}_{d/2-1} = \{f : f$ changes $< d/2$ codeword symbols $\}$. We know that ECC exist, and thus NMC also exist, for $\mathcal{F}_{d/2-1}$ (e.g. Reed Solomon Codes achieve this bound).

We now define the slightly larger class $\mathcal{F}_{d/2} = \{f : f$ changes $\leq d/2$ symbols$\}$. In Theorem 20 we show that even inefficient NMC do not exist for $\mathcal{F}_{d/2}$, unless exactly one bit needs to be protected. Recall that distance for randomized coding schemes is upper bounded by the notion of distance for (non-randomized) codes with the same message/codeword-space parameters. Specifically, for a set of codewords S, we define the distance of S ($\mathsf{dist}(S)$) as the minimum pairwise distance over all pairs of codewords in S. Let $\mathcal{S}$, be the set that consists of all sets S that contain exactly one codeword for each message in the message space. Then the distance of the code is defined as $\max_{S \in \mathcal{S}} \mathsf{dist}(S)$. Refer to definition 3 for formal definition of distance of coding scheme, presented earlier.

3.4.1 Lower bound for encoding more than a bit

Theorem 20. *Let* (E, D) *be a coding scheme with message space of size greater than* 2*, alphabet* Σ *and distance* d*. Then, for any* $\epsilon > 0$*,* (E, D) *is not a* $\frac{1}{8} - \epsilon$*-NMC for* $\mathcal{F}_{d/2}$*.*

Proof. We begin with some notation Given $\alpha, \beta \in \Sigma^n$, we denote by $\|\alpha - \beta\|_0$ the number of positions $i \in [n]$ such that $\alpha_i \neq \beta_i$.

Let $(\mathsf{E} : U \to V, \mathsf{D} : V \to U)$ be a randomized encoding scheme, where $U \subseteq \Sigma^k, V \subseteq \Sigma^n$ and $|U| > 2$.

Claim 2. $\exists x \in U$ such that $\forall c_x \in \mathsf{E}(x)$ there is a $z = z(c_x) \in V$:

1. $\|c_x - z\|_0 \leq \frac{d}{2}$
2. $\Pr[\mathsf{D}(z) \neq x] =\geq \frac{1}{2}$.

Assuming the claim, consider the following tampering function $f \in \mathcal{F}_{d/2}$. Let z_c be the z for each $c \in \mathsf{E}(x^*)$ guaranteed to exist for some $x^* \in U$ by the above claim.

$$f(c) := \begin{cases} z_c & \text{if } c \in \mathsf{E}(x^*) \\ c & \text{otherwise} \end{cases}$$

Let $\Pr_{c_{x^*} \leftarrow \mathsf{E}(x^*)}[\mathsf{D}(z(c_{x^*})) \neq x^*] = p \geq \frac{1}{2}$. Then, $\exists y^* \neq x^* \in U$ such that $\Pr_{c_{x^*} \leftarrow \mathsf{E}(x^*)}[\mathsf{D}(z(c_{x^*})) = y^*] \leq \frac{p}{|U|-1}$, but $\Pr[\mathsf{D}(f(\mathsf{E}(y^*))) = y^*] = 1$. This means that a distribution $D^f_{x^*}$ that exactly agrees with $\mathsf{D}(f(\mathsf{E}(\cdot)))$ on x^* must output `same` or x^* with probability $1 - p$ and y^* with probability at most $\frac{p}{|U|-1}$. A distribution $D^f_{y^*}$ that exactly agrees with $\mathsf{D}(f(\mathsf{E}(\cdot)))$ on y^* must output `same` or y^* with probability 1. Thus, any distribution D^f can only agree with $\mathsf{D}(f(\mathsf{E}(\cdot)))$ for both x^* and y^* at most $(1 - p) + \frac{p}{|U|-1} \leq 3/4$ fraction of the time (and must have statistical distance at least $1/8$ from one of them), since $p \geq 1/2$ and $|U| > 2$.

All that remains is to prove the claim.

Suppose for the sake of contradiction that $\forall x \in U, \exists c_x \in \mathsf{E}(x)$ such that $\forall z \in V$ with $\|c_x - z\|_0 \leq \frac{d}{2}$ it is the case that $\Pr[\mathsf{D}(z) \neq x] < \frac{1}{2}$. Fix any such set of codewords corresponding to all messages $S = \{c_x : \forall z \in V \, \|c_x - z\|_0 \leq \frac{d}{2} \implies \Pr[\mathsf{D}(z) \neq x] < \frac{1}{2}\}_{x \in U}$. Note that the distance of S ($\min_{c_x \neq c_y \in S} \|c_x - c_y\|_0$) is at most d (by definition of the distance of a randomized code). Let $c_x \neq c_y \in U$ be two such codewords such that $\|c_x - c_y\| \leq d$. Then, $\exists z \in V$ such that $\|z - c_x\|_0 \leq d/2$ and $\|z - c_y\|_0 \leq d/2$. But then by assumption it follows that $\Pr[\mathsf{D}(z) = x] > \frac{1}{2}$

and $\Pr[\mathsf{D}(z) = y] > \frac{1}{2}$, which is a contradiction because $x \neq y$.

□

We observe that a randomized coding scheme with message space $\mathcal{M}$, codeword space C and distance d (as defined above for randomized coding schemes), implies a (possibly inefficient) error correcting code with the same message/codeword space that can correct up to $d/2 - 1$ errors. To see this, note that one can take the set $S \in \mathcal{S}$ (as in the definition of distance for randomized coding schemes) achieving the maximum $\mathsf{dist}(S)$. Since $S \subseteq C$ contains exactly one codeword for each message in the message space, the set S itself comprises a code with message space $\mathcal{M}$, codeword space C and distance d. This, in turn, implies a (possibly inefficient) error correcting code with message space $\mathcal{M}$ and codeword space C that can correct up to $d/2 - 1$ errors. We thus obtain the following corollary:

Corollary 2. *Fix a message space $\mathcal{M}$ ($|\mathcal{M}| > 2$) and a codeword space C. If the optimal (inefficient) error-correcting code for $(\mathcal{M}, C)$ can correct at most t errors, then there is no non-malleable code with message space $\mathcal{M}$ and codeword space C against tampering class $\mathcal{F}_{t+1}$.*

3.4.2 Protecting a single bit against $n - 1$ errors

In this section, we complement the above by showing when the message space has size 2 (i.e. single bit messages), non-malleable codes are possible against $d - 1$ arbitrary errors, whereas error correcting codes can tolerate at most $(d - 1)/2$ arbitrary errors. In the next section, we will show that if the message space is increased to 3 or more, then non-malleable codes are impossible even against $d/2$ errors.

The construction is simply a repetition code (E, D). On input a bit b, E outputs the string b^d (the bit b repeated d times). On input a string $b_1, \ldots, b_d$, D outputs 1 if there is some $i \in [d]$ such that $b_i = 1$. Otherwise, D outputs 0. Note that this code has distance d.

We next prove that (E, D) is a 0-non-malleable code (i.e. the two distributions in the security definition for non-malleable codes–see Definition 2–are identical). Applying Lemma 1, it is

sufficient to show that for every tampering function f that modifies at most $d - 1$ symbols,

$$\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1 - b] \leq \frac{1}{2},$$

We will use the fact that for the decode algorithm defined above,

$$\Pr[\mathsf{D}(f(\mathsf{E}(1))) = 0] = 0,$$

since a tampering function that modifies at most $d - 1$ bits cannot flip a 1 codeword to a tampered codeword that decodes to 0 under D.

Therefore,

$$\begin{aligned}\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1 - b] &= \frac{1}{2}\Pr[\mathsf{D}(f(\mathsf{E}(0))) = 1] + \frac{1}{2}\Pr[\mathsf{D}(f(\mathsf{E}(1))) = 0] \\ &= \frac{1}{2}\Pr[\mathsf{D}(f(\mathsf{E}(0))) = 1] \\ &\leq \frac{1}{2}.\end{aligned}$$

This completes the proof.

Chapter 4: Bounded-Communication Tampering (Leakage-Resilient Split-State)

In this chapter we construct non-malleable codes resilient to tampering attacks which can be formulated as two-player low-communication games, or leaky split-state tampering. More specifically, this class of tampering attacks corresponds to arbitrary two player protocol where they players, Alice and Bob, each get n-bit inputs, can communicate significantly less than n bits, and then each produce n-bit outputs. In our tampering experiment, we will imagine the codeword consists of two halves, L and R (where $|L| = |R| = n$) that are the inputs to Alice and Bob respectively. Alice and Bob then run their interactive protocol and we apply decoding to their outputs. Non-malleable codes resilient to this tampering class are also known as leakage-resilient split-state non-malleable codes. [13]

Definition 24 (Leaky/Bounded-Communication Split-State Model)**.** Let $\alpha \in [0, 1]$ be a parameter. We say $f \in \{0,1\}^{2k} \to \{0,1\}^{2k}$ is in *α-leaky split-state model*, $\alpha - \mathrm{SS}_k$ if there exists a communication protocol between Alice and Bob such that for $x = (x_{1:k}, x_{k+1:2k}) \in \{0,1\}^{2k}$, $f(x)$ can be computed by a communication protocol with parameter α between Alice and Bob where Alice has access to $x_{1:k}$, Bob has access to $x_{k+1:2k}$. Alice and Bob send information back and forth depending on their own inputs and the current transcript of the communication so far. Overall the total communication is at most αk bits and finally Alice outputs $f(x)_{1:k}$ and Bob outputs $f(x)_{k+1:2k}$.

We provide a construction of non-malleable codes for this class that can tolerate communication of up to a $1/4$-fraction of the codeword which, to our knowledge, is the best known.[1]

[1] [85] does not give an explicit bound on leakage and [13] allows $1/12$-fraction leakage (or $1/6$ in a more restricted model where the leakage amount from each side has to be the same).

Theorem 21. *There is an explicit split-state non-malleable code supporting leakage of up to a* $1/4$*-fraction of the bits with rate* $\Omega(\log\log n/\log n)$ *and error* $\exp(-\Omega(n\log\log n/\log n))$.

Additionally, there is an explicit split-state non-malleable code supporting leakage of up to a $1/4$*-fraction of the bits with rate* $\Omega(1)$ *and error* $\exp(-n^{\Omega(1)})$.

In particular, our construction allows one to reduce leakage-resilient split-state tampering to *any* split-state tampering with just a constant factor increase in codeword size. Theorem 21 is thus a corollary of our reduction, Lemma 12, with Theorem 4 and Theorem 3. Previously, such reductions were only known with worse parameters and if the underlying split-state code had decoder with certain properties [13]. On the other hand, [85] observe that split-state seedless non-malleable extractors *are*, in fact, leakage-resilient, yielding codes with rate comparable to that of the present work (but the leakage bound is left unspecified).

Unlike either of these previous constructions, our reduction yields improved leakage-resilient split-state non-malleable codes from *any* future improvement in the rate of explicit split-state non-malleable codes. In fact, our reduction immediately yields leakage-resilient split-state non-malleable codes with *constant rate* when combined with the recent construction of constant-rate split-state non-malleable codes due to Aggarwal and Obremski [36]. (This corresponds to the second item in Theorem 21.)

As we will later see in Chapter 9, our compiler also preserves certain useful properties of the underlying non-malleable code. Finally, our analysis is much simpler than that of [13].

We conclude the chapter by observing that our reduction preserves certain properties of the underlying split-state non-malleable code, which will be useful in Chapter 9.

4.1 Technical Overview

Our leakage-resilient reduction is quite intuitive. We show that given a statistically-secure leakage-resilient encryption scheme (where an adversary can receive bounded leakage from both ciphertext *and* the key used to encrypt it) it suffices to simply encrypt the left and right split-state codewords independently (with their own keys) and place the keys in the opposite state. By the

strong security property of the encryption scheme, the ciphertext hides each underlying split-state codeword piece, whatever (bounded amount) is leaked from the key in the opposite state.

To complete the reduction, we construct such a statistically-secure leakage-resilient encryption scheme from extractors. Our notion of leakage-resilience essentially combines the notions of "forward-secure storage" [86] and "leakage-resilient storage" [87] to get the best of both worlds, and our construction essentially combines the ideas behind the constructions of the above two objects.

4.2 A Reduction from Leaky Split-State to Split-State

In this section, we show how to reduce leaky split-state to split-state non-malleability. In other words, we show how to add leakage-resilience generically to any split-state non-malleable code. Our construction handles up to $\frac{1}{4}$ fraction leakage. Concretely, we prove the following lemma, where the tampering classes SS_k (split-state) and α-SS_n (leaky split-state) are defined in Definition 12 and 24 respectively.

Lemma 12. *For any constant $\alpha \in [0, 1/4)$, there is an $\left(\alpha\text{-}\mathrm{SS}_n \Rightarrow \mathrm{SS}_k, \exp(-\Omega(n))\right)$ non-malleable reduction with constant rate.*

The main theorem of this chapter is a corollary of this Lemma and Theorem 3 and Theorem 4.

Our main tool is new notion of (information-theoretic, one-time) leakage-resilient encryption defined below.

Definition 25 (Leakage-Resilient Encryption)**.** We consider a (randomized) encryption scheme $(\mathsf{Encrypt}, \mathsf{Decrypt})$ which encrypts message x of length $|x| = k$ using a key of size $|\mathsf{key}| = m$. For some message $x \in \{0,1\}^k$ we consider the following randomized experiment $\mathsf{Game}^{\mathsf{LRENC}}(x)$:

- Choose $\mathsf{key} \leftarrow \{0,1\}^m$, $\mathsf{ct} \leftarrow \mathsf{Encrypt}(\mathsf{key}, x)$.
- Alice gets ct and Bob gets key. They can run an arbitrary protocol with each other subject to the total communication being at most ℓ_1 bits. Let $\mathsf{trans} \in \{0,1\}^{\ell_1}$ be the transcript.

- At the end of the protocol, Alice also outputs an additional value $\mathsf{aux} \in \{0,1\}^{\ell_2}$.
- The output of the game is $\mathsf{key}, \mathsf{trans}, \mathsf{aux}$.

We say that an encryption scheme is $(\ell_1, \ell_2, \epsilon)$-leakage-resilient if for any adversarial strategy of Alice and Bob and for any x_0, x_1 the outputs of $\mathsf{Game}^{\mathsf{LRENC}}(x_0)$ and $\mathsf{Game}^{\mathsf{LRENC}}(x_1)$ have statistical distance at most ϵ.

The above definition is similar to the "forward-secure storage" of Dziembowski [86], which corresponds to our notion with $\ell_1 = 0$ (there is only leakage on the ciphertext; it is completely independent of the key but can be much larger than the key). It is also similar to the notion of "leakage-resilient storage" of Davi, Dziembowski and Venturi [87], which corresponds to our notion with $\ell_2 = 0$ (there is back-and-forth leakage on the ciphertext and the key but the total leakage is smaller than either the ciphertext or the key). Our definition combines the two notions. We will crucially rely on a setting of parameters where, if the key size is m and the message size is k then we need $\ell_1 < m \leq \ell_2 < k$. In other words, we allow ℓ_1 bits of back-and-forth leakage between the ciphertext and the key where ℓ_1 is smaller than either component, but then allow and additional ℓ_2 bits of leakage on the ciphertext where ℓ_2 is larger than the key.

Reduction via Leakage-Resilient Encryption. We first show how to use leakage-resilient encryption as defined above to construct a reduction from leaky split-state to split-state tampering. We do so by encrypting the two states x_L, x_R using leakage-resilient encryption and storing the key with the other state (i.e., the key used to encrypt the left state is stored on the right sides and vice versa). Intuitively, the leakage-resilient encryption ensures that the leakage is independent of the actual states x_L, x_R. However, we face the challenge that, by tampering the key on the right side we can influence how the left side is decrypted and vice versa. We get around this by thinking of the tampered keys as additional leakage (aux).

Let $\mathcal{E} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ be a leakage-resilient encryption with message size k and key length m. We define our reduction $(\mathrm{Enc}, \mathrm{Dec})$ below:

- $\mathrm{Enc}(x_L, x_R)$: Sample $\mathsf{key}_L \leftarrow \{0,1\}^m, \mathsf{ct}_L \leftarrow \mathsf{Encrypt}(x_L)$, $\mathsf{key}_R \leftarrow \{0,1\}^m, \mathsf{ct}_R \leftarrow \mathsf{Encrypt}(x_R)$. Output (y_L, y_R) where $y_L = (\mathsf{ct}_L, \mathsf{key}_R)$, $y_R = (\mathsf{ct}_R, \mathsf{key}_L)$.

- $\mathrm{Dec}(y_L, y_R)$ to parse $y_L = (\mathsf{ct}_L, \mathsf{key}_R)$, $y_R = (\mathsf{ct}_R, \mathsf{key}_L)$ and output $x_L = \mathsf{Decrypt}(\mathsf{key}_L, \mathsf{ct}_L)$ and $x_R = \mathsf{Decrypt}(\mathsf{key}_R, \mathsf{ct}_R)$.

Lemma 13. *Assume $\mathcal{E}$ is an $(\ell_1, \ell_2, \epsilon)$-leakage-resilient encryption with message length k, key length $m \leq \ell_2$ and ciphertext length c. Then* $(\mathrm{Enc}, \mathrm{Dec})$ *defined above is a 2ϵ-non-malleable reduction from $(\ell_1/(c+m))$-leaky split-state to split-state. For a messages (x_L, x_R) of length $2k$, the resulting codeword* $\mathrm{Enc}(x_L, x_R)$ *has length $2(c+m)$*

Proof. Consider the following game: $\mathsf{Game}^{\mathsf{NM}}(x_L, x_R)$:

1. Compute $(y_L = (\mathsf{ct}_L, \mathsf{key}_R), y_R = (\mathsf{ct}_R, \mathsf{key}_L)) \leftarrow \mathrm{Enc}(x_L, y_L)$

2. Give Alice y_L and Bob y_R. They can run an arbitrary protocol with each other subject to the total communication being at most ℓ_1 bits. Let $\mathsf{trans} \in \{0,1\}^{\ell_1}$ be the transcript.

3. At the end of the protocol Alice outputs $y'_L = (\mathsf{ct}'_L, \mathsf{key}'_R)$ and Bob outputs $y'_R = (\mathsf{ct}'_R, \mathsf{key}'_L)$.

4. The output of the game $(x'_L, x'_R) = \mathrm{Dec}(y'_L, y'_R)$ so that $x'_L = \mathsf{Decrypt}(\mathsf{key}'_L, \mathsf{ct}'_L)$ and $x'_R = \mathsf{Decrypt}(\mathsf{key}'_R, \mathsf{ct}'_R)$.

To prove the Lemma, we fix an arbitrary strategy of Alice and Bob and need to show that there exist some distribution G over functions (g_L, g_R) such that for every x_L, x_R the output of $\mathsf{Game}^{\mathsf{NM}}(x_L, x_R)$ is 2ϵ-statistically close to $g_L(x_L), g_R(x_R)$ where $(g_L, g_R) \leftarrow G$.

Let us define the distribution $z \leftarrow D(x_L, x_R)$ to be the distribution of the values

$$z = (\mathsf{key}_L, \mathsf{key}_R, \mathsf{trans}, \mathsf{key}'_L, \mathsf{key}'_R)$$

in the context of $\mathsf{Game}^{\mathsf{NM}}(x_L, x_R)$. We make two observations, which we then combine to prove our lemma.

Observation 1. In $\mathsf{Game}^{\mathsf{NM}}(x_L, x_R)$, if we condition on some particular choice of $z \leftarrow D(x_L, x_R)$, then the distribution of x'_L is a completely independent of (x_R, x'_R) and similarly x'_R is independent of (x_L, x'_L). In particular, we can define the randomized process g^z_L which has z hard-coded and samples $x'_L \leftarrow g^z_L(x_L)$ by first sampling Alice's view in the game conditioned on z and x_L, computing her output ct'_L and setting $x'_L = \mathsf{Decrypt}(\mathsf{key}'_L, \mathsf{ct}'_L)$. We can define the randomized process $x'_R \leftarrow g^z_R(x_R)$ analogously. It is easy to see that the distribution of $\mathsf{Game}^{\mathsf{NM}}(x_L, x_R)$ is then identical to sampling $z \leftarrow D(x_L, x_R)$ and outputting $x'_L \leftarrow g^z_L(x_L), x'_R \leftarrow g^z_R(x_R)$.

Observation 2. For any x_L, x_R the distribution of $D(x_L, x_R)$ is 2ϵ-statistically close to $D(0^k, 0^k)$. We argue that this holds via two steps. We first argue that $D(x_L, x_R)$ is ϵ-close to $D(0^k, x_R)$ and then argue that $D(0^k, x_R)$ is ϵ-close to $D(0^k, 0^k)$. For the first step, we can fix any worst case choice of $\mathsf{key}_R, \mathsf{ct}_R$ and use the security of leakage-resilient encryption to argue that the joint distribution of $\mathsf{key}_L, \mathsf{trans}, \mathsf{key}'_R$ is ϵ-close between $D(x_L, x_R)$ and $D(0^k, x_R)$; we set $\mathsf{aux} = \mathsf{key}'_R$ in this argument and use the fact that $|\mathsf{key}'_R| = m \leq \ell_2$. We then note that key'_L is just a function of $\mathsf{ct}_R, \mathsf{key}_L$ and trans and therefore the total distribution of $(\mathsf{key}_L, \mathsf{key}_R, \mathsf{trans}, \mathsf{key}'_L, \mathsf{key}'_R)$ is ϵ close between $D(x_L, x_R)$ and $D(0^k, x_R)$. The argument that $D(0^k, x_R)$ is ϵ-close to $D(0^k, 0^k)$ is identical.

By combining observations 1 and 2, we see the distribution of $\mathsf{Game}^{\mathsf{NM}}(x_L, x_R)$ is 2ϵ statistically close to sampling $z \leftarrow D(0^k, 0^k)$ and outputting $x'_L \leftarrow g^z_L(x_L), x'_R \leftarrow g^z_R(x_R)$. This concludes our reduction as desired; we define the distribution G over functions (g_L, g_R) by sampling $z \leftarrow D(0^k, 0^k)$ and setting $g_L = g^z_L$ and $g_R = g^z_R$. The output of $\mathsf{Game}^{\mathsf{NM}}(x_L, x_R)$ is 2ϵ-statistically close to $g_L(x_L), g_R(x_R)$ where $(g_L, g_R) \leftarrow G$. □

Construction of Leakage-Resilient Encryption. Let Ext be a seeded strong-extractor with r-bit source, d-bit seed and output size k which is $(r - \ell_1 - \ell_2, \epsilon_1)$-secure. Let $2\mathsf{Ext}$ be a strong two-source extractor with m-bit sources and d-bit output which is $(2m - \ell_1, \epsilon_2)$-secure.

Define the scheme $(\mathsf{Encrypt}, \mathsf{Decrypt})$ as follows:

- $\mathsf{Encrypt}(\mathsf{key}, x)$: Choose $u \leftarrow \{0,1\}^r, y \leftarrow \{0,1\}^m, s = 2\mathsf{Ext}(\mathsf{key}, y), z = \mathsf{Ext}(u; s) \oplus x$. Output $\mathsf{ct} = (u, y, z)$.

- $\mathsf{Decrypt}(\mathsf{key}, \mathsf{ct} = (u, y, z))$ to compute $s = 2\mathsf{Ext}(\mathsf{key}, y)$ and output $z \oplus \mathsf{Ext}(u; s)$.

Claim 3. Consider a variant of the leakage-resilient encryption game, which we denote "weak leakage resilience", where Alice does not get the z part of the ciphertext during the game but the output of the game is $\mathsf{key}, \mathsf{trans}, \mathsf{aux}, z$. If the scheme is $(\ell_1, \ell_2, \epsilon)$-"weak leakage resilient" then it also satisfies $(\ell_1, \ell_2, \epsilon \cdot 2^k)$-leakage resilience.

Proof. Assume there exists some (Alice, Bob, Distinguisher) strategy in the original game such that the Distinguisher has an $\epsilon 2^k$ advantage in distinguishing the game with x_0 and x_1. We convert this into an (Alice', Bob, Distinguisher') strategy for the weak game by guessing a random value $v \leftarrow \{0,1\}^k$ at the beginning of the game and having Alice' run Alice with v in place of z. Then Distinguisher' gets z and if $v = z$ it runs the original Distinguisher else outputs 0. It's easy to see that the advantage of Distinguisher' is the same as that of Distinguisher when $v = z$, which happens with probability 2^{-k}, and 0 otherwise. Therefore, Distinguisher' has advantage 2^{-k} smaller than Distinguisher which proves the claim. □

Lemma 14. *The scheme* $(\mathsf{Encrypt}, \mathsf{Decrypt})$ *is* $(\ell_1, \ell_2, (\epsilon_1 + \epsilon_2)2^{k+1})$*-leakage-resilient.*

Proof. It suffices to show that the scheme is $(\ell_1, \ell_2, 2(\epsilon_1 + \epsilon_2))$ weak leakage resilient and the rest follows by the preceding claim. We use a statistical hybird argument.

Hybrid 1: This is the weak leakage resilient game with message x_0. Recall that in the game Alice gets (u, y) and Bob gets key. They run a protocol with ℓ_1 bits of communication resulting in transcript trans. At the termination of the protocol, Bob also outputs an additional ℓ_2-bit value aux. The output of the protocol is $\mathsf{key}, \mathsf{trans}, \mathsf{aux}, z$ where $z = \mathsf{Ext}(u; s) \oplus x_0$ and $s = 2\mathsf{Ext}(\mathsf{key}, y)$.

Hybrid 2: Note that, in Hybrid 1, conditioned on the random variable $V_1 = (s, y, \mathsf{trans})$ the random variables $V_2 = (u, \mathsf{aux}, z)$ and $V_3 = \mathsf{key}$ are independent. Therefore, we can define

Hybrid 2 to run the same game as Hybrid 1, which defines (V_1, V_2, V_3), but then, instead of key, output a freshly re-sampled key' from the correct distribution of V_3 conditioned on V_1. This is distributed identically to Hybrid 1.

Hybrid 3: In this hybrid, we choose s uniformly at random instead of $s = 2\mathsf{Ext}(\mathsf{key}, y)$ and set $z = \mathsf{Ext}(r; s) \oplus x_0$. We still sample key' from the same distribution of V_3 conditioned on V_1 at the end of the game, just like in Hybrid 2.

The statistical distance between Hybrid 2 and Hybrid 3 is ϵ_2. We rely on the fact that $2\mathsf{Ext}$ is a strong extractor and that trans amounts to ℓ_1 bits of entropy loss from key to argue that, even given (u, y, trans) the value s is ϵ_2-close to uniform.

Hybrid 4: In this hybrid, we set z to uniform instead of $z = \mathsf{Ext}(u; s) \oplus x_0$.

The statistical distance between Hybrid 3 and Hybrid 4 is ϵ_1. This follows from the strong-extractor property of Ext and the fact that $\mathsf{trans}, \mathsf{aux}$ gives $\ell_2 + \ell_1$ bits of leakage on u.

Hybrid 5,6,7: Are the same as 3,2,1 with x_0 replaced by x_1. In particular Hybrid 7 is the weak leakage resilience game with message x_1.

Hybrids 4,5 are ϵ_1 close (same argument as Hybrids 3,4), Hybrids 5,6 are ϵ_2 close (same argument as Hybrids 2,3) and Hybrids 6,7 are identical (same argument as 1,2).

Combining the above we get a total distance of $2(\epsilon_1 + \epsilon_2)$ between Hybrids 1 and 7 as we wanted to show. □

We can now plug in parameters using the inner-product two-source extractor [48], and the strong extractor [46] to prove the main Lemma of this section:

Proof of Lemma 12. For $\epsilon \in (0, 1)$, let $\epsilon_1 = \epsilon_2 = \epsilon/2^{k+3}$ and $\ell_2 = m$. By Theorem 9, there exists some constant $c_1, c_2 \geq 1$ and explicit Ext such that for $r - \ell_1 - \ell_2 \geq c_1 \cdot k$, Ext can extract k bits from $(r, r - \ell_1 - \ell_2)$ source with $d = c_2 \log(r/\epsilon_1)$-bit seed and error ϵ_1. By Theorem 10, for $2m - \ell_1 \geq m + d + 2\log(1/\epsilon_2)$, there exists explicit $2\mathsf{Ext}$ that extracts d bits with error ϵ_2 from

m-bit sources with entropy $2m - \ell_1$. Plugging in Ext and $2\mathsf{Ext}$, by Lemma 14 and Lemma 13, we obtain $(\ell_1, \ell_2, (\epsilon_1 + \epsilon_2)2^{k+1})$ leakage-resilient encryption and a $(\epsilon_1 + \epsilon_2)2^{k+2}$-non-malleable reduction from (ℓ_1/n)-split state to split state with $n = r + 2m + k$. By setting $r = \ell_1 + m + c_2 k$, $m = \ell_1 + d + 2\log(1/\epsilon_2)$ and $d = c_2(\log(r/\epsilon_1))$, we obtain that $n \leq 4\ell_1 + c_3(k + \log 1/\epsilon)$ for some constant $c_3 \geq 1$. Therefore for any $\alpha < 1/4$ and $\ell_1 = \alpha n$, we can set $n = \Theta(\frac{k+\log(1/\epsilon)}{1/4-\alpha})$. The desired conclusion follows from setting $\epsilon = \exp(-\Omega(k))$ and $n = \Theta(k)$. □

Chapter 5: Local (Junta) Tampering

In this chapter, we construct codes that are non-malleable with respect to *local* tampering: each output bit only depends on a few inputs (importantly, each output may depend on *different inputs*). More formally,

Definition 26. We say that a bit x_i *affects* the boolean function f,
if $\exists \{x_1, x_2, \cdots x_{i-1}, x_{i+1}, \cdots x_n\} \in \{0,1\}^{n-1}$ such that,
$f(x_1, x_2, \cdots x_{i-1}, 0, x_{i+1}, \cdots x_n) \neq f(x_1, x_2, \cdots x_{i-1}, 1, x_{i+1}, \cdots x_n)$.

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

Definition 27 (Locality)**.** A function $f : \{0,1\}^n \to \{0,1\}^n$ is said to have

- *Output locality m* if every output bit f_i is dependent on at most m input bits.
- *Input locality* ℓ if every input bit f_i affects at most ℓ output bits.

We take Local_ℓ^m to denote the class of functions with input locality ℓ and output locality m, and let Local^m denote the class of functions with output locality m (and no restriction on input locality).

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

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

The class of bounded-depth bounded-fan-in circuits is natural both as a complexity class and in the context of practical tampering attacks, where it seems to realistically capture the capabilities

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

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

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

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

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

While our result for NC^0correspond to constant depth circuits, the first corollary above implies as a special case that the code is also non-malleable against any $\delta \log n$ depth NC circuit, for any

constant $0 \leq \delta < 1$. Note that, since separations between P and NC^1 are not known, constructing (unconditional) non-malleable codes against NC^1 is unlikely, since an attacker in P can always decode and re-encode a related message, thus immediately breaking non-malleability.

5.1 Technical Overview

We give a high level technical overview of our construction. We use as an underlying tool a form of binary ramp secret sharing called "Reconstructable Probabilistic Encoding" (RPE) or "Linear Error Correcting Secret Sharing" (Def. 19), a code that can correct a constant fraction of errors (denoted c^{err}), and enjoys some additional secret-sharing like properties: any (smaller) constant fraction c^{sec} of a codeword reveals no information about the encoded message, and can in fact be completed to a (correctly distributed) encoding of any message. This or similar tools have utilized elsewhere either implicitly or explicitly, e.g., the construction based on Reed Solomon codes and Shamir secret sharing with Berlekamp-Welch correction, as used already in [88] is a RPE (any small enough subset of shares is distributed uniformly at random, and any such collection of shares can be extended to be the sharing of any message of our choice). Other RPE schemes with possibly improved parameters can be constructed from, e.g., [69, 70, 67, 68].

5.1.1 Handling Local Functions

Local functions are functions that have both small input and small output locality (i.e. each input bit affects a small number of output bits and each output bit depends on a small number of input bits). Our goal is to show a *non-malleable reduction* from a class of local functions with appropriate parameters, to the class of split-state functions. Loosely speaking, a non-malleable reduction from a class $\mathcal{F}$ to a class $\mathcal{G}$, is a pair (E, D) of encoding/decoding functions along with a reduction that transforms every $f \in \mathcal{F}$ into a distribution G_f over functions $g \in \mathcal{G}$, such that for every x, the distributions $\mathsf{D}(f(\mathsf{E}(x)))$ and $G_f(x)$ are statistically close. In the case of reductions to split-state, we let $x = (\mathsf{L}, \mathsf{R})$ where $\mathsf{L}, \mathsf{R} \in \{0,1\}^k$. We want to construct (E, D) such that, informally, given any local f, the effect of applying f to the encoding $\mathsf{E}(x)$ and then

decoding $\mathsf{D}(f(\mathsf{E}(x)))$, can be simulated by applying some split state function $g = (g_L, g_R)$ directly to $x = (\mathsf{L}, \mathsf{R})$.

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

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

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

A first (failed) attempt would be to just make sure that the encoding on the right can correct up to $n_L\ell_i$ errors, and hope that we can therefore set s^{L} arbitrarily when computing f^R and the resulting encoding would still be decoded to the same initial value R. While this argument works if the only changes made to s^{R} (a valid codeword) are caused by the "crossing bits" from s^{L}, it fails to take into account that f^R can in fact apply other changes inside s^{R}, and so it could be that s^{R} is malformed in such a way that applying f^R will cause it to decode differently in a way that

crucially depends on s^{L}. The issue here seems to be that there is an exact threshold for when the decoding algorithm succeeds or not, and thus the function can be designed so that f^R is just over or under the threshold depending on the left side.

To overcome this problem, we use randomized decoding and a "consistency check" technique introduced in [67], and a forthcoming version by the same authors [68], in a different context. Roughly speaking, we make the right side encoding redundant, so that any large enough subset of bits is enough to recover R. An RPE has this property, due its error correction capabilities. The decoding algorithm will decode via the first such subset, but will check a *random* subset of bits were consistent with a particular corrected codeword. This will yield similar behavior, regardless of which subset is used to decode. This construction has various subtleties, but they are all inherited from previous work, so we do not explain them here. The main point is that, like in the followup to [67], while the real decoding algorithm uses the first large enough subset, its output can be simulated by using any other large enough subset.

Now, if we use ℓ_{sec} rows (a parameter), then given f we can find a row with at most $(n_L \ell_i)/\ell_{\mathsf{sec}}$ bits from s^{L} influencing it. We will simulate the decoding using this row, plus the random subset of columns needed for the column check. We will use a RPE on the left and set all the parameters such that the overall number of bits on the left that influences either this row (can be bounded using ℓ_i on the left) or the column check (can be bounded using ℓ_o on the right) is low enough (below $c^{\mathsf{sec}} n_L$), and so can again be fixed to some random values independently of L, and f^R can be turned local.

Putting the above together will indeed yield what we are looking for, namely a non-malleable reduction from local to split state functions, albeit with worse parameters than the construction we provide. We note that the proof above in fact works for a more general class of functions (a fact we will use in our second construction). In particular, the first part requires a limit on the output locality of f^L, and the second part requires a limit on the output locality of f^R and the input locality of f^L, where all of these only refer to "cross-over" influences (within each part separately f can be arbitrary). Moreover, due to our use of encoding, security is maintained even with leakage, as

long as the leakage is a constant fraction of bits on the left and a constant fraction on the right, independently. Similarly, security is maintained even when a constant fraction of bits on the left do not adhere to the input locality bound.

5.1.2 Removing Input Locality

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

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

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

string so that it "knows" which tampered output bits on the left, $\widetilde{X}^{\mathsf{L}}$, are relevant.

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

Moreover, because the tampering attacker is very limited, in the sense that it must choose the tampering function before learning any information about the output of the PRG, we are able to show that Nisan's PRG (see Definition 21), an *unconditional* PRG is sufficient for our construction. Thus, our construction does not rely on any computational assumption.

5.1.3 On tampering functions where each output symbol depends on $n - \log(1/(4\epsilon))$ input symbols

At this point, we have shown explicit non-malleable codes for tampering functions with locality n^{δ}, for constant $\delta < 1$. The size of the class of all output-local tampering functions (with locality sufficiently smaller than n) is also bounded in size and therefore NMC for this class follow from the existential results of [42]. A natural question is how large can the output-locality be?

We prove the impossibility of ϵ-non-malleable codes (see Definition 2) for the class of $(n - \log(1/(4\epsilon)))$-output-local tampering functions. In addition to the above motivation, parity over n bits is average-case hard for this class.[1] Therefore, our impossibility result can be viewed as a "separation" between average-case hardness and non-malleability, as it exhibits a class for which we have average-case hardness bounds, but cannot construct non-malleable codes for. Furthermore, the class $\mathcal{F}'$ constructed in our lower bound proof has size only $4^n \cdot 2^{2^{n-\log(1/(4\epsilon))}}$, which in turn means that $\log\log|\mathcal{F}'| = n - \log(1/(4\epsilon)) = n - \log(1/\epsilon) + 2$. On the other hand, the aforementioned result of Dziembowski et al. [42] shows existence of an ϵ-non-malleable code for any

[1] Note that, even arbitrary decision trees of depth $n - 1$ have no advantage in computing the parity of n bits with respect to the uniform distribution. [89]

class $\mathcal{F}$ such that $\log\log|\mathcal{F}| \leq n - 2\log(1/\epsilon)$. Thus, our lower bound result is close to matching the existential upper bound.

5.1.4 On tampering functions where each input symbol affects at most one output symbol

A natural question is whether non-malleable codes can be constructed for the class of input-local functions, where for codeword length n, each input bit affects $\ll n$ output bits.

We conclude this chapter by showing a strongly negative answer to this question. We show that even achieving NMC for 1-input local functions (where each input bit affects at most one output bit) is impossible. In fact, our proof shows an even stronger result: the impossibility holds even if each input symbol can only affect the same single output symbol. That is, it is impossible to construct NMC for the tampering class that allows to change only one codeword symbol in a way that depends on the input (while the other symbols may be changed into constant values). Stated like this, this result can also be viewed as an extension of our first result above on the maximum number of symbols that can be modified in a non-malleable code.

5.1.5 Remark on deterministic vs. randomized decoding.

The standard (and original) definition of NMC requires deterministic decoding and perfect correctness, although some later work took advantage of randomized decoding.[2] We note that our lower bound for the class of input-local functions holds for standard schemes (with deterministic decoding and perfect correctness). Our lower bound (with $\epsilon = \frac{1}{4n}$) for the class of $n - \log(n)$ output-local functions holds even for coding schemes that have *randomized* decoding and perfect correctness. The lower bound for the class of functions that change $d/2$ symbols holds even for coding schemes with *randomized* decoding and *imperfect* correctness.

[2]For the class of output-local functions (where each output depends on at most ℓ inputs) we have constructed non-malleable codes with randomized decoding (and $\epsilon = \text{negl}(n)$) for $\ell < n/\log n$ [2], whereas constructions with deterministic decoding are known for locality up to $n^{1/2-\epsilon}$ for small ϵ. [15, 90].

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

Theorem 22. (E, D) *is a* $(\mathrm{Local}^{\ell_o(k)}_{\ell_i(k)} \Rightarrow \mathrm{SS}_k, \mathrm{negl}(k))$*-non-malleable reduction given the following parameters for* $\mathrm{Local}^{\ell_o(k)}_{\ell_i(k)}$ *(where* $c^{\mathsf{rate}}, c^{\mathsf{err}}, c^{\mathsf{sec}}$ *are taken from lemma 6):*

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

Putting together Theorem 22 with Theorems 1 and 4, we obtain the following.

Corollary 3. $(\mathsf{E} \circ \mathsf{E}_{\mathrm{SS}}, \mathsf{D}_{\mathrm{SS}} \circ \mathsf{D})$ *is a* $(\mathrm{Local}^{\ell}_{\ell}, k, \mathrm{negl}(k))$*-non-malleable code with rate* $\Theta(1/\ell)$*, where* $\ell = \tilde{O}(\sqrt{n})$.

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

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

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

We construct an encoding scheme (E, D) summarized in Figure 5.1 and parametrized below. We then show that the pair (E, D) is an $(\mathrm{Local}^{\ell_o(k)}_{\ell_i(k)}, \mathrm{SS}_k, \mathrm{negl}(k))$-non-malleable reduction. This

immediately implies that given a non-malleable encoding scheme $(\mathsf{E}^{\mathsf{ss}}, \mathsf{D}^{\mathsf{ss}})$ for class SS_k (where SS is the class of split-state functions), the encoding scheme scheme $(\mathsf{E}^{\mathsf{bd}}, \mathsf{D}^{\mathsf{bd}})$, where $\mathsf{E}^{\mathsf{bd}}(m) := \mathsf{E}(\mathsf{E}^{\mathsf{ss}}(m))$ and $\mathsf{D}^{\mathsf{bd}}(\vec{s}) := \mathsf{D}^{\mathsf{ss}}(\mathsf{D}(\vec{s}))$ yields a non-malleable code against $\mathrm{Local}^{\ell_o(k)}_{\ell_i(k)}$.

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

- $(\mathrm{E}_L, \mathrm{D}_L)$ parametrized by $(k, n_L, c_L^{\mathsf{err}}, c_L^{\mathsf{sec}}) := (k, c^{\mathsf{rate}}k, c^{\mathsf{err}}, c^{\mathsf{sec}})$ where $c^{\mathsf{err}}, c^{\mathsf{sec}}, c^{\mathsf{rate}}$ are taken from lemma 6.
- $n_{\mathsf{check}} := \log^2(k)$.
- $\ell_{\mathsf{sec}} := \sqrt{\frac{cn_L}{n_{\mathsf{check}}}} = \Theta(\frac{\sqrt{k}}{\log(k)})$.
- $(\mathrm{E}_R, \mathrm{D}_R)$ parametrized by $(k, n_R, c_R^{\mathsf{err}}, c_R^{\mathsf{sec}}) := (k, \frac{\ell_o c^{\mathsf{rate}}k}{c^{\mathsf{sec}}}, c^{\mathsf{err}}, c^{\mathsf{sec}})$.
- $n := \ell_o c^{\mathsf{rate}}k + c^{\mathsf{rate}}k = O(\frac{k^2}{\log^2(k))}$.

Note that this setting of parameters is taken with our forthcoming reduction in mind. (See Corollary 4 and Theorem 23.) One may take any parametrization for which (a) such RPEs exist, (b) $(1 - c^{\mathsf{err}}/4)^{n_{\mathsf{check}}}$ is negligible in k, and (c) Observation 1 (below) is satisfied. For certain applications, parametrization other than ours may be advantageous.

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

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

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

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

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

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

1. Let $(\vec{\sigma}^{\mathsf{L}}, \vec{\sigma}^{\mathsf{R}}) := ([\sigma_i^{\mathsf{L}}]_{i\in[n_L]}, [\sigma_i^{\mathsf{R}}]_{i\in[n_R]})$.
2. Compute $((w_1^{\mathsf{L}}, \ldots, w_{n_L}^{\mathsf{L}}), \mathsf{L}) \leftarrow \mathrm{D}_L(\sigma_1^{\mathsf{L}}, \ldots, \sigma_{n_L}^{\mathsf{L}})$. If the decoding fails, set $\mathsf{L} := \perp$.
3. (`decoding-check on right`) Let $t := \lceil n_R(1 - c_R^{\mathsf{err}}/4) \rceil$ Define $\vec{\sigma}'^{\mathsf{R}} := \sigma_1'^{\mathsf{R}}, \ldots, \sigma_{n_R}'^{\mathsf{R}}$ as follows: Set $\sigma_\ell'^{\mathsf{R}} := \sigma_\ell^{\mathsf{R}}$ for $\ell = 1, \ldots, t$. Set $\sigma_\ell'^{\mathsf{R}} := 0$ for $\ell = t+1, \ldots, n_R$. Compute $((w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}}), \mathsf{R}) \leftarrow \mathrm{D}_R(\sigma_1'^{\mathsf{R}}, \ldots, \sigma_t'^{\mathsf{R}})$. If the decoding fails or $(w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}})$ is not $c_R^{\mathsf{err}}/4$-close to $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{t_R}^{\mathsf{R}})$, set $\mathsf{R} := \perp$.
4. (`codeword-check on right`) Pick a random subset $R_{\mathsf{check}} \subset [n_R]$ of size $n_{\mathsf{check}} < c_R^{\mathsf{sec}} \cdot n_R$. For all $\ell \in R_{\mathsf{check}}$, check that $\sigma_\ell^{\mathsf{R}} = w_\ell^{\mathsf{R}}$. If the check fails, set $\mathsf{R} := \perp$.
5. (`output`) Output $x := (\mathsf{L}, \mathsf{R})$.

Figure 5.1: THE $(\mathrm{Local}_{\ell_i(k)}^{\ell_o(k)}, \mathrm{SS}, \mathrm{negl}(k))$-NON-MALLEABLE REDUCTION (E, D)

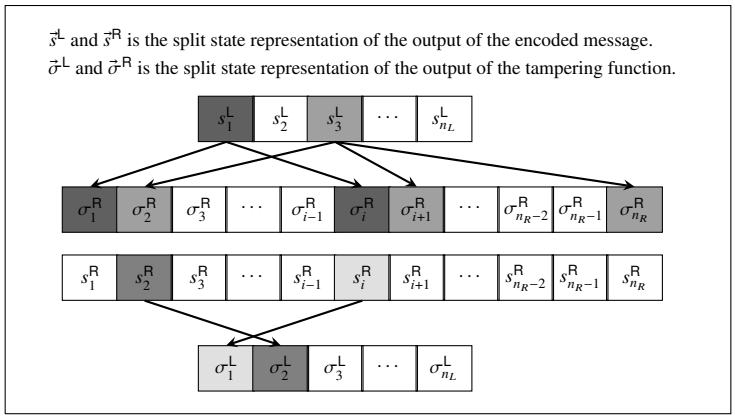

Figure 5.2: The adversary chooses tampering function $f = (f^{\mathsf{L}}, f^{\mathsf{R}}) \in \mathrm{Local}^{\ell_o(k)}_{\ell_i(k)}$ which takes inputs $(\vec{s}^{\mathsf{L}}, \vec{s}^{\mathsf{R}})$ and produces outputs $(\vec{\sigma}^{\mathsf{L}}, \vec{\sigma}^{\mathsf{R}})$. The highlighted bits of $\vec{s}^{\mathsf{L}}$ and $\vec{s}^{\mathsf{R}}$ are the "bad" bits. E.g. note that bits s^{R}_2 and s^{R}_i *affect* the output bits σ^{L}_2 and σ^{L}_1 respectively after f^{L} is applied to $(\vec{s}^{\mathsf{L}}, \vec{s}^{\mathsf{R}})$. Thus we add 2 and i to the set $\mathcal{S}_{\mathsf{R}\to\mathsf{L}}$. Similarly, the bits s^{L}_1 and s^{L}_3 *affect* the bits $\{\sigma^{\mathsf{R}}_1, \sigma^{\mathsf{R}}_i\}$ and the bits $\{\sigma^{\mathsf{R}}_2, \sigma^{\mathsf{R}}_{i+1}, \sigma^{\mathsf{R}}_{n_R,}\}$ respectively after the tampering function f^{R} is applied to $(\vec{s}^{\mathsf{L}}, \vec{s}^{\mathsf{R}})$. We therefore add 1 to the sets $\mathcal{S}^1_{\mathsf{L}\to\mathsf{R}}$ and $\mathcal{S}^i_{\mathsf{L}\to\mathsf{R}}$, while we add 3 to the sets $\mathcal{S}^2_{\mathsf{L}\to\mathsf{R}}, \mathcal{S}^{i+1}_{\mathsf{L}\to\mathsf{R}}$ and $\mathcal{S}^{n_R}_{\mathsf{L}\to\mathsf{R}}$. We also add both 1 and 3 to the set $\mathcal{S}_{\mathsf{L}\to\mathsf{R}}$.

The sets defined above are illustrated in Figure 5.2. We observe the following immediate facts about their sizes:

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

1. There is some set $J^* \subset [n_R]$ such that $|J^*| = t$ and $|\mathcal{S}^{J^*}_{\mathsf{L}\to\mathsf{R}}| = 0$ (from now on, J^* denotes the lexicographically first such set).

   (Since $|\mathcal{S}_{\mathsf{L}\to\mathsf{R}}| \le \ell_i \cdot n_L \le n_R - t$.)

2. By choice of parameters $n_L, n_{\mathsf{check}}, c^{\mathsf{sec}}_L$, we have that $|\mathcal{S}_{\mathsf{check}}| \le n_L \cdot c^{\mathsf{sec}}_L$.

   (Since $\mathcal{S}_{\mathsf{check}} \le \ell_o \cdot n_{\mathsf{check}}$.)

3. By choice of parameters $n_L, n_R, c^{\mathsf{sec}}_R$, we have that $|\mathcal{S}_{\mathsf{R}\to\mathsf{L}}| \le \ell_o \cdot n_L \le n_R \cdot c^{\mathsf{sec}}_R$.

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

- Choose a random subset $R_{\mathsf{check}} \subseteq [n_R]$ of size n_{check}.
- Choose vectors $I^{\mathsf{L}} \in \{0,1\}^{n_L} \times \{*\}^{n_L}$, $I^{\mathsf{R}} \in \{*\}^{n_L} \times \{0,1\}^{n_R}$ uniformly at random.

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

  $g_L(\mathsf{L})$:

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

  $g_R(\mathsf{R})$:

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

- Output $g = (g_L, g_R)$.

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

By the definition of a non-malleable reduction (Definition 5), in order to complete the proof of Theorem 22, we must show that (E, D) have the following properties:

1. For all $x \in \{0,1\}^k$, we have $\mathsf{D}(\mathsf{E}(x)) = x$ with probability 1.

2. For all $f \in \mathrm{Local}^{\ell_o}_{\ell_i}$,

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

where G_f is the distribution defined above.

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

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

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

1. (`plain decode on left`) Let $(\vec{\sigma}^{\mathsf{L}}, \vec{\sigma}^{\mathsf{R}}) := ([\sigma^{\mathsf{L}}_i]_{i\in[n_L]}, [\sigma^{\mathsf{R}}_\ell]_{\ell\in[n_R]})$.
   Compute $((w^{\mathsf{L}}_1, \ldots, w^{\mathsf{L}}_{n_L}), \mathsf{L}) \leftarrow \mathrm{D}_L(\sigma^{\mathsf{L}}_1, \ldots, \sigma^{\mathsf{L}}_{n_L})$. If the decoding fails, set $\mathsf{L} := \perp$.

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

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

4. (`output`) Output $x := (\mathsf{L}, \mathsf{R})$.

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

Claim 4.

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

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

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

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

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

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

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

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

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

$$\Pr[\mathsf{D} \restriction_{\vec{\sigma}^{\mathsf{L}}} (\vec{\sigma}^{\mathsf{R}}) = \mathsf{D}' \restriction_{\vec{\sigma}^{\mathsf{L}}} (\vec{\sigma}^{\mathsf{R}})] \geq 1 - \mathrm{negl}(n), \tag{5.1}$$

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

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

To analyze (5.1), we define the following set of instances (which intuitively corresponds to the set of instances on which both $\mathsf{D} \restriction_{\vec{\sigma}^{\mathsf{L}}}$ and $\mathsf{D}' \restriction_{\vec{\sigma}^{\mathsf{L}}}$ are likely to output $\perp$).

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

So, now consider the two cases:

- Suppose $\vec{\sigma}^{\mathsf{R}} \in \Pi_{\perp}$. Then, both $\mathsf{D}(\vec{\sigma}^{\mathsf{R}})$ and $\mathsf{D}'(\vec{\sigma}^{\mathsf{R}})$ will fail the codeword-check with probability $\geq 1 - \rho(n_R, \delta, n_{\mathsf{check}})$.

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

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

□

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

- Choose a random subset $R_{\mathsf{check}} \subseteq [n_R]$ of size n_{check}.
- Choose vectors $I^{\mathsf{L}} \in \{0,1\}^{n_L}$, $I^{\mathsf{R}} \in \{0,1\}^{n_R}$ in the following way: $I^{\mathsf{L}} \leftarrow \mathrm{E}_L(\mathsf{L})$, $I^{\mathsf{R}} \leftarrow \mathrm{E}_R(\mathsf{R})$.

- Let J^* be the subset of $[n_R]$ as described in Observation 1.

- The split-state tampering function $g := (g_L, g_R) \in \mathrm{SS}_k$ has $I^{\mathsf{L}}, I^{\mathsf{R}}$ hardcoded into it and is specified as follows:

  $g_L(\mathsf{L})$:

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

2. (`output`) Output $\widetilde{\mathsf{L}}$.

$g_R(\mathsf{R})$:

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

- Output $g = (g_L, g_R)$.

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

Claim 5.

$$H_1 \equiv H_2.$$

The claim can be verified by inspection.

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

Claim 6.

$$H_2 \equiv H_3.$$

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

5.2.1 Extending to Leaky Local

The construction from Section 5.2 is actually secure against a slightly larger class of tampering functions beyond $\text{Local}_{\ell_i}^{\ell_o}$ functions, which we call LLocal, or "Leaky Local." Notice that the parameters given above (as in observation 1) in fact yield:

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

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

Definition 28. Let $\text{LLocal} \subseteq \{\{0,1\}^{n_L} \times \{0,1\}^{n_R} \to \{0,1\}^{n_L} \times \{0,1\}^{n_R}\}$, *Leaky Local*, be the set of functions $\{\psi_{f,h_1,h_2}\}$, parametrized by functions (f, h_1, h_2), where

$$\psi_{f,h_1,h_2}(\vec{s}^{\mathsf{L}}, \vec{s}^{\mathsf{R}}) := C_{univ}(f(h_1(\vec{s}^{\mathsf{L}}), h_2(\vec{s}^{\mathsf{R}})), \vec{s}^{\mathsf{L}}, \vec{s}^{\mathsf{R}})$$

, f outputs a circuit C and C_{univ} is a universal circuit that computes the output of the circuit C on input $(\vec{s}^{\mathsf{L}}, \vec{s}^{\mathsf{R}})$. Moreover, we require that f, h_1, h_2 have the following form:

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

The following corollary can be easily verified.

Corollary 4. $(\mathsf{E} \circ \mathsf{E}_{\mathrm{SS}}, \mathsf{D}_{\mathrm{SS}} \circ \mathsf{D})$ *is an* $(\text{LLocal}, \mathrm{SS}_k, \mathrm{negl}(k))$*-non-malleable reduction.*

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

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

Theorem 23. $(\mathsf{E}', \mathsf{D}')$ *is a* $(\text{Local}^{\ell_o'} \Rightarrow \text{LLocal}, \text{negl}(n))$*-non-malleable reduction given the following parameters for* $\text{Local}^{\ell_o'}$*:*

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

We construct an encoding scheme $(\mathsf{E}', \mathsf{D}')$ summarized in Figure 5.3 and parametrized below. In brief, our encoding simply distributes the bits of the left input pseudorandomly in a string comparable in length to the right input. We then append a short description of where the encoding is hiding, a seed to pseudorandom generator.

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

Corollary 5. $(\mathsf{E}', \mathsf{D}')$ *yields, with previous results, a* $(\text{Local}^{\tilde{O}(\sqrt{n})}, k, \text{negl}(k))$*-non-malleable reduction with sublinear rate, where* $n = \Theta(\frac{k^2}{\log^2(k)})$*.*

Remark 2. As before, the encoding scheme presented below is independent on the left and right. Therefore, our reduction holds for not just for $\text{Local}^{\ell_o'}$ but additionally any split-state function, independent on each side, trivially.

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

- $r := \log^4(k)$

Let prg be a pseudorandom generator for space bounded computations (see Definition 21), with inputs of length r and outputs of length $\log(\tau) \cdot \tau$.
Let $\mathrm{G}(\zeta)$ be defined as follows:

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

Let $\mathsf{E}' : \{0,1\}^n \to \{0,1\}^N$ and $\mathsf{D}' : \{0,1\}^N \to \{0,1\}^n$.
$\mathsf{E}'(x^{\mathsf{L}} := x^{\mathsf{L}}_1, \ldots, x^{\mathsf{L}}_{n_L}, x^{\mathsf{R}})$:

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

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

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

Figure 5.3: THE $(\mathrm{Local}^{\ell_o'}, \mathrm{LLocal}, \mathrm{negl}(n))$-NON-MALLEABLE REDUCTION $(\mathsf{E}', \mathsf{D}')$

- $\tau := 2(n - n_L)$, where n is the length of the output of LLocal and n_L is the length of the left output of LLocal.

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

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

By the definition of a non-malleable reduction (Definition 5), in order to complete the proof of Theorem 23, we must show that $(\mathsf{E}', \mathsf{D}')$ has the following properties:

1. For all $x \in \{0,1\}^n$, we have $\mathsf{D}'(\mathsf{E}'(x)) = x$ with probability 1.

2. For all $\mu \in \text{Local}^{\ell_o'}$,

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

where G_μ is the distribution defined above.

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

In the following, we prove Item (2), above, by noting that the statistical distance

$$\Delta(\mathsf{D}'(\mu(\mathsf{E}'(x))); G_\mu(x))$$

is upper bounded by the probability that either ρ does not contain at least n_L number of ones or C is not in LLocal.

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

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

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

of positions i in X^{L} are such that (1) X_i^{L} has "high input locality" and (2) $\rho_i = 1$ is negligible.

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

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

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

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

T can clearly be implemented by a read-once, Finite State Machine (FSM) with $2^{O(\log^2(\tau))}$ number of states. However, note that by Theorem 12, prg is a pseudorandom generator for space $\log^3(k)$ with parameter $2^{-\log^3(k)}$. Thus, existence of distinguisher T as above, leads to contradiction to the security of the Nisan PRG.[3]

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

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

[3]Note that by Theorem 16, t-wise independence would suffice in the analysis, where $\mathsf{prg}(s) = h_s(1), \ldots, h_s(N)$ for some t-wise independent hash family$\{h_s : [N] \to \{0,1\}\}_s$.

only substantial difference is in the analysis.

Theorem 24. $(\mathsf{E}', \mathsf{D}')$ *is a* $(\mathrm{Local}^{\ell_o} \Rightarrow SS, \mathrm{negl}(n))$*-non-malleable reduction given the following parameters for* Local^{ℓ_o}*:*

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

Putting together Theorem 24 with Theorems 1 and 4, we obtain the following.

Corollary 6. $(\mathsf{E} \circ \mathsf{E}_{\mathrm{SS}}, \mathsf{D}_{\mathrm{SS}} \circ \mathsf{D})$ *is a* $(\mathrm{Local}^{\ell_o}, k, \mathrm{negl}(k))$*-non-malleable code with rate* $\Theta(1/\ell_o)$*, where* $\ell_o = o(n/\log n)$.

Remark 3. Note that $n = \Theta(\ell_o k)$. Thus, for resilience against $\ell_o = n^{1-\varepsilon}$ our codes is of polynomial length $n = k^{1/\varepsilon}$.

We construct an encoding scheme (E, D) summarized in Figure 5.1 and parametrized below. We then show that the pair (E, D) is a $(\mathrm{Local}^{\ell_o}, \mathrm{SS}_k, \mathrm{negl}(k))$-non-malleable reduction.

We parameterize our construction for $\mathrm{Local}^{\ell_o} \Rightarrow \mathrm{SS}_k$ with the following:

- $t := \lceil n_R(1 - c_R^{\mathsf{err}}/4) \rceil$.
- $c^{\mathsf{dec}} := 1 - \frac{t}{n_R}$.
- $\delta := \frac{c^{\mathsf{sec}}}{9}$.
- $(\mathrm{E}_L, \mathrm{D}_L)$ parametrized by $(k, n_L, c_L^{\mathsf{err}}, c_L^{\mathsf{sec}}) := (k, c^{\mathsf{rate}}k, c^{\mathsf{err}}, c^{\mathsf{sec}})$ where $c^{\mathsf{err}}, c^{\mathsf{sec}}, c^{\mathsf{rate}}$ are taken from lemma 6.
- $n_{\mathsf{check}} := k$.
- $n_R := \lceil \frac{2\ell_o c^{\mathsf{rate}} k}{\delta c^{\mathsf{dec}} c^{\mathsf{sec}}} \rceil$
- $(\mathrm{E}_R, \mathrm{D}_R)$ parametrized by $(k, n_R, c_R^{\mathsf{err}}, c_R^{\mathsf{sec}}) := (k, n_R, c^{\mathsf{err}}, c^{\mathsf{sec}})$.
- $r := k$

- $\tau := \lceil \frac{30\ell_o c^{\mathsf{rate}} k}{(c^{\mathsf{sec}})^2 c^{\mathsf{dec}}} + \frac{1}{\delta} \rceil$
- $n := r + \tau + n_R = O(\ell_o k)$.

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

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

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

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

- Let $U = \{i \in [\tau] : \rho_i = 1\}$ denote the (relative) locations of $\vec{s}^{\mathsf{L}}$.
- Let $\mathcal{S}_{\mathsf{R}\to\mathsf{L}}$ denote the set of positions j such that input bit $\vec{s}^{\mathsf{R}}_j$ affects the output of f^{L}.
- Let $\mathcal{S}_{\mathsf{L}\to\mathsf{R}}$ denote the set of positions i such that input bit $\vec{s}^{\mathsf{L}}_i$ affects the output of f^{R}.

Let $(\mathrm{E}_L, \mathrm{D}_L, \mathrm{R}_L)$ be a binary reconstructable probabilistic encoding scheme with parameters $(k, n_L, c_L^{\mathsf{err}}, c_L^{\mathsf{sec}})$ and let $(\mathrm{E}_R, \mathrm{D}_R, \mathrm{R}_R)$ be a binary reconstructable probabilistic encoding scheme with parameters $(k, n_R, c_R^{\mathsf{err}}, c_R^{\mathsf{sec}})$.
Also, let n_{check} be a parameter.
Let prg be a pseudorandom generator for space bounded computations (see Definition 21), with inputs of length r and outputs of length $\log(\tau) \cdot \tau$.
Let $\mathrm{G}(\zeta)$ be defined as follows:

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

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

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

Figure 5.4: THE $(\mathrm{Local}^{\ell_o}, \mathrm{SS}, \mathrm{negl}(k))$-NON-MALLEABLE REDUCTION (E, D) (Continued in Figure 5.5)

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

1. (`Recover` $\widetilde{\rho}$) Let $\widetilde{y} := \mathsf{prg}(\widetilde{\zeta})$, where $\widetilde{y} = \widetilde{y}_1, \ldots, \widetilde{y}_\tau$. For $i \in [\tau]$, compute $\widetilde{\rho}_i := \phi(\widetilde{y}_i)$. Let $\widetilde{\mathsf{num}}$ denote the number of ones in $\widetilde{\rho} := \widetilde{\rho}_1, \ldots, \widetilde{\rho}_\tau$.

2. (`Recover` x) For $j \in [\widetilde{\mathsf{num}}]$, let pos_j denote the j-th position i such that $\widetilde{\rho}_i = 1$.

3. Let $\vec{\sigma}^{\mathsf{L}} \in \{0,1\}^{n_L}$ be defined in the following way: For $j \in [\min(\widetilde{\mathsf{num}}, n_L)]$, $\sigma_j^{\mathsf{L}} := \widetilde{X}^{\mathsf{L}}_{\mathsf{pos}_j}$.

4. (`plain decoding on left`) Compute $((w_1^{\mathsf{L}}, \ldots, w_{n_L}^{\mathsf{L}}), \mathsf{L}) \leftarrow \mathrm{D}_L(\sigma_1^{\mathsf{L}}, \ldots, \sigma_{n_L}^{\mathsf{L}})$. If the decoding fails, set $\mathsf{L} := \perp$.

5. (`decoding-check on right`) Let $t := \lceil n_R(1 - c_R^{\mathsf{err}}/4) \rceil$ Define $\vec{\sigma}'^{\mathsf{R}} := \sigma_1'^{\mathsf{R}}, \ldots, \sigma_{n_R}'^{\mathsf{R}}$ as follows: Set $\sigma_\ell'^{\mathsf{R}} := \sigma_\ell^{\mathsf{R}}$ for $\ell = 1, \ldots, t$. Set $\sigma_\ell'^{\mathsf{R}} := 0$ for $\ell = t+1, \ldots, n_R$. Compute $((w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}}), \mathsf{R}) \leftarrow \mathrm{D}_R(\sigma_1'^{\mathsf{R}}, \ldots, \sigma_t'^{\mathsf{R}})$. If the decoding fails or $(w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}})$ is not $c_R^{\mathsf{err}}/4$-close to $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{t_R}^{\mathsf{R}})$, set $\mathsf{R} := \perp$.

6. (`codeword-check on right`) Pick a random subset $R_{\mathsf{check}} \subset [n_R]$ of size $n_{\mathsf{check}} < c_R^{\mathsf{sec}} \cdot n_R$. For all $\ell \in R_{\mathsf{check}}$, check that $\sigma_\ell^{\mathsf{R}} = w_\ell^{\mathsf{R}}$. If the check fails, set $\mathsf{R} := \perp$.

7. (`output`) Output $x := (\mathsf{L}, \mathsf{R})$.

Figure 5.5: CONTINUATION OF THE $(\mathrm{Local}^{\ell_o}, \mathrm{SS}, \mathrm{negl}(k))$-NON-MALLEABLE REDUCTION (E, D) (Begins in Figure 5.4)

- For $J \subset [n_R]$, let $\mathcal{S}^{J}_{\mathsf{L}\to\mathsf{R}}$ denote the set of positions i such that input bit $\vec{s}^{\mathsf{L}}_i$ affects the output of f^{R}_j for some $j \in J$.

- For a set $R_{\mathsf{check}} \subseteq n_R$ of size n_{check}, let $\mathcal{S}_{\mathsf{check}}$ denote the set of positions i such that embedded input bit $\vec{s}^{\mathsf{L}}_i$ affects the output of f^{R}_ℓ for some $\ell \in R_{\mathsf{check}}$.

- For a set $R_{\mathsf{check}} \subseteq n_R$ of size n_{check}, let $\mathcal{S}^{+}_{\mathsf{check}}$ denote the set of positions i such that input bit X^{L}_i affects the output of f^{R}_ℓ for some $\ell \in R_{\mathsf{check}}$.

- For a set $R_{\mathsf{check}} \subseteq n_R$ of size n_{check}, let $\mathcal{S}_{\mathsf{check}}$ denote the set of positions i such that embedded input bit $\vec{s}^{\mathsf{L}}_i$ affects the output of f^{R}_ℓ for some $\ell \in R_{\mathsf{check}}$.

- Let $S^{+}_{\mathsf{in}}(\ell_i)$ denote the set of i such that X^{L}_i has input locality greater than ℓ_i.

- Let $S_{\mathsf{in}}(\ell_i)$ denote the set of i such that $\vec{s}^{\mathsf{L}}_i$ has input locality greater than ℓ_i.

- Let $\mathcal{S}^{+}_{\mathsf{L}\to\zeta}$ denote the set of positions i such that input bit X^{L}_i affects the output of f_ζ.

- Let $\mathcal{S}_{\mathsf{L}\to\zeta}$ denote the set of positions i such that input bit $\vec{s}^{\mathsf{L}}_i$ affects the output of f_ζ.

- Let $\mathcal{S}_{\mathsf{R}\to\zeta}$ denote the subset of $\mathcal{S}_{\mathsf{R}\to\mathsf{L}}$ that affects f_ζ.

We next define the following event Good_f.

Definition 29. The event Good_f occurs if for tampering function $f \in \mathrm{Local}^{\ell_o}$ all of the following hold:

1. ρ contains at least n_L ones.

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

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

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

Claim 7. Suppose ρ is chosen truly at random (ones occuring with bias $p = 3n_L/2\tau$). Then for every $f \in \text{Local}^{\ell_o}$, $\Pr[\mathsf{Good}_f] \geq 1 - \text{negl}(n)$.

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

1. Recall that $|\rho| = \tau, n_L = c^{\mathsf{rate}}k, n = O(\ell_o k)$ and $p = \frac{3n_L}{2\tau}$.

   Let the number of ones in $\rho = \rho_1$. Note that ρ_1 is a Binomial random variable with parameters (τ, p).

   Therefore, let $\mu = \mathbf{E}[\rho_1] = \tau p = \tau\frac{3n_L}{2\tau} = \frac{3n_L}{2}$ Using Chernoff's bound we can write, $\Pr[\rho_1 \leq 2/3\mu] \leq e^{\frac{-1/3^2}{2}\mu}$. Therefore,

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

   which is negligible.

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

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

$\mathcal{S}_{\mathsf{L}\to\zeta}| \geq 9n_L \cdot \delta = c^{\mathsf{sec}} \cdot n_L] \leq e^{-\frac{c^{\mathsf{sec}} \cdot n_L}{6}}$, which is negligible.

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

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

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

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

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

□

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

- Choose $\zeta \leftarrow \{0,1\}^r$ uniformly at random. Compute $\rho := G(\zeta)$.
- If Good_f does not hold, then set g to the constant function 0.
- Otherwise, choose vectors $I^{\mathsf{R}} \in \{0,1\}^{n_R}, I^{\mathsf{L}} \in \{0,1\}^{n_L}$ uniformly at random. $I^{\text{mask}} \in \{0,1,*\}^\tau$ such that if $i \leq \mathsf{pos}$ (the location of the n_L-th 1 in ρ) and $\rho_i = 1$, then $I_i^{\text{mask}} = *$ and is uniformly independently drawn from $\{0,1\}$ otherwise.

Let I^X be I^{mask} where the i-th $*$ is replaced by the ith bit of I^{L}. (id $\restriction_{I^{\text{mask}}}(I^{\mathsf{L}})$.)

- Compute $\widetilde{\zeta}$ according to f_ζ, given inputs ζ and $I^{\mathsf{L}}_{\mathcal{S}_{\mathsf{L}\to\zeta}}, I^{\mathsf{R}}_{\mathcal{S}_{\mathsf{R}\to\zeta}}$.
- Compute $\widetilde{\rho} := \mathrm{G}(\widetilde{\zeta})$, where $\widetilde{\rho} = \widetilde{\rho}_1, \ldots, \widetilde{\rho}_\tau$. Let $\widetilde{U} = \{i \in [\tau] : \widetilde{\rho}_i = 1\}$.
- Choose a random subset $R_{\mathsf{check}} \subseteq [n_R]$ of size n_{check}.
- Let J^* be the the lexicographically first subset of $[n_R]$ such that $|J^*| = t$ and $|\mathcal{S}^{J^*}_{\mathsf{L}\to\mathsf{R}} \setminus S_{\text{in}}(\ell_o/\delta)| = 0$. Note that such J^* must exist since Good_f occurs.
- The split-state tampering function $g := (g_L, g_R) \in \mathrm{SS}_k$ has $I^{\mathsf{L}}, I^{\mathsf{R}}, I^{\text{mask}}, \zeta$ hardcoded into it (as well as $\widetilde{U}, \widetilde{\zeta}, \widetilde{\rho}$) and is specified as follows:

  $g_L(\mathsf{L})$:

  1. (`apply tampering and plain decode on left`) Let $\vec{s}^{\mathsf{L}} := \mathrm{R}(\mathcal{S}_{\mathsf{check}} \cup \mathcal{S}_{\mathsf{L}\to\zeta} \cup S_{\text{in}}(\ell_o/\delta), I^{\mathsf{L}}, \mathsf{L})$. Let $(\sigma^{\mathsf{L}}_1, \ldots, \sigma^{\mathsf{L}}_{n_L}) := (f_X \restriction_{\zeta, I^{\text{mask}}, I^{\mathsf{R}}} (\vec{s}^{\mathsf{L}}))_{\widetilde{U}}$. Recall that the above notation denotes (1) applying the function f_X to the input $(\zeta, X^{\mathsf{L}}, I^{\mathsf{R}})$, where X^{L} is I^{mask} where the i-th $*$ is replaced by the ith bit of $\vec{s}^{\mathsf{L}}$. (2) outputting the positions of f_X indexed by $\widetilde{U}$. Compute $((w^{\mathsf{L}}_1, \ldots, w^{\mathsf{L}}_{n_L}), \widetilde{\mathsf{L}}) \leftarrow \mathrm{D}_L(\sigma^{\mathsf{L}}_1, \ldots, \sigma^{\mathsf{L}}_{n_L})$. If the decoding fails, set $\widetilde{\mathsf{L}} := \perp$.
  2. (`output`) Output $\widetilde{\mathsf{L}}$.

  $g_R(\mathsf{R})$:

  1. (`apply tampering and decoding-check on right`) Let $\vec{s}^{\mathsf{R}} = (s^{\mathsf{R}}_1, \ldots, s^{\mathsf{R}}_{n_R}) := \mathrm{R}(\mathcal{S}_{\mathsf{R}\to\mathsf{L}}, I^{\mathsf{R}}, \mathsf{R})$. Let $(\sigma^{\mathsf{R}}_1, \ldots, \sigma^{\mathsf{R}}_{n_R}) := f^{\mathsf{R}} \restriction_{\zeta, I^X} (\vec{s}^{\mathsf{R}})$. Define $\vec{\sigma}'^{\mathsf{R}} := \sigma'^{\mathsf{R}}_1, \ldots, \sigma'^{\mathsf{R}}_{n_R}$ as follows: Set $\sigma'^{\mathsf{R}}_\ell := \sigma^{\mathsf{R}}_\ell$ for $\ell \in [J^*]$. Set $\sigma'^{\mathsf{R}}_\ell := 0$ for $\ell \notin [J^*]$. Compute $((w^{\mathsf{R}}_1, \ldots, w^{\mathsf{R}}_{n_R}), \widetilde{\mathsf{R}}) \leftarrow \mathrm{D}_R(\sigma'^{\mathsf{R}}_1, \ldots, \sigma'^{\mathsf{R}}_t)$. If the decoding fails or $(w^{\mathsf{R}}_1, \ldots, w^{\mathsf{R}}_{n_R})$ is not $c^{\mathsf{err}}_R/4$-close to $(\sigma^{\mathsf{R}}_1, \ldots, \sigma^{\mathsf{R}}_{n_R})$, then set $\widetilde{\mathsf{R}} := \perp$.
  2. (`codeword-check on right`) For all $\ell \in R_{\mathsf{check}}$, check that $\sigma^{\mathsf{R}}_\ell = w^{\mathsf{R}}_\ell$. If the check fails, set $\widetilde{\mathsf{R}} := \perp$.

3. (`output`) Output $\widetilde{\mathsf{R}}$.

- Output $g = (g_L, g_R)$.

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

Remark 4. Note that $g = (g_L, g_R)$ drawn from the distribution G_f is a split-state function with probability 1. This is true since if the event Good_f does not hold, then g is set to the constant function 0 (which is in split-state). If the event Good_f does hold, then as can be seen by inspection above, g_L takes only L as input, g_R takes only R as input.

By the definition of a non-malleable reduction (Definition 5), in order to complete the proof of Theorem 22, we must show that (E, D) have the following properties:

1. For all $x \in \{0,1\}^{2k}$, we have $\mathsf{D}(\mathsf{E}(x)) = x$ with probability 1.

2. For all $f \in \text{Local}^{\ell_o}$,

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

where G_f is the distribution defined above.

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

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

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

Firstly, we will argue that with high probability, the pseudorandomness of the PRG[4] is sufficient to obtain that the event Good_f holds even when ρ is chosen via the PRG (instead of being truly

[4]As before, we note that a t-wise indpendent generator would also suffice here.

random). This will give us bounds on the "bad" bits in the output of the encoding of the left input, L.

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

So for our final hybrid, we will use a technique from [67] to show that we can decode from a "clean" portion of $\vec{\sigma}^{\mathsf{R}}$, J^*. This is possible because, by Claim 8 (and previous hybrids), the only non-uniformly-random bits in $\vec{s}^{\mathsf{L}}$ have bounded input locality. Thus, these bits on the left can only effect a constant fraction of bits on the right.

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

Claim 8. In any given tampering experiment, with tampering function $f \in\in \mathrm{Local}^{\ell_o}$, the event Good_f holds with all but negligible probability, even when ρ generated according to Figure 5.4.

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

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

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

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

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

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

T can clearly be implemented by a read-once, Finite State Machine (FSM) with $2^{O(\log^2(\tau))}$ number of states. However, note that by Theorem 12, prg is a pseudorandom generator for space $\log^3(k)$ with parameter 2^{-k}. Thus, existence of distinguisher T as above, leads to contradiction to the security of the Nisan PRG. □

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

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

1. Given f, find $\mathcal{S}_{\mathsf{R}\to\zeta}$, $\mathcal{S}_{\mathsf{R}\to\mathsf{L}}$, $\mathcal{S}_{\mathsf{L}\to\zeta}$, $\mathcal{S}_{\mathsf{check}}$, and $S_{\text{in}}(\ell_o/\delta)$.
2. Compute $\vec{s}^{\mathsf{L}} = (s_1^{\mathsf{L}}, \ldots, s_{n_L}^{\mathsf{L}}) \leftarrow \mathrm{R}(\mathcal{S}_{\mathsf{check}} \cup \mathcal{S}_{\mathsf{L}\to\zeta} \cup S_{\text{in}}(\ell_o/\delta), \mathrm{E}_L(\mathsf{L}), \mathsf{L})$ and $\vec{s}^{\mathsf{R}} = (s_1^{\mathsf{R}}, \ldots, s_{n_R}^{\mathsf{R}}) \leftarrow \mathrm{R}(\mathcal{S}_{\mathsf{R}\to\mathsf{L}}, \mathrm{E}_R(\mathsf{R}), \mathsf{R})$.
3. Choose $\zeta \leftarrow \{0,1\}^r$ uniformly at random. Compute $\rho := \mathrm{G}(\zeta)$.
4. Otherwise, for $j \in [n_L]$, let pos_j denote the j-th position i such that $\rho_i = 1$.

5. Let $X^{\mathsf{L}} \in \{0,1\}^{\tau}$ be defined in the following way: For $j \in [n_L]$, $X^{\mathsf{L}}_{\mathsf{pos}_j} := s^{\mathsf{L}}_j$. In all other locations, X^{L}_i is set uniformly at random.

6. Output the encoding $(\zeta, X^{\mathsf{L}}, \vec{s}^{\mathsf{R}})$.

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

Claim 9.

$$H_0 \equiv H_1.$$

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

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

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

2. Choose $I^{\mathsf{L}} \in \{0,1\}^{n_L}, I^{\mathsf{R}} \in \{0,1\}^{n_R}$ uniformly at random.

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

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

5. For $j \in [n_L]$, let pos_j denote the j-th position i such that $\rho_i = 1$.

6. Let $X^{\mathsf{L}} \in \{0,1\}^{\tau}$ be defined in the following way: For $j \in [n_L]$, $X^{\mathsf{L}}_{\mathsf{pos}_j} := s^{\mathsf{L}}_j$. In all other locations, X^{L}_i is set uniformly at random.

7. Output the encoding $(\zeta, X^{\mathsf{L}}, \vec{s}^{\mathsf{R}})$.

Claim 10.

$$H_1 \equiv H_2.$$

By Claim 8, the sets reconstructed from are below the security thresholds of the respective RPE schemes. Thus by definition, the distribution of $\mathsf{E}^{(2)}(x)$ is identical to that of $\mathsf{E}(x)$.

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

1. (`Recover` $\widetilde{\rho}$) Let $\widetilde{y} := \mathsf{prg}(\widetilde{\zeta})$, where $\widetilde{y} = \widetilde{y}_1, \ldots, \widetilde{y}_\tau$. For $i \in [\tau]$, compute $\widetilde{\rho}_i := \phi(\widetilde{y}_i)$. Let $\widetilde{\mathsf{num}}$ denote the number of ones in $\widetilde{\rho} := \widetilde{\rho}_1, \ldots, \widetilde{\rho}_\tau$.

2. (`Recover` x) For $j \in [\widetilde{\mathsf{num}}]$, let pos_j denote the j-th position i such that $\widetilde{\rho}_i = 1$.

3. Let $\vec{\sigma}^{\mathsf{L}} \in \{0,1\}^{n_L}$ be defined in the following way: For $j \in [\min(\widetilde{\mathsf{num}}, n_L)]$, $\sigma_j^{\mathsf{L}} := \widetilde{X}^{\mathsf{L}}_{\mathsf{pos}_j}$.

4. (`plain decoding on left`) Compute $((w_1^{\mathsf{L}}, \ldots, w_{n_L}^{\mathsf{L}}), \mathsf{L}) \leftarrow \mathrm{D}_L(\sigma_1^{\mathsf{L}}, \ldots, \sigma_{n_L}^{\mathsf{L}})$. If the decoding fails, set $\mathsf{L} := \perp$.

5. (`decoding-check on right`) Let $t := \lceil n_R(1 - c_R^{\mathsf{err}}/4) \rceil$ Define $\vec{\sigma}'^{\mathsf{R}} := \sigma_1'^{\mathsf{R}}, \ldots, \sigma_{n_R}'^{\mathsf{R}}$ as follows: Set $\sigma_\ell'^{\mathsf{R}} := \sigma_\ell^{\mathsf{R}}$ for $\ell = 1, \ldots, t$. Set $\sigma_\ell'^{\mathsf{R}} := 0$ for $\ell = t+1, \ldots, n_R$. Compute $((w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}}), \mathsf{R}) \leftarrow \mathrm{D}_R(\sigma_1'^{\mathsf{R}}, \ldots, \sigma_t'^{\mathsf{R}})$. If the decoding fails or $(w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}})$ is not $c_R^{\mathsf{err}}/4$-close to $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{t_R}^{\mathsf{R}})$, set $\mathsf{R} := \perp$.

6. (`codeword-check on right`) Pick a random subset $R_{\mathsf{check}} \subset [n_R]$ of size $n_{\mathsf{check}} < c_R^{\mathsf{sec}} \cdot n_R$. For all $\ell \in R_{\mathsf{check}}$, check that $\sigma_\ell^{\mathsf{R}} = w_\ell^{\mathsf{R}}$. If the check fails, set $\mathsf{R} := \perp$.

7. (`output`) Output $x := (\mathsf{L}, \mathsf{R})$.

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

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

10. (`output`) Output $x := (\mathsf{L}, \mathsf{R})$.

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

Claim 11.

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

We want to show that for every $\vec{\sigma} = (\vec{\sigma}^{\mathsf{L}}, \vec{\sigma}^{\mathsf{R}}) \leftarrow f(\mathsf{E}(x))$, $\mathsf{D}(\vec{\sigma}) = \mathsf{D}'(\vec{\sigma})$ with high probability, over the coins of D, D'. The proof is identical to the proof of Claim 4.

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

Claim 12.

$$H_3 \equiv H_4.$$

This claim follows by inspection.

5.5 Limits to Local-Tamper Resilience

A function $f : \{0,1\}^n \to \{0,1\}^n$ is $(n - \log(n))$-output-local if each output bit depends on at most $n - \log(n)$ input bits. The particular class $\mathcal{F}'$ that we use in our lower bound proof is a subclass of all $(n - \log(n))$-local tampering functions $\mathcal{F}$. Each $f \in \mathcal{F}'$ has the following structure: First, $f_1, \ldots, f_{n-\log(n)}$ (the functions that output the first $n - \log(n)$ bits) are all the same, except that two different bits from $\{0,1\}$ are hardcoded in each. Second, $f_{n-\log(n)+1}, \ldots, f_n$ are also the same, except that a different value from $\{0,1\}$ is hardcoded in each. Finally, the set of input bits upon which $f_1, \ldots, f_{n-\log(n)}$ depend and the set of input bits upon which $f_{n-\log(n)+1}, \ldots, f_n$ depend are fixed. Taken together, this means that the total number of functions f in $\mathcal{F}$ is at most $4^n \cdot 2^{2^{n-\log(n)}}$, so $\log\log|\mathcal{F}'| = n - \log(n)$. On the other hand, Dziembowski et al. [42] showed existence of a $1/n$-non-malleable code for any class $\mathcal{F}$ such that $\log\log|\mathcal{F}| \leq n - 2\log(n)$. Thus, our lower bound result is nearly tight matching the existential upper bound. In our theorem, we prove a more general statement:

Theorem 25. *Let* (E, D) *be a coding scheme with* $\mathsf{E} : \{0,1\} \to \{0,1\}^n$ *and* $\mathsf{D} : \{0,1\}^n \to \{0,1\}$. *Let* $\mathcal{F}$ *be the class of* $(n - \log(1/\epsilon) + 2)$*-output- local functions, where* $1/8 \geq \epsilon \geq 1/2^n$. *Then* (E, D) *is* ϵ*-malleable with respect to* $\mathcal{F}$.

Note that the parameters discussed above can be obtained by setting $\epsilon = \frac{1}{4n}$.

Additionally, note that non-malleable codes whose decode function D may output values in $\{0, 1, \perp\}$ *imply* non-malleable codes whose decode function D may only output values in $\{0, 1\}$. Thus, ruling out the latter *implies* ruling out the former and only makes our result stronger.

Proof. Fix an arbitrary (E, D) with $\mathsf{E} : \{0,1\} \to \{0,1\}^n$ and $\mathsf{D} : \{0,1\}^n \to \{0,1\}$. Our analysis considers two cases and shows that for each case, there exists $f \in \mathcal{F}$ such that

$$\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1 - b] \geq \frac{1}{2} + \epsilon.$$

This is sufficient to prove Theorem 25.

We begin with some notation and the proceed to the case analysis. For codeword $c = c_1, \ldots, c_n$, let c^{top} (resp. c^{bot}) denote the first $n - \log(1/\epsilon) + 2$ bits (resp. last $\log(1/\epsilon) - 2$ bits) of c. I.e. $c^{\mathsf{top}} := c_1, \ldots, c_{n-\log(1/\epsilon)+2}$ ($c^{\mathsf{bot}} := c_{n-\log(1/\epsilon)+3}, \ldots, c_n$). For $t \in \mathbb{N}$, let S_t denote the set of all t-bit strings and let U_t denote the uniform distribution over t bits. Assume $n \geq 2$.

Case 1:

$$\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(c^{\mathsf{top}} || r) = b \mid c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}] \geq 1/2 + \epsilon.$$

Let $c^{*,0} = c_1^{*,0}, \ldots, c_n^{*,0}$ (resp. $c^{*,1} = c_1^{*,1}, \ldots, c_n^{*,1}$) be the lexicographically first string that decodes to 0 (resp. 1) under D (i.e. $\mathsf{D}(c^{*,0}) = 0$ and $\mathsf{D}(c^{*,1}) = 1$.

In this case we consider the following distribution over tampering circuits $f = f_1, \ldots, f_n$, where f_i outputs the i-th bit of f:

Sample $r \leftarrow U_{\log(1/\epsilon)-2}$, construct circuits f_i for each $i \in [n]$, which take input c^{top} and output c'_i. Each f_i does the following:

- Compute $d := \mathsf{D}(c^{\mathsf{top}}||r)$.
- Output $c'_i = c_i^{*,1-d}$.

We now analyze $\Pr_{b\leftarrow\{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1-b]$.

$$\begin{aligned}\Pr_{b\leftarrow\{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1-b] &= \Pr_{b\leftarrow\{0,1\}}[f(\mathsf{E}(b)) \text{ outputs } c^{*,1-b}] \\ &= \Pr_{b\leftarrow\{0,1\}}[\mathsf{D}(c^{\mathsf{top}}||r) = b \mid c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}] \\ &\geq 1/2+\epsilon,\end{aligned}$$

where the two equalities follow from the definition of the tampering function f, and the inequality follows since we are in Case 1. This implies the ϵ-malleability of (E,D).

Case 2:

$$\Pr_{b\leftarrow\{0,1\}}[\mathsf{D}(c^{\mathsf{top}}||r) = 1-b \mid c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}] \geq 1/2-\epsilon.$$

In this case we consider the following distribution over tampering circuits $f = f_1,\ldots,f_n$, where f_i outputs the i-th bit of f:

The first $n-\log(1/\epsilon)+2$ circuits $(f_1,\ldots,f_{n-\log(1/\epsilon)+2})$ simply compute the identity function: I.e. f_i for $i \in [n-\log(1/\epsilon)+2]$ takes c_i as input and produces c_i as output.

We next describe the distribution over circuits f_i for $i \in \{n-\log(1/\epsilon)+3,\ldots,n\}$. Sample $r' \leftarrow [1/(4\epsilon)-1]$. Construct circuits f_i for each $i \in \{n-\log(1/\epsilon)+3,\ldots,n\}$ that take input c^{bot} and produce output c'_i. Each f_i does the following:

- Let $r := r_{n-\log(1/\epsilon)+3},\ldots,r_n$ be the r'-th lexicographic string in the set $S_{\log(1/\epsilon)-2} \setminus \{c^{\mathsf{bot}}\}$[5].
- Output $c'_i = r_i$.

We now analyze $\Pr_{b\leftarrow\{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1-b]$.

[5]Recall that, $t \in \mathbb{N}$, let S_t denote the set of all t-bit strings and let U_t denote the uniform distribution over t bits.

Since we are in Case 2 we have that:

$$
\begin{aligned}
1/2 - \epsilon &\leq \Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(c^{\mathsf{top}}||r) = 1 - b \mid c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2}] \\
&= \Pr_{b \leftarrow \{0,1\}}\left[\begin{array}{c} c^{\mathsf{bot}} = r \mid \\ c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2} \end{array}\right] \cdot \Pr_{b \leftarrow \{0,1\}}\left[\begin{array}{c} \mathsf{D}(c^{\mathsf{top}}||r) = 1 - b \mid \\ c^{\mathsf{bot}} = r \wedge c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2} \end{array}\right] \\
&\quad + \Pr_{b \leftarrow \{0,1\}}\left[\begin{array}{c} c^{\mathsf{bot}} \neq r \mid \\ c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log n} \end{array}\right] \cdot \Pr_{b \leftarrow \{0,1\}}\left[\begin{array}{c} \mathsf{D}(c^{\mathsf{top}}||r) = 1 - b \mid \\ c^{\mathsf{bot}} \neq r \wedge c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2} \end{array}\right] \\
&= \Pr_{b \leftarrow \{0,1\}}\left[\begin{array}{c} c^{\mathsf{bot}} = r \mid \\ c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2} \end{array}\right] \cdot 0 \\
&\quad + \Pr_{b \leftarrow \{0,1\}}\left[\begin{array}{c} c^{\mathsf{bot}} \neq r \mid \\ c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2} \end{array}\right] \cdot \Pr_{b \leftarrow \{0,1\}}\left[\begin{array}{c} \mathsf{D}(c^{\mathsf{top}}||r) = 1 - b \mid \\ c^{\mathsf{bot}} \neq r \wedge c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2} \end{array}\right] \\
&= \Pr_{b \leftarrow \{0,1\}}\left[\begin{array}{c} c^{\mathsf{bot}} \neq r \mid \\ c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2} \end{array}\right] \cdot \Pr_{b \leftarrow \{0,1\}}\left[\begin{array}{c} \mathsf{D}(c^{\mathsf{top}}||r) = 1 - b \mid \\ c^{\mathsf{bot}} \neq r \wedge c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2} \end{array}\right] \\
&= (1 - 4\epsilon) \cdot \Pr_{b \leftarrow \{0,1\}}\left[\begin{array}{c} \mathsf{D}(c^{\mathsf{top}}||r) = 1 - b \mid \\ c^{\mathsf{bot}} \neq r \wedge c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2} \end{array}\right].
\end{aligned}
$$

Note that

$$
\Pr_{b \leftarrow \{0,1\}}\left[\begin{array}{c} \mathsf{D}(c^{\mathsf{top}}||r) = 1 - b \mid \\ c^{\mathsf{bot}} \neq r \wedge c \leftarrow \mathsf{E}(b), r \leftarrow U_{\log(1/\epsilon)-2} \end{array}\right] = \Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1 - b].
$$

Thus, we have that

$$
1/2 - \epsilon \leq (1 - 4\epsilon) \Pr_{b \leftarrow \{0,1\}}[\mathsf{D}(f(\mathsf{E}(b))) = 1 - b].
$$

Since

$$\begin{aligned}(1/2+\epsilon)\cdot(1-4\epsilon) &= 1/2+\epsilon-2\epsilon-4\epsilon^2\\ &\leq 1/2-\epsilon,\end{aligned}$$

we have that

$$\begin{aligned}\Pr_{b\leftarrow\{0,1\}}[\mathsf{D}(f(\mathsf{E}(b)))=1-b] &\geq \frac{1/2-\epsilon}{1-4\epsilon}\\ &\geq 1/2+\epsilon.\end{aligned}$$

This implies the ϵ-malleability of (E,D).

□

Chapter 6: Decision Tree Tampering

In this chapter we construct non-malleable codes that are resilient to tampering by low-depth decision trees.

Decision trees of depth d capture tampering where each output bit is set arbitrarily after adaptively reading d locations of the input, where the choice of which input location to read next at any point in time can depend on the values of all the previous locations read. In contrast, local tampering corresponds to the locations being statically chosen a priori.[1]

Definition 30 (Decision Trees)**.** A decision tree with n input bits is a binary tree whose internal nodes have labels from $x_1, \ldots, x_n$ and whose leaves have labels from $\{0, 1\}$. If a node has label x_i then the test performed at that node is to examine the i-th bit of the input. If the result is 0, one descends into the left subtree , whereas if the result is 1, one descends into the right subtree. The label of the leaf so reached is the output value on that particular input. The *depth* of a decision tree is the number of edges in a longest path from the root to a leaf. Let $\mathrm{DT}(t)$ denote decision trees with depth at most t.

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

Theorem 26. *For any $t = O(n^{1/4}/\log^{3/2} n)$, there is an explicit and efficient non-malleable code that is unconditionally secure against depth-t decision trees with codeword length $n = O(kt^2 \log^4 n/\log\log n)$ and error $\exp(-\Omega(n/t^4 \log^5 n))$ for a k-bit message.*

[1] Note that any decision tree of depth d can also be represented by a 2^d-local function or as a DNF with 2^d clauses of width d. Thus, the results of Chapter 5 yield non-malleable codes against $(1-\epsilon)\log n$-depth decision trees. The present chapter improves upon this exponentially.

6.1 Technical overview

We prove our theorem by constructing a non-malleable reduction (Def. 5) from decision-tree tampering to leaky split-state tampering. Theorem 26 is thus a corollary of this reduction, Lemma 15, and Theorem 21 from Chapter 4.

We will outline our reduction for decision tree tampering. The key idea here, as in Chapter 5, is to exploit size differences. Our encoder and decoder will work independently on the left and right pieces of the message, so we will in turn think of having left and right encoders, decoders, codewords, and tampering functions (corresponding to the respective outputs).

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

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

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

concentrations bounds guarantee that this is more or less what actually happens. Finally we simply union bound over all such subsets to guarantee that collectively the right tampering function makes few queries to the left with overwhelming probability.

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

6.2 Decision Trees to Leaky Split-State Model

In this section, we give a non-malleable reduction from decision tree tampering to leaky split-state tampering.

Lemma 15. *For any constant $\alpha \in (0,1)$ and $t = O(n^{1/4}/\log^{3/2} n)$, there is a $(\mathrm{DT}(t) \implies \alpha - \mathrm{SS}_k, \epsilon)$-non-malleable reduction with rate $\Omega(1/t^2 \log^3 n)$ where $\epsilon \leq \exp(-\Omega(n/t^4 \log^5 n))$.*

A single decision tree from $\{0,1\}^n$ to $\{0,1\}$ of depth t can be viewed as an adversary (or computation) that first adaptively queries at most t input coordinates, and then outputs a bit. Given a tampering function f from $\{0,1\}^n$ to $\{0,1\}^n$ such that each output bit is computed by a decision tree of depth t (i.e., each output is the result of adaptively querying at most t coordinates from the input, here a codeword), we will non-malleably reduce f to a special subclass of leaky split-state functions.

Recall that an α-leaky split-state function $g = (g_L, g_R)$ from $\{0,1\}^{2k}$ to $\{0,1\}^{2k}$ can be computed by a communication protocol with parameter α between Alice and Bob where Alice has access to $x_L := x_{1:k}$ and Bob has access to $x_R := x_{k+1:2k}$. Alice and Bob send information back and forth depending on their own inputs and the current transcript of the communication so far. And finally, Alice outputs $g_L(x_L, x_R)$ and Bob outputs $g_R(x_L, x_R)$. We consider a special subclass of α-leaky split-state function where Alice and Bob simply make a bounded number of adaptive queries to each other's input. In particular, when Alice (resp. Bob) makes a query to x_R (resp. x_L), Alice (resp. Bob) sends location $i \in [k]$ and ask Bob (resp. Alice) to send back the ith coordinate of x_R (resp. x_L).

Note that the communication cost for each query (and answer) is $\lceil \log k \rceil + 1$ bits. (For the remainder of this section, we will assume that both k and n are powers of 2.) So if both Alice and Bob make at most $\alpha k/(2(\log k + 1))$ queries to each other's input, the total communication is at most αk and g is an α-leaky split-state function. We will show that our reduction reduces decision tree tampering functions (of appropriate depth) to this subclass of leaky split-state functions.

Given $f = (f_L, f_R) \in \mathrm{DT}(t)$, we sample an α-split state g from the distribution G_f as follows: sample and hardwire the randomness required for $\mathrm{Enc}_L, \mathrm{Enc}_R$. Let ζ_L and ζ_R be the respective seeds used for G in Enc_L and Enc_R. Then, $g = (g_L, g_R)$ is defined as follows: on input $x = (x_L, x_R)$

- $g_L(x)$ simulates $\mathrm{Dec}_L(f_L(\mathrm{Enc}_L(x_L), \mathrm{Enc}_R(x_R)))$ with at most $\alpha k/2(\log k + 1)$ queries to x_R (*)
  1. Decode the tampered seed $\widetilde{\zeta_L}$ by computing the first $4m_L$ outputs of $f_L(\mathrm{Enc}_L(x_L), \mathrm{Enc}_R(x_R))$ then applying Dec_{RSS}.
     (To evaluate the necessary portion f_L the simulator evaluates the corresponding decision trees. When a bit is queried in $\mathrm{Enc}_R(x_R)$ we have three cases (recall that these encodings consist of two parts, $\mathrm{Enc}_{RSS}(\zeta_R)$ and c_R): (a) if the index corresponds to a location in $\mathrm{Enc}_{RSS}(\zeta_R)$, it is hardwired already in g_L; (b) if the index corresponds to c_R and a location specified by ζ_R, query/request the relevant bit from x_R; (c) if the index corresponds to c_R and a location *not* specified by ζ_R, simply use 0.)
  2. Compute the output bits of $f_L(\mathrm{Enc}_L(x_L), \mathrm{Enc}_R(x_R))$ indexed by set $G(\widetilde{\zeta_L})$.
     (The decision trees corresponding to indices specified by $\widetilde{\zeta_L}$ are evaluated identically to the preceeding step.)

  (*) whenever $g_L(x)$ makes more than $\alpha k/(2(\log k + 1))$ queries to x_R, abort and output 0^k.
- $g_R(x)$ simulates $\mathrm{Dec}_R(f_R(\mathrm{Enc}_L(x_L), \mathrm{Enc}_R(x_R)))$ with at most $\alpha k/2(\log k + 1)$ queries to x_L (*)
  1. Compute $\widetilde{y_R} = f_R(\mathrm{Enc}_L(x_L), \mathrm{Enc}_R(x_R))$.
     (The corresponding decision trees in f_R are evaluated in a symmetric manner to those needed for g_L.)
  2. Compute $\mathrm{Dec}_R(\widetilde{y_R})$.

  (*) whenever $g_R(x)$ makes more than $\alpha k/(2(\log k + 1))$ queries to x_L, abort and output 0^k.

Figure 6.1: Simulator for (Enc, Dec)

Our reduction relies on a ramp secret sharing scheme over binary alphabet. For parameter m, let $(\mathrm{Enc}_{RSS}, \mathrm{Dec}_{RSS})$ be an efficient coding scheme such that Enc_{RSS} maps an m-bit to a (random) $4m$-

bit string so that for any $\zeta \in \{0,1\}^m$ and subset $S \neq \emptyset$ of size at most m, $\mathrm{Enc}_{RSS}(\zeta)_S$ distributes uniformly and randomly over $\{0,1\}^{|S|}$. As observed by Ball et al. [90] (see Lemma 4), such a coding scheme can be constructed efficiently from any linear error correcting code from m bits to $4m$ bits with minimal distance $m+1$. We choose constant 4 to simplify the presentation.

Based on $(\mathrm{Enc}_{RSS}, \mathrm{Dec}_{RSS})$, we define a coding scheme that hides a message in random locations. Let $G_{k,n}$ be the function that given a short "seed," ζ, of k distinct indices in $[n]$ expands it to the corresponding n-bit string with hamming weight k (where the ones are in locations indexed by ζ). Let $D_{k,n}$ be a distribution such that $G_{k,n}(D_{k,n})$ is uniform over n-bit strings with hamming weight k. Note that $G_{k,n}$ can be computed efficiently and $D_{k,n}$ can be sampled efficiently by simply sampling k locations from $[n]$ without replacement. For k, n and $m \geq k \log n$, we define an encoding $\mathrm{Enc}^*_{n,k,m} : \{0,1\}^k \to \{0,1\}^{4m} \times \{0,1\}^n$, as follows: on a k-bit string x, sample a random seed $\zeta \leftarrow D_{k,n}$ for $G_{k,n}$, and output $(\mathrm{Enc}_{RSS}(\zeta), c)$ so that c is 0 everywhere except $c_{G(\zeta)} = x$. $\mathrm{Dec}^*_{n,k,m}$ is defined in the straightforward way (it first decodes ζ using Dec_{RSS}, then outputs $c_{G(\zeta)}$).

The high level idea of our reduction is to use two copies of Enc^* to hide inputs x_L and x_R independently inside two long messages (with different length). Let n_L, m_L, n_R, m_R be parameters to be determined later. We define Enc as $\mathrm{Enc}(x_L, x_R) = (\mathrm{Enc}_L(x_L), \mathrm{Enc}_R(x_R))$ where $\mathrm{Enc}_L = \mathrm{Enc}^*_{n_L,k,m_L}$ and $\mathrm{Enc}_R = \mathrm{Enc}^*_{n_R,k,m_R}$. And we define Dec as $\mathrm{Dec}(y_L, y_R) = (\mathrm{Dec}_L(y_L), \mathrm{Dec}_R(y_R))$ where $\mathrm{Dec}_L = \mathrm{Dec}^*_{n_L,k,m_L}$ and $\mathrm{Dec}_R = \mathrm{Dec}^*_{n_R,k,m_R}$.

Now we show (Enc, Dec) non-malleably reduces $\mathrm{DT}(t)$ to α-SS_k and prove Lemma 15. First observe that Dec$\circ$Enc is the identity function due to the correctness of $(\mathrm{Enc}_L, \mathrm{Dec}_L)$ and $(\mathrm{Enc}_R, \mathrm{Dec}_R)$. It remains to show that for any $f : \{0,1\}^n \to \{0,1\}^n \in \mathrm{DT}(t)$, $\mathrm{Dec} \circ f \circ \mathrm{Enc}$ becomes α-split state functions. In fact, in Figure 6.1, we reduce decision trees to a simpler subclass of α-SS_k where Alice and Bob, in parallel, make at most $\alpha k/(2(\log k + 1))$-bounded number of adaptive queries to locations of the other party's inputs, then output tampered values.

By the special condition $(*)$ in g_L, g_R, $g = (g_L, g_R)$ is indeed α-split state function because Alice and Bob communicates at most $\log k + 1$ bits per query (including the answer). Moreover, for any x, $G_f(x)$ distributes identically to $\mathrm{Dec} \circ f \circ \mathrm{Enc}(x)$ conditioning on that $(*)$ doesn't happen.

Therefore ϵ, the difference between the simulation and the real experiment, is at most the probability that $(*)$ happens. To bound the event that $(*)$ happens, we begin by proving the following more general proposition.

Proposition 6. *For integers $n, k, m \geq k \log n$. Let A be an arbitrary algorithm that makes at most m adaptive queries to $(\mathrm{Enc}_{RSS}(\zeta), G(\zeta))$. Let Y denote the number of distinct 1's in $G(\zeta)$ which are queried by A. It holds that over the randomness of ζ and Enc_{RSS},*

$$\Pr[Y \geq 2mk/n] \leq \exp(-mk/3n).$$

Proof. Note that, for any fixed ζ, any A that makes at most m adaptive queries cannot distinguish $(\mathrm{Enc}_{RSS}(\zeta), G(\zeta))$ and $(U, G(\zeta))$ where U is uniformly distributed over $\{0,1\}^{4m}$. That's because $(U, G(\zeta))$ generates any possible transcript $(i_1, b_1, \ldots, i_m, b_m)$, $(\mathrm{Enc}_{RSS}(\zeta), G(\zeta))$ with exactly the same probability due to the secrecy property of Enc_{RSS}. Because U and ζ are independent, it suffices to bound the probability that $A^{U,G(\zeta)}$ queries more than $2mk/n$ number of 1's for an arbitrary fixed choice of U and a random ζ.

Without loss of generality, we assume A queries m distinct locations of $G(\zeta)$ because any algorithm can be made into one which sees more ones from $G(\zeta)$ by querying distinct locations. Let $Y_1, \ldots, Y_m$ be indicators that $G(\zeta)$ returns 1 for these m queries made by A. Note that $Y = Y_1 + \cdots + Y_m$ and $\mathrm{E}[Y] \leq mk/n$. In addition, observe that for any $b_1, \ldots, b_m \in \{0,1\}$, $\Pr[\forall i \in [m], Y_i = b_i] = \binom{n-m}{k-|b|_0} / \binom{n}{k}$. It follows that for any set $S \subseteq [m]$, $\mathrm{E}[\Pi_{i \in S} Y_i] = \binom{n-|S|}{k-|S|} / \binom{n}{k} \leq (k/n)^{|S|}$. By the generalized Chernoff bound by Theorem 15 with $\delta = k/n$, we obtain the desired conclusion. □

We then apply proposition 6 in following two claims to bound (*), the probability that the number of bits required from the opposite side exceeds some threshold for either half of the simulated tampering function. We will handle each side separately. In particular, these claims will bound the number of queries or probes made to bits on the opposite side that depend on the input (if we fix the randomness of encoding). Because in the simulated tampering, both Alice and Bob jointly know the randomness of encoding, if a bit on the opposite side is not dependent on the input, then

both Alice and Bob know this and do not need to request its value. So, in order to complete the proof we only need to bound queries the simulator makes to the opposite side that additionally correspond to locations specified by the respective ζ (in the respective Enc*).

Claim 13. Suppose $m_R \geq (4m_L + k)t$, then for any $x \in \{0,1\}^{2k}$, the event that g_L makes more than $2(4m_L + k)tk/n_R$ queries to x_R happens with probability at most $\exp(-(4m_L + k)tk/3n_R)$.

Proof. Fix any x_L, x_R and the randomness for $\mathrm{Enc}_L(x_L)$. Note that Dec_L reads at most $4m_L + k$ coordinates from its input, $4m_L$ to reconstruct the tampered "seed" $\tilde{\zeta}$ and then the at most k locations specified by $G(\tilde{\zeta})$. Each decision tree tampering one of these bits makes at most t queries $\mathrm{Enc}_R(x_R)$ (it makes at most t queries total). Therefore g_L, in order to simulate the tampering of the bits Dec_L requires, makes at most $(4m_L + k)t$ queries to $\mathrm{Enc}_R(x_R) = (\mathrm{Enc}_{RSS}(\zeta_R), G(\zeta_R))$. By Proposition 6, g_L queries more than $2(4m_L + k)tk/n_R$ locations happens with probability at most $\exp(-(4m_L + k)tk/3n_R)$. □

Claim 14. Suppose $m_L \geq t$, then for any $x \in \{0,1\}^{2k}$, the event that g_R makes more than $2(4m_R + n_R)tk/n_L$ queries to x_L happens with probability at most $(n_R + 4m_R)t/m_L \cdot \exp(-m_L k/3n_L)$.

Proof. Fix any x_L, x_R and the randomness for $\mathrm{Enc}_R(x_R)$. Note that any subset of m_L/t outputs of f_R makes at most m_L queries to $\mathrm{Enc}_L(x_L)$. By Proposition 6, the probability that $2m_L k/n_L$ ones in $G(\zeta_L)$ are queried is at most $\exp(-m_L k/3n_L)$. We partition the output bits of f_R into $(n_R + 4m_R)t/m_L$ disjoint subsets of size m_L/t. By a union bound over these subsets, the even that f_R makes more than $2(4m_R + n_R)tk/n_L$ queries to x_L happens with probability at most $(n_R + 4m_R)t/m_L \cdot \exp(-m_L k/3n_L)$. The number of queries made by g_R to x_L is at most the queries made by f_R and the desired conclusion follows. □

Then for fixed α, there exists constants c_1, c_2, c_3 (only dependent on α) such that if we set $m_L = c_1 k \log n, n_R = m_R = c_2 tk \log n \log k$ and $n_L = c_3 t^2 k \log n \log^2 k$, then (*) (the event that the number of queries to either opposing side exceeds $\alpha k/(2(\log k + 1))$ happens with probability at most $(t^2 \log k) \cdot \exp(-\Omega(k/t^2 \log^2 k))$. Note that it follows that the rate is $k/n =$

$\Omega(1/t^2 \log^3 n)$ and for $t = O(n^{1/4}/\log^{3/2} n)$, the error can be simplified to $\exp(-\Omega(n/t^4 \log^5 n))$ because $\Omega(n/t^4 \log^5 n) = \Omega(\log n)$ and $t^2 \log k = \exp(O(\log n))$.

Chapter 7: Small-Depth Circuit Tampering

In this chapter we construct non-malleable codes for small-depth circuits.

Small-Depth Circuits. Let $\mathsf{AC}_d(S)$ denote alternating depth d circuits of size at most S with unbounded fan-in. Let w-$\mathsf{AC}_d(S)$ denote alternating depth d circuits of size at most S with fan-in at most w at the first level and unbounded fan-in elsewhere. For depth 2 circuits ($\mathsf{AC}_2(S)$), a DNF is an OR of ANDs (terms) and a CNF is an AND or ORs (clauses). The *width* of a DNF (respectively, CNF) is the maximum number of variables that occur in any of its terms (respectively, clauses). We use w-DNF (contained in w-$\mathsf{AC}_2(S)$) to denote the set of DNFs with width at most w.

We prove the following theorem.

Theorem 27. *For any constant $c \in (0,1)$, there exist constants $c_1, c_2 \in (0,1)$ such that for any $d \leq c_1 \log n / \log\log n$ and $S = \exp(n^{c_2/d})$ there is an explicit, efficient, information theoretic non-malleable code for depth d size S circuits with error $\exp(-n^{\Omega(1/d)})$ and encoding length $n = k^{1+c}$.*

Our result again proceeds by designing an efficient *non-malleable reduction* from small-depth tampering to decision tree tampering, essentially showing how we can construct an efficient non-malleable code that is secure against small-depth adversaries given one that is secure against decision-tree tampering. The theorem is then a simple corollary of this reduction and Theorem 26.

Remark 5 (On limits of extending our result). Because any function in NC^1 can be computed by a polynomial-size unbounded fan-in circuit of depth $O(\log(n)/\log\log(n))$ [91, 92], any non-trivial non-malleable code for larger depth circuits would yield a separation of NC^1 from P. Hence, in this respect Theorem 27 is the limit of what we can hope to establish given the current state of the art in circuit and complexity theory.

7.1 Technical Overview

To prove our results, we use the *non-malleable reduction* framework, introduced by Aggarwal et al. [13]. Loosely speaking, an encoding scheme (E, D) non-malleably reduces a "complex" tampering class, $\mathcal{F}$, to a "simpler" tampering class, $\mathcal{G}$, if the tampering experiment (encode, tamper, decode) behaves like the "simple" tampering (for any $f \in \mathcal{F}$, $\mathrm{D}(f(\mathrm{E}(\cdot))) \approx G_f$, a distribution over $\mathcal{G}$). [13] showed that a non-malleable code for the simpler $\mathcal{G}$, when concatenated with an (inner) non-malleable reduction (E, D) from $\mathcal{F}$ to $\mathcal{G}$, yields a non-malleable code for the more "complex" $\mathcal{F}$.

Our main technical lemma is a non-malleable reduction from tampering via small depth circuits to split-state tampering, where left and right halves of a codeword may be tampered arbitrarily, but independently. Combining with best known split state non-malleable codes [35], the result is a non-malleable code against small depth circuits.

We achieve the aforementioned reduction by composing two non-malleable reductions, with a variant of local tampering (each output bit is a function of some not-too-large set of input bits), where the choice of local tampering may depend on leakage from the codeword. First, we collapse small-depth circuit tampering to the variant of local tampering using a carefully designed pseudorandom restriction with extractable seeds (for which there exists a "switching lemma"), which may be of independent interest. Then, we demonstrate that a small modification to a construction from [2] yields non-malleability against this leaky local tampering class.

Chattopadhyay and Li [15] also used a switching lemma to reduce small-depth tampering to local tampering. [15] uses a framework from [11] to derive non-malleable codes from non-malleable extractors. An unfortunate side effect of the [11]-framework is that the rate of the code is tied to the error of the extractor. As the parameters of known switching lemmas yield (at best) quasi-polynomial error when reducing to local functions, this translates (via the [11] framework) into codes with at best exponentially small rate. Circumventing this limitation therefore necessitates a significantly different approach, and indeed, in this work we construct our NMCs that achieve

efficient rate without using extractors as an intermediary.

Small-Depth Circuits to Local Functions. Let us consider the simpler case of reducing w-DNFs (each clause contains at most w literals) to the family of leaky local functions. The reduction for general small-depth circuits will follow from a recursive composition of this reduction.

A non-malleable reduction (E, D) reducing DNF-tampering to (leaky) local-tampering needs to satisfy two conditions (i) $\Pr[\mathrm{D}(\mathrm{E}(x)) = x] = 1$ for any x and, (ii) $\mathrm{D} \circ f \circ \mathrm{E}$ is a distribution over (leaky) local functions for any width-w DNF f. A classic result from circuit complexity, the switching lemma [93, 94, 95, 12], states DNFs become local functions under random restrictions ("killing" input variables by independently fixing them to a random value with some probability).[1] Thus a natural choice of E for satisfying (ii) is to simply sample from the generating distribution of restrictions and embed the message in the surviving variable locations (fixing the rest according to restriction). However, although $f \circ \mathrm{E}$ becomes local, it is not at all clear how to decode and fails even (i). To satisfy (i), a naive idea is to simply append the "survivor" location information to the encoding. However, this is now far from a random restriction (which requires among other things that the surviving variables are chosen independently of the random values used to fix the killed variables) and consequently no longer guaranteed to "switch" the DNFs to Local functions with overwhelming probability.

To circumvent those limitations, we consider *pseudorandom switching lemmas*, which arose in the context of derandomization [96, 97, 98, 99, 78, 100], to relax the stringent properties of the distribution of random restrictions needed for classical switching lemmas. In particular, we invoke a pseudorandom switching lemma from Trevisan and Xue [78], which reduces DNFs to local functions while only requiring that randomness specifying survivors and fixed values be σ-wise independent. This allows us to avoid problems with independence arising in the naive solution above. Now, we can append a σ-wise independent encoding of the (short) random seed that specifies the surviving variables. This gives us a generating distribution of random restrictions

[1]The switching lemma actually shows that DNFs become *small-depth decision trees* under random restrictions. However, it is this (straightforward) consequence of the switching lemma that we will use in our reduction.

such that (a) DNFs are switched to Local functions, and (b) the seed can be decoded and used to extract the input locations.

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

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

7.2 Reducing Small-Depth Circuits to Decision-Trees

To capture our reduction in a more general sense, we introduce yet another "leaky" tampering model. In particular, given an arbitrary class of tampering functions, we consider a variant of the class of tampering functions which may depend in some limited way on limited leakage from the

underlying code word.

Definition 31 (Leaky Function Families)**.** Let $\mathrm{LL}^{i,m,N}[C]$ denote tampering functions generated via the following game:

1. The adversary first commits to N functions from a class C, $F_1, \ldots, F_N = \boldsymbol{F}$.

   (Note: $F_j : \{0,1\}^N \to \{0,1\}$ for all $j \in [N]$.)

2. The adversary then has i-adaptive rounds of leakage. In each round $j \in [i]$,

   - the adversary selects s indices from $[N]$, denoted S_j,
   - the adversary receives $\boldsymbol{F}(x)_{S_j}$.

   Formally, we take $h_j : \{0,1\}^{m(j-1)} \to [N]^m$ to be the selection function such that

   $$h_j(F(X)_{S_1}, \ldots, F(X)_{S_{j-1}}) = S_j.$$

   Let h_1 be the constant function that outputs S_1.

3. Finally, selects a sequence of n functions $(F_{t_1}, \ldots, F_{t_n})$ ($T = \{t_1, \ldots, t_n\} \subseteq [N]$ such that $t_1 < t_2 < \cdots < t_n$) to tamper with.

   Formally, we take $h : \{0,1\}^{mi} \to [N]^n$ such that $h(F(X)_{S_1}, \ldots, F(X)_{S_i}) = T$.

Thus, any $\tau \in \mathrm{LL}^{i,m,N}[C]$ can be described via $(\boldsymbol{F}, h_1, \cdots, h_i, h)$. In particular, we take $\tau = \mathrm{Eval}(\boldsymbol{F}, h_1, \cdots, h_i, h)$ to denote the function whose output given input X is $T(X)$, where T is, in turn, outputted by the above game given input X and adversarial strategy $(\boldsymbol{F}, h_1, \cdots, h_i, h)$.

We note that the "leakage" is simply a restricted form $i \cdot m$ adaptive queries to depth-t decision trees. Thus, $\mathrm{LL}^{i,m,n}[\mathrm{DT}(t)] \subseteq \mathrm{DT}(imt)$. Therefore non-malleable code for large depth decision trees immediately yields a new non-malleable code for small depth circuits (with improved error). However, $\mathrm{LL}^{i,m,n}[\mathrm{DT}(t)]$ gives decision trees that are identical excepting (up to) the last t queries before output (and that the last t-queries must be consistent with one of n depth-t decision trees).

Lemma 16. *For $S, d, n, t \in \mathbb{N}$, $p, \delta \in (0,1)$, there exist*

$$\sigma = \text{poly}(t, \log(2^t S), \log(1/\delta), \log(1/p))^2$$

and $m = O(\sigma \log n)$ such that, for any $2m \leq k \leq n(p/4)^d$,

$$(\mathsf{AC}_d(S) \implies \text{LL}^{d,m,n}[\text{DT}(t)], d\varepsilon)$$

where

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

For constant-depth polynomial-size circuits (i.e.,AC^0), by setting $p = n^{-O(1/d)}$ (such as $n^{-1/100d}$), $t' = 1/40p$ and $\delta = \exp(n^{-\Omega(1/d)})$, the error of our non-malleable reduction, Lemma 16, is $\exp(-n^{\Omega(1)})$.

The same setting of parameters lead to non-malleable reduction for circuits of depth as large as $\Theta(\log(n)/\log\log(n))$ and size $S = \exp(n^{O(1/d)})$ with error $\exp(-n^{\Omega(1/d)})$. Combining our reduction, Lemma 16, with Theorem 26 yields our main theorem for this chapter.

Lemma 16 will ultimately proven by a recursive application of a more general Lemma 17 that allows generically transforms pseudorandom switching lemmas (Section 2.6.6) into non-malleable reductions. However before continuing we introduce some additional notation that will streamline our presentation.

Helpful Functions. For a random restriction $\rho = (\rho^{(1)}, \rho^{(2)}) \in \{0,1\}^n \times \{0,1\}^n$, $\text{ExtIndices}(\rho^{(1)}) := (i_1, \ldots, i_k) \in [n+1]^k$ are the last k indices of 1s in $\rho^{(1)}$ where $i_1 \leq i_2 \cdots \leq i_k$ and $i_j = n+1$ for $j \in [k]$ if such index doesn't exist (k should be obvious from context unless otherwise noted).

We define a pair of functions for embedding and extracting a string x according to a random restriction, ρ. Let $\text{Embed} : \{0,1\}^{k+2n} \to \{0,1\}^n$, such that for $\rho = (\rho^{(1)}, \rho^{(2)}) \in \{0,1\}^n \times \{0,1\}^n$ and $x \in \{0,1\}^k$, and $i \in [n]$,

[2]The exponent of this polynomial is a fixed absolute constant independent of all other parameters.

$$\text{Embed}(x,\rho)_i = \begin{cases} x_j & \text{if } \exists j \in [k] : i = \text{ExtIndices}(\rho^{(1)})_j \\ \rho_i^{(2)} & \text{otherwise} \end{cases}$$

And, let Extract : $\{0,1\}^{2n} \to \{0,1\}^k \times \{\perp\}$ be such that if $c \in \{0,1\}^n, \rho^{(1)} \in \{0,1\}^n$, and $\|\rho^{(1)}\|_0 \geq k$, then $\text{Extract}(c,\rho^{(1)}) = c_{\text{ExtIndices}(\rho^{(1)})}$. Otherwise, $\text{Extract}(c,\rho^{(1)}) = \perp$.

Note that, for any ρ such that $\|\rho^{(1)}\|_0 \geq k$, $\text{Extract}(\text{Embed}(x,(\rho^{(1)},\rho^{(2)})),\rho^{(1)}) = x$.

7.2.1 Proof of Lemma 17

The simple encoding and decoding scheme based on the pseudorandom switching lemma is defined in Figure 7.1.

Lemma 17. *Let $\mathcal{F}$ and $\mathcal{G}$ be two classes of functions. Suppose for $n \in \mathbb{N}$, $p \in (0,1)$ and any σ-wise independent distribution $\mathcal{D}$ over $\{0,1\}^{n\log(1/p)} \times \{0,1\}^n$, it holds that for any $F\colon \{0,1\}^n \to \{0,1\} \in \mathcal{F}$,*

$$\Pr_{\rho \leftarrow \mathcal{R}(\mathcal{D})}\left[F_\rho \text{ is not in } \mathcal{G}\right] \leq \varepsilon.$$

Then for $i,N,k \in \mathbb{N}$, $(\mathrm{E}^\star_{k,n,p,\sigma}, \mathrm{D}^\star_{k,n,p,\sigma})$ defined in figure 7.1 is an

$$(\mathrm{LL}^{i,m,N}[\mathcal{F}] \implies \mathrm{LL}^{i+1,m,N}[\mathcal{G}], N\varepsilon + \exp(-\frac{\sigma}{2\log(1/p)}))$$

non-malleable reduction when $(4\sigma/\log(1/p)) \leq k \leq (n-m)p/2$.

The Lemma follows immediately from Claims 15, 16, and 17 below.

Claim 15. For any $x \in \{0,1\}^k$, $\Pr[\mathrm{D}^*(\mathrm{E}^*(x)) = x] = 1$.

Proof. The second step of E^* guarantees that $\text{ExtIndices}(L(G(\zeta)))_1 > m$ and $\|L(G(\zeta)\|_0 \geq k$. Therefore, $\mathrm{E}_R(\zeta)$ is located in the first m bits of c and the entire x is embedded inside the remaining $n-m$ bits of c according to $L(G(\zeta))$. By the decoding property of RPE from lemma 4, $\Pr[\mathrm{D}_R(c,\ldots,c_m) = \zeta] = 1$, namely, $\Pr[\tilde{\zeta} = \zeta] = 1$. Conditioned on $\tilde{\zeta} = \zeta$, because $\|L(G(\zeta)\|_0 \geq k$, $\mathrm{D}^*(\mathrm{E}^*(x)) = \text{Extract}(c, L(G(\zeta))) = x$ holds. The desired conclusion follows. □

Take k, n, p, σ to be parameters.
Let $G = G_\sigma \colon \{0,1\}^{s(\sigma)} \to \{0,1\}^{n \log 1/p}$ be an σ-wise independent generator from Lemma 7.
Let $(\mathrm{E}_R, \mathrm{D}_R, \mathrm{R}_R)$ denote the RPE from lemma 4 with codewords of length $m(s) \geq \sigma/c^{\mathsf{sec}}$.
Let $\zeta^* \in \{0,1\}^{s(\sigma)}$ be some fixed string such that $\|L(G(\zeta^*))_{n-m+1,\ldots,n}\|_0 \geq k$. (For our choice of G, such a ζ^* can be found efficiently via interpolation.)

$\mathrm{E}^\star(x)$**:**

1. Draw (uniformly) random seed $\zeta \leftarrow \{0,1\}^s$ and (uniformly) random string $U \leftarrow \{0,1\}^{n-m}$.
2. Generate pseudorandom restriction, $\rho = (\rho^{(1)}, \rho^{(2)})$:
   $\rho^{(1)} \leftarrow L(G(\zeta))$; $(*)$ If $\|L(G(\zeta))_{n-m+1,\ldots,n}\|_0 < k$, set $\zeta = \zeta^*$.
   $\rho^{(2)} \leftarrow \mathrm{E}_R(\zeta) \| U$.
3. Output $c = \mathrm{Embed}(x, \rho)$.

$\mathrm{D}^\star(\tilde{c})$**:**

1. Recover tampered seed: $\tilde{\zeta} \leftarrow \mathrm{D}_R(\tilde{c}_1, \ldots, \tilde{c}_m)$.
   If $\|L(G(\tilde{\zeta}))_{n-m+1:n}\|_0 < k$, output $\perp$ and halt.
2. Output $\mathrm{Extract}\big(\tilde{c}, L(G(\tilde{\zeta}))\big)$.

Figure 7.1: A Pseudorandom Restriction Based Non-Malleable Reduction, $(\mathrm{E}^\star_{k,n,p,\sigma}, \mathrm{D}^\star_{k,n,p,\sigma})$

given $\mathrm{LL}^{i,m,N}[\mathcal{F}]$ tampering $\tau = (\boldsymbol{F}, h_1, \ldots, h_i, h)$ output $\tau' = (\boldsymbol{F}', h'_1, \ldots, h'_{i+1}, h')$:

1. Draw (uniformly) random seed $\zeta \leftarrow \{0,1\}^s$ and (uniformly) random string $R \leftarrow \{0,1\}^{n-m}$.
2. Generate pseudorandom restriction, $\rho = (\rho^{(1)}, \rho^{(2)})$:
   $\rho^{(1)} \leftarrow L(G(\zeta))$. $(*)$ If $\|L(G(\zeta))_{n-m+1,\ldots,n}\|_0 < k$, set $\zeta = \zeta^*$.
   $\rho^{(2)} \leftarrow \mathrm{E}_R(\zeta) \| R$
3. Apply (constructive) switching lemma with pseudorandom restriction to get function $\boldsymbol{F}' \equiv \boldsymbol{F}|_\rho$ (n-bit output).
   If $\boldsymbol{F}$ is not in $\mathcal{G}$, halt and output some constant function.
4. For $j \in [i]$, $h'_j \equiv h_j$.
5. $h'_{i+1}(y'_1, \ldots, y'_i) := h(y'_1, \ldots, y'_i)_{[m]}$.
6. $h'(y'_1, \ldots, y'_{i+1}) := h(y'_1, \ldots, y'_i)_{\mathrm{ExtIndices}(L(G(\mathrm{D}_R(y'_{i+1}))))}$.
7. Finally, output $\tau' = (\boldsymbol{F}', h'_1, \ldots, h'_{i+1}, h')$.

Figure 7.2: Simulator, Sim, for $(\mathrm{E}^\star, \mathrm{D}^\star)$

Claim 16. Given any $\tau = \mathrm{Eval}(\boldsymbol{F}, h_1, \ldots, h_i, h) \in \mathrm{LL}^{i,m,N}[\mathcal{F}]$, there is a distribution S_τ over $\tau' \in \mathrm{LL}^{i+1,m,N}[\mathcal{G}]$, such that for any $x \in \{0,1\}^k$, $\mathrm{D}^\star \circ \tau \circ \mathrm{E}^\star(x)$ is δ-close to $\tau'(x)$ where $\tau' \leftarrow \mathsf{Sim}_\tau$

and $\delta \leq \Pr[\boldsymbol{F} \circ \mathrm{E}^* \text{ is not in } \mathcal{G}]$.

Proof. Recall that a function τ in $\mathrm{LL}^{i,m,N}[\mathcal{F}]$ can be described via $(\boldsymbol{F}, h_1, \ldots, h_i, h)$ where $\boldsymbol{F}$ is a function in $\mathcal{F}$ from $\{0,1\}^k$ to $\{0,1\}^N$ and for every $x \in \{0,1\}^k$, h takes $\boldsymbol{F}(x)_{S_1}, \ldots, \boldsymbol{F}(x)_{S_i}$ (where S_j are sets adaptively chosen by h_j for $j \in [i]$) as input and outputs a set T of size k. And the evaluation of τ on x is $\boldsymbol{F}(x)_T$.

Let Sim_τ be defined in Figure 7.2. We call a chocie of randomness ζ, U, r "good for $\boldsymbol{F} = (F_1, \cdots, F_N)$" (where r is the randomness for E_R) if $\boldsymbol{F} \circ \mathrm{E}^\star(\cdot; \zeta, U, r)$ is in $\mathcal{G}$. We will show for any good ζ, U, r for $\boldsymbol{F}$, $\mathrm{D}^\star \circ \tau \circ \mathrm{E}^\star(\cdot; \zeta, U, r) \equiv \tau'(\cdot)$, where $\tau' = S_\tau(\zeta, U, r)$.

For good ζ, U, r, note that (1) $\boldsymbol{F}' \equiv \boldsymbol{F}|_\rho$ and (2) ρ was used in both $\mathrm{E}^\star$ and S_τ. It follows that for all x, $\boldsymbol{F}'(x) = \boldsymbol{F}|_\rho(x) = \boldsymbol{F}(\mathrm{E}^\star(x; \zeta, R, r))$. Because $h'_j \equiv h_j$ for $j \in [i]$, it follows by induction that $y'_j = y_j$ (the output of each h'_j and h_j respectively, $j \in [i]$). Therefore, $h(y_1, \cdots, y_i) = h(y'_1, \cdots, y'_i)$. It follows that $\tilde{c}_{[m]} = y'_{i+1}$ and $L(G(\mathrm{D}_R(y'_{i+1}))) = L(G(\tilde{\zeta}))$. Consequently, $h'(y'_1, \cdots, y'_{i+1})$ outputs that exact same indices that the decoding algorithm, $\mathrm{D}^\star$, will extract its output from. Thus, $\tau'(x) = \mathrm{D}^\star \circ \tau \circ \mathrm{E}^\star(x; \zeta, R, r)$ for any x.

Because Sim and $\mathrm{E}^\star$ sample their randomness identically, the distributions are identical, conditioned on the randomness being "good." Hence δ is at most the probability that ζ, U, r are not "good for $\boldsymbol{F}$", i.e., $\Pr[\boldsymbol{F} \circ \mathrm{E}^* \text{ is not in } \mathcal{G}]$.

□

Claim 17. $\Pr[\boldsymbol{F} \circ \mathrm{E}^* \text{ is not in } \mathcal{G}] \leq N\varepsilon + \exp(-\sigma/2 \log(1/p))$.

Proof. We first show $\mathcal{D} = G(\zeta) \| \mathrm{E}_R(\zeta) \| U$ is σ-wise independent when $\zeta \leftarrow \{0,1\}^s$ and $U \leftarrow \{0,1\}^{n-m}$. As U is uniform and independent of the rest, it suffices to simply consider $Z = G(\zeta) \| \mathrm{E}_R(\zeta)$. Fix some $S \subseteq [n \log(1/p) + m]$ such that $|S| \leq \sigma$. By the secrecy property of the RPE and $m \cdot c^{\mathsf{sec}} \geq \sigma$, conditioned on any fixed ζ, $Z_{S \cap \{n \log(1/p)+1, \ldots, n \log(1/p)+m\}}$is distributed uniformly. Therefore, ζ is independent of $Z_{S \cap \{n \log(1/p)+1, \ldots, n \log(1/p)+m\}}$, so G guarantees that $Z_{S \cap \{1, \ldots, n \log(1/p)\}}$ is independently of $S \cap \{n \log(1/p) + 1, \ldots, n \log(1/p) + m\}$ and also distributed uniformly. Therefore, Z_S is distributed uniformly.

Note that ρ in E^* is distributed identically to $\mathcal{R}(\mathcal{D})$, except when ζ^* is used. Hence

$$\Pr[\boldsymbol{F} \circ \mathrm{E}^* \text{ is not in } \mathcal{G}] \leq \Pr_{\rho \leftarrow \mathcal{R}(\mathcal{D})} \left[\boldsymbol{F}_\rho \text{ is not in } \mathcal{G}\right] + \Pr[\|L(G(\zeta))_{n-m+1,\ldots,n}\|_0 < k].$$

By our assumption and a union bound over the N boolean functions, $\boldsymbol{F}_\rho \notin \mathcal{G}$ happens with probability at most $N\varepsilon$ when $\rho \leftarrow \mathcal{R}(\mathcal{D})$. Observe that $L(G(\zeta))_{n-m+1,\ldots,n}$ is a $\frac{\sigma}{\log(1/p)}$-wise independent distribution over $\{0,1\}^{n-m}$ and each coordinate is 1 with probability p. Let $\mu = (n-m)p$ denote the expected number of 1's in $L(G(\zeta))_{n-m+1,\ldots,n}$. By linearity of expectation $\mu = (n-m)p$. For $k \leq \mu/2$ and $\frac{\sigma}{\log(1/p)} \leq \mu/8$, we can use the concentration bound from Theorem 17 to conclude that $\|L(G(\zeta))_{n-m+1,\ldots,n}\|_0 < k$ happens with probability at most $\exp(-\frac{\sigma}{2\log(1/p)})$. The desired conclusion follows. □

7.2.2 Proof of Lemma 16

To prove Lemma 16, we instantiate Lemma 17 using the pseudorandom switching lemma of Theorem 14 (in fact, Corollary 1) and iteratively reduce $\mathsf{AC}_d(S)$ to leaky local functions. Each application of the reduction, after the first, will allow us to trade a level of depth in the circuit for an additional round of leakage until we are left with a depth-2 circuit. The final application of the reduction will allow us to convert this circuit to local functions at the expense of a final round of leakage.

Let $t := \log(\ell)$ and let $\sigma := \mathrm{poly}(t, \log(2^t S), \log(1/\delta), \log(1/p))$ as in Corollary 1 so that any depth-2 circuits with bottom fan-in t become depth t decision trees with probability at least $1-(2^{2t+1}(5pt)^t+\delta)$ under pseudorandom restrictions drawn from σ-wise independent distribution.

We use $\mathsf{AC}_d(S) \circ \mathrm{DT}(t)$ to denote alternating (unbounded fan-in) circuits of depth d, size S that take the output of depth t decision trees as input. (Note may contain up to S decision trees.) Similarly it is helpful to decompose an alternating circuit (from w-AC_d) into a base layer of CNFs or DNFs and the rest of the circuit, $\mathsf{AC}_{d-2}(S) \circ w\text{-}\mathsf{AC}_2(S')$. (Again, the base may contain up to S CNFs/DNFs of size S'.)

Claim 18. $(\mathsf{AC}_d(S) \implies \mathrm{LL}^{1,m,n}[\mathsf{AC}_{d-2}(S) \circ t\text{-}\mathsf{AC}_2(2^t S)], \varepsilon)$.

Proof. Let $F \in \mathsf{AC}_d(S)$ be a boolean function. Note that Theorem 14 and Corollary 1 are only useful for bounded width DNF and CNF. So, we view F as having an additional layer of fan-in 1 AND/OR gates, namely, as a function in $1\text{-}\mathsf{AC}_{d+1}(S)$. Because there are at most S DNFs (or CNFs) of size S at the bottom layers of F, by Corollary 1, the probability that F is not in $\mathsf{AC}_{d-1}(S) \circ \mathrm{DT}(t)$ is at most $S\left(2^{t+2}(5p)^t + \delta\right)$ under the pseudorandom switching lemma with parameters p, δ, σ. So by Corollary 1, $(\mathrm{E}^\star, \mathrm{D}^\star)$ reduces $\mathsf{AC}_d(S)$ to $\mathrm{LL}^{1,m,n}[\mathsf{AC}_{d-1}(S) \circ \mathrm{DT}(t)]$ with error $n(S\left(2^{t+2}(5p)^t + \delta\right)) + \exp(-\Omega(\frac{\sigma}{\log(1/p)})) \leq \varepsilon$.

By the fact that $\mathrm{DT}(t)$ can be computed either by width-t DNFs or width-t CNFs of size at most 2^t, any circuit in $\mathsf{AC}_{d-1}(S) \circ \mathrm{DT}(t)$ is equivalent to a circuit in $\mathsf{AC}_{d-2}(S) \circ t\text{-}\mathsf{AC}_2(2^t S)$, in other words, a depth d circuit with at most S width-t size-$S2^t$ DNFs or CNFs at the bottom. Hence, $\mathsf{AC}_{d-1}(S) \circ \mathrm{DT}(t)$ is a subclass of $\mathsf{AC}_{d-2}(S) \circ t\text{-}\mathsf{AC}_2(2^t S)$ and the claim follows. □

Claim 19. $(\mathrm{LL}^{i,m,n}[\mathsf{AC}_{d-i-1}(S) \circ t\text{-}\mathsf{AC}_2(2^t S)] \implies \mathrm{LL}^{i+1,m,n}[\mathsf{AC}_{d-i-2}(S) \circ t\text{-}\mathsf{AC}_2(2^t S)], \varepsilon)$.

Proof. For a boolean function $F \in \mathsf{AC}_{d-i-2}(S) \circ t\text{-}\mathsf{AC}_2(2^t S)$, because there are at most S DNFs (or CNFs) of size $2^t S$ at the bottom layers of F, Corollary 1 shows F is not in $\mathsf{AC}_{d-2}(S) \circ \mathrm{DT}(t)$ with probability at most $S\left(2^{2t+2}(5pt)^t + \delta\right)$ under a pseudorandom switching lemma with paramemters p, δ, σ. So by Lemma 17, $(\mathrm{E}^\star, \mathrm{D}^\star)$ reduces $(\mathrm{LL}^{i,m,n}[\mathsf{AC}_{d-i-2}(S) \circ t\text{-}\mathsf{AC}_2(2^t S)]$ to $\mathrm{LL}^{i+1,m,n}[\mathsf{AC}_{d-i-2}(S) \circ \mathrm{DT}(t)]$ with error at most ε. Similarly as the previous proof, because $\mathsf{AC}_{d-i-2}(S) \circ \mathrm{DT}(t)$ is a subclass of $\mathsf{AC}_{d-i-3}(S) \circ \mathsf{AC}_2(2^t S)$, the claim follows. □

Claim 20. $(\mathrm{LL}^{d-1,m,n}[t\text{-}\mathsf{AC}_2(2^t S)] \implies \mathrm{LL}^{d,m,n}[\mathrm{Local}^{2^t}], \varepsilon)$

Proof. Finally, for a boolean function $F \in t\text{-}\mathsf{AC}_2(2^t S)$, Corollary 1 shows F is not in $\mathrm{DT}(t)$ with probability at most $S\left(2^{2t+2}(5pt)^t + \delta\right)$. So by Lemma 17, $(\mathrm{E}^\star, \mathrm{D}^\star)$ reduces $\mathrm{LL}^{d-1,m,n}[t\text{-}\mathsf{AC}_2(2^t S)]$ to $\mathrm{LL}^{d,m,n}[\mathrm{DT}(t)]$ with error at most ε. The desired conclusion follows from the fact that $\mathrm{DT}(t)$ is a subclass of Local^{2^t}. □

By applying Claim 18 once, then Claim 19 $(d-2)$ times and Claim 20 once, $\mathsf{AC}_d(S)$ reduces to $\mathrm{LL}^{d,m,n}[\mathrm{Local}^{2^t}]$ with error at most $d\varepsilon$. Note that $m = O(\sigma \log n)$ throughout, and during each application of above claims, given a codeword of length $n' \geq k \geq 2m$, Lemma 17 holds for messages of length $(n'-m)p/2 \geq n'(p/4)$. Therefore, the composed reduction works for any $2m \leq k \leq n(p/4)^d$.

Chapter 8: Low-Degree Polynomial Tampering

In this chapter we construct non-malleable codes resilient to tampering by low-degree polynomials. In contrast to prior chapters, here we consider the codeword alphabet to, accordingly, consist of field elements. Our techniques also follow a very different framework. In contrast to previous chapters which constructed a sequence of *non-malleable reductions*, here we follow a paradigm introduced by Cheraghchi and Guruswami [11] and construct an invertible seedless non-malleable extractor that is resilient to tampering by low degree polynomials.

Polynomial Tampering. Let q be a prime, and $\text{Poly}_{n,q,d}$ denote the family of n-variate polynomials over $\mathbb{F}_q$ of degree at most d. We are interested in the following family of tampering functions:

$$\mathcal{F}_{n,q,d} := \{(p_1, \ldots, p_n) : \forall i \in [n], p_i \in \text{Poly}_{n,q,d}\}.$$

For $P = (p_1, \ldots, p_n) \in \mathcal{F}_{n,q,d}$, and $x \in \mathbb{F}_q^n$, define $P(x) := (p_1(x), \ldots, p_n(x))$.

The following is our main result.

Theorem 28 (NMCs for bounded-degree polynomials)**.** *There exists a constant $C > 0$ such that for all integers n, d, m, any $\epsilon > 0$ and any prime $q > (Cn^2d^4m2^{2m}/\epsilon^2) \cdot \log(nd/\epsilon)$, there exists a non-malleable code on alphabet $[q]$, with block length n, message length m, relative rate $\Omega(m/n \log q)$ and error ϵ, with respect to the family $\mathcal{F}_{n,q,d}$.*

To prove Theorem 28, we construct new explicit seedless non-malleable extractors that can handle the tampering class $\mathcal{F}_{n,q,d}$. A similar strategy was adopted in [15], where they constructed seedless non-malleable extractors against affine tampering functions (i.e, $\mathcal{F}_{n,q,1}$). However, their construction of such extractors heavily exploit the linearity of the tampering functions and explicit constructions of extractors that are linear, and their techniques seem to break down even against

quadratic tampering functions. We introduce a completely different approach to construct seedless non-malleable extractors against higher degree polynomial tampering.

We use Theorem 28 to derive a non-malleable code that is secure against tampering by arithmetic circuits. Consider the following family of tampering functions:

$$\mathcal{E}_{n,q,s} := \{(e_1, \ldots, e_n) : \ e_i \text{ is an } n\text{-variate size-}s \text{ arithmetic circuit over } \mathbb{F}_q\}.$$

For $E = (e_1, \ldots, e_n) \in \mathcal{E}_{n,q,s}$ and $x \in \mathbb{F}_q^n$, we define $E(x) := (e_1(x), \ldots, e_n(x))$.

Corollary 7 (NMCs for arithmetic circuits)**.** *There exists a constant $C > 0$ such that for all integers n, s, m, any $\epsilon > 0$ and any prime $q > (Cn^2 sm 2^{4s+2m}/\epsilon^2) \cdot \log(n/\epsilon)$, there exists a non-malleable code on alphabet $[q]$, with block length n, message length m, relative rate $\Omega(m/n \log q)$ and error ϵ, with respect to the family $\mathcal{E}_{n,q,s}$.*

Theorem 7 follows as a straightfoward consequence of Theorem 28, using the fact that a size-s arithmetic circuit computes a polynomial of degree at most 2^s.

As mentioned above, to construct this code we first construct a seedless non-malleable extractor.

Theorem 29. *There exists a constant $C > 0$ such that for all integers n, d, m, any prime q and any $\epsilon > 0$ such that $q > (Cn^2 d^4 m 2^{2m}/\epsilon^2) \cdot \log(nd/\epsilon)$, there exists an explicit function* $\mathrm{NMExt} : \mathbb{F}_q^n \to \{0,1\}^m$*, that is a seedless non-malleable extractor with respect to the family of sources*

$$\mathcal{X} = \{X : X \textit{ is a skew affine source on } \mathbb{F}_q^n \textit{ of dimension} \geq 1\}$$

and the tampering family $\mathcal{F}_{n,q,d}$.

We note that our seedless non-malleable extractor can also be used to construct non-malleable secret sharing schemes resilient to polynomial tampering using a framework introduced by Lin et al. [17]. In brief, a non-malleable secret sharing scheme is a threshold secret sharing scheme that

is resilient to tampering attacks performed jointly across all shares. We refer the reader to [101] for the full definition and details.

As is the case above, the existence of seedless non-malleable extractors for arithmetic circuit tamepring is an immediate corollary of Theorem 29.

Corollary 8. *There exists a constant $C > 0$ such that for all integers n, s, m, any prime q and any $\epsilon > 0$ such that $q > (Cn^2 sm 2^{4s+2m}/\epsilon^2) \cdot \log(n/\epsilon)$, there exists an explicit function* $\mathrm{NMExt} : \mathbb{F}_q^n \to \{0,1\}^m$*, that is a seedless non-malleable extractor with respect to the*

$$\mathcal{X} = \{X : X \text{ is a skew affine source on } \mathbb{F}_q^n \text{ of dimension} \geq 1\}$$

and the tampering family $\mathcal{E}_{n,q,s}$.

8.1 Technical Overview

In this section we discuss the main ideas that are used in our explicit constructions of non-malleable codes, non-malleable extractors, and non-malleable secret sharing schemes. We start by discussing the explicit non-malleable extractor against polynomial tampering (Theorem 29). We then discuss ideas that go into using this construction to construct efficient non-malleable codes and non-malleable secret sharing schemes that are robust to polynomial tampering.

Seedless non-malleable extractors against polynomials. We discuss the main ideas behind the construction of the non-malleable extractor from Theorem 29. We consider the simpler setting and assume the the source is uniform (instead of being a skew affine source as in Theorem 29). This setting cleanly captures our main ideas. The setup is as follows:

Let n, d be arbitrary integers, and fix any $\epsilon > 0$. Let $q = \mathrm{poly}(n, d, 1/\epsilon)$ be a large enough prime (for exact details, see the statement of Theorem 29). Let X be the uniform distribution on $\mathbb{F}_q^n$. Our goal is to construct a polynomial time function $\mathrm{NMExt} : \mathbb{F}_q^n \to \{0,1\}^m$ such that for any tampering function $P = (p_1, \ldots, p_n)$ from the class $\mathcal{F}_{n,q,d}$, such that there exists $i \in [n]$ for which

$p_i(x) \neq x_i$, we have

$$|(\mathrm{NMExt}(X), \mathrm{NMExt}(P(X))) - (\mathrm{U}_m, \mathrm{NMExt}(P(X)))| \leq \epsilon.$$

The high level idea of our construction is to observe that we can express X as a convex combination of distributions that are flat[1] on lines in $\mathbb{F}_q^n$, and then design a non-malleable extractor for such line sources. We note that Gabizon and Raz [53] used such an approach for constructing affine extractors on large fields.

We now describe our approach more precisely. Our plan is to construct a low-degree multivariate polynomial $h : \mathbb{F}_q^n \to \mathbb{F}_q$ such that the following hold: for all $\beta \in \mathbb{F}_q$, the polynomial

$$g_\beta = h(x) + \beta h(P(x))$$

is non-constant. (We stress that the choice of h cannot depend on P.) Now, for a suitable choice of m (we pick $m = \nu \log q$ for some small enough ν), we claim that for such an h, defining

$$\mathrm{NMExt}(x) = h(x) \pmod{2^m}$$

would satisfy the conclusion of Theorem 29.

Before constructing such an h, we first discuss why this is indeed enough. For any $a \in \mathbb{F}_q^n$, $b \in \mathbb{F}_q^n \setminus \{0^n\}$, define the line $L_{a,b} = \{(a_1 + tb_1, \ldots, a_n + tb_n) : t \in \mathbb{F}_q\}$. We abuse notation, also use $L_{a,b}$ to denote the flat distribution on $L_{a,b}$. Then clearly, X can be sampled by first uniformly sampling a, b (from their respective domains), and then sampling from $L_{a,b}$.

The first observation is the following: let $D = deg(g_\beta)$, and let $g_{\beta,a,b}(t)$ be the univariate restriction of g_β to the line $L_{a,b}$. We note that the coefficient of t^D is $g_\beta(b)$. Appealing to the fact that a low degree polynomial has few roots (Lemma 11), it follows that with high probability (over sampling a, b), the univariate polynomial $g_{\beta,a,b}(t)$ is a non-constant polynomial of degree D.

[1] We say a distribution is *flat* if it is uniformly distributed on its support.

Fix such vectors a, b so that $g_{\beta,a,b}$ is a non-constant polynomial. We now use a deep result from algebraic geometry known as the Weil bound (see Theorem 11) to conclude that for any non-trivial character[2] χ of $\mathbb{F}_q$, we have

$$|\mathbb{E}_{t\sim\mathbb{F}_q}[\chi(g_{\beta,a,b}(t))]| \leq D/\sqrt{q}.$$

Roughly, this asserts the fact that the non-trivial Fourier coefficients of the distribution $g_{\beta,a,b}(U_{\mathbb{F}_q})$ are bounded, where $U_{\mathbb{F}_q}$ denotes the uniform distribution on $\mathbb{F}_q$. Such a bound can be now be translated into statistical closeness of the distribution $(\text{NMExt}(L_{a,b}), \text{NMExt}(P(L_{a,b})))$ to $(\text{U}_m, \text{NMExt}(P(L_{a,b})))$ using known XOR lemmas (see Lemma 8, Lemma 9). To conclude that $(\text{NMExt}(X), \text{NMExt}(P(X)))$ is close to $(\text{U}_m, \text{NMExt}(P(X)))$, we combine the fact that X is a convex combination of the flat sources $L_{a,b}$, and that for most a, b, we have $(\text{NMExt}(L_{a,b}), \text{NMExt}(P(L_{a,b})))$ is close to $(\text{U}_m, \text{NMExt}(P(L_{a,b})))$.

Given the above discussion, all that remains to construct the required non-malleable extractor is to find such an h. We recall the guarantee we need from h for convenience of the reader:

- for all $\beta \in \mathbb{F}_q$ and $P = (p_1, \ldots, p_n) \in \mathcal{F}_{n,q,d}$ satisfying that for some $i \in [n]$ $p_i(x) \neq x_i$, the polynomial $g_\beta(x) = h(x) + \beta h(P(x))$ is a non-constant polynomial.
- h must a low degree polynomial. In particular, we require $deg(h) \ll q^{1/2}$.

An initial attempt to construct such an h could be to use a polynomial similar to the one used by Gabizon and Raz [53] in their affine extractor construction and define

$$h(x_1, x_2, \ldots, x_n) = x_1^{c_1} + x_2^{c_2} + \ldots + x_n^{c_n},$$

where $c_1, c_2, \ldots, c_n$ are arbitrary distinct positive integer. It is not hard to see that this does not work as follows. It is always possible to find $\beta, \gamma_1, \gamma_2, \ldots, \gamma_n \in \mathbb{F}_q^*$ such that $\gamma_i^{c_i} = -\beta^{-1}$ for every i and $\gamma_i \neq 1$ for at least one i. Now defining $P = (\gamma_1 x_1, \ldots, \gamma_n x_n)$ gives the desired counterexample since for this choice of β and P, $h(x) + \beta h(P(x))$ is identically the zero polynomial.

[2] See Section 2.6.4 for a quick recap of characters of finite fields.

We avoid the above counterexample as follows: Pick $c_1, c_2, \ldots, c_{2n}$ from an arithmetic progression such that the common difference is co-prime with $q-1$, and define

$$h(x_1, x_2, \ldots, x_n) = \sum_{i=1}^{n} \left(x_i^{c_{2i-1}} + x_i^{c_{2i}} \right).$$

For this choice of h, it is not hard to prove that if each $p_i(x) = \gamma_i x_i$ (for some $\gamma_i \in \mathbb{F}_q$), and $g(x)$ is a constant polynomial, it must be that each γ_i is 1, and $\beta = -1$. However this contradicts our assumption on P that for some i, $p_i(x) \neq x_i$. Thus we avoid the counterexample discussed above.

We in fact prove that this choice of h works for all $P \in \mathcal{F}_{n,q,d} \setminus \{(x_1, \ldots, x_n)\}$. To prove this, we rely on a result (Lemma 10) which shows that for such a choice of c_i's, for any distinct $i_1, i_2 \in [n]$, $deg(p_{i_1}^{c_{i_1}})$ is well separated from $deg(p_{i_2}^{c_{i_2}})$. With a careful case analysis, we use this to show that some monomial (of degree at least 1) in $g(x)$ survives. We provide the details in Section 8.2.

Non-malleable extractors for skew affine sources against polynomial tampering. In the previous paragraph we sketched how to construct a non-malleable extractor against polynomial tampering assuming access to a uniform source on $\mathbb{F}_q^n$. In Section 8.2, we actually show that the non-malleable extractor works for any affine source which is non-constant on every coordinate. We call such source a *skew affine source*. In other words, our non-malleable extractor is resilient to affine leakage which does not reveal any single coordinate in the source. We will see the application of this property in non-malleable secret sharing.

To prove this stronger property of the non-malleable extractor, recall that in the previous section we defined a polynomial $g_\beta(x) = h(x) + \beta h(P(x))$, and its restriction to the line $L_{a,b}$, denoted by $g_{\beta,a,b}(t)$. We then sketched a proof that $g_{\beta,a,b}$ is non-constant if $g_\beta(b) \neq 0$, which happens with high probability over b. In Section 8.2, we actually show the following stronger result: $\forall i, b_i \neq 0$ is a sufficient condition for $g_{\beta,a,b}$ to be non-constant. In fact, it is also a necessary condition. If there exists i such that $b_i = 0$, the adversary can set $p_j(x) = x_j$ for every $j \neq i$ and $p_i(x) = c$ for a constant $c \neq a_i$. One can verify that $g_{-1,a,b}$ is a constant in this case.

The proof idea is that a similar case analysis as sketched in the previous section also works for $g_{\beta,a,b}$ if $b_i \neq 0$ for every i. We then show that every skew affine source is a convex combination of line source $L_{a,b}$ where $b_i \neq 0$ for every i (Claim 18) to finish the proof.

Non-malleable codes against polynomial tampering. We now turn to cryptographic applications of our non-malleable extractors. To build a non-malleable code against polynomial tampering, we use the connection between non-malleable code and non-malleable extractor established in [11]. To apply the reduction in [11], we need an efficient algorithm which samples almost uniformly from a pre-image of our non-malleable extractor on any output.

Recall that our non-malleable extractor is of the form $\mathrm{NMExt}(x) = \sigma(h(x))$, where σ is modulo 2^m and h is a bounded-degree polynomial. Inverting σ is easy, and there exists an algorithm by Cheraghchi and Shokrollahi [102] which almost-uniformly samples a pre-image of bounded-degree polynomial (over any large enough prime field). An initial attempt to sample from $\mathrm{NMExt}^{-1}(z)$ would be first sample $y \in \sigma^{-1}(z)$ and then sample from $h^{-1}(y)$. However this does not work since $h^{-1}(y)$ might have different size for different $y \in \mathbb{F}_q$. So we need to sample $y \in \sigma^{-1}(z)$ with probability proportional to $|h^{-1}(y)|$. A possible way to perform such weighted sampling from $\sigma^{-1}(z)$ is to do a rejection sampling which samples $y \in \sigma^{-1}(z)$ uniformly in each round and accept with probability proportional to $|h^{-1}(y)|$. However, we need to (approximately) count $|h^{-1}(y)|$ in this approach, which is difficult in general.

Chattopadhyay and Zuckerman [37] handled a similar sampling task while constructing efficient non-malleable codes in the split-state model, with the crucial difference being that they were dealing with polynomials on a constant number of variables. In [37], they adopted a similar sampling strategy as the one sketched above, and they count $|h^{-1}(y)|$ with an algorithm from [103], which has running time doubly exponential in the number of variables (which, in their case, still takes constant time).

To get around this difficulty, we observe that the algorithm in [102] is actually a rejection sampling which has accepting probability proportional to $|h^{-1}(y)|$ in each round. Therefore, we can

embed an uniform sampling of y in each round of [102] and bypass the computation of $|h^{-1}(y)|$. We provide the details of our sampling algorithm in Section 8.3.

8.2 Non-malleable extractors against polynomials

We present the proof of Theorem 29 in this section. On a high level, our idea is to express X as a convex combination of sources on lines in $\mathbb{F}_q^n$, and design a non-malleable extractor for such line sources. We note that Gabizon and Raz [53] adopted such an approach for constructing affine extractors over large fields. First we show that a skew affine source is a convex combination of skew line source.

Lemma 18. *Let q be a prime, $n < q$ be a integer and $X \in \mathbb{F}_q^n$ be a skew affine source of dimension k. Then there exists a distribution $A \in \mathbb{F}_q^n$ and a vector $b \in (\mathbb{F}_q \backslash \{0\})^n$ such that $X \equiv A + Tb$, where T is uniform over $\mathbb{F}_q$. In other words,*

$$X = \sum_{a \in F_q^n} \Pr[A = a] \cdot L_{a,b},$$

where $L_{a,b}$ is the uniform distribution over the line $\{a + tb : t \in \mathbb{F}_q\}$.

Proof. Suppose X is uniform over the affine subspace $W + z$ where W is a linear subspace of $\mathbb{F}_q^n$ and $z \in \mathbb{F}_q^n$ is a fixed vector. Our goal is to find a vector $b \in W$ s.t. $b_i \neq 0$ for every $i \in [n]$. Given such b we can set $A \equiv X$, and the lemma holds because $tb \in W$ for every $t \in \mathbb{F}_q$, and $X + w \equiv X$ for every $w \in W$.

Fix a basis $\{w_1, \ldots, w_k\}$ of the linear subspace W. For every $i \in [k]$, define $S_i = \{j \in [n] : (w_i)_j \neq 0\}$ (i.e. the indices of the non-zero coordinates of w_i) and $\overline{S}_i = \bigcup_{j=1}^{i} S_j$. Note that $\overline{S}_k = [n]$ because $W + z$ does not have any constant coordinate. We will prove by induction that for every $i \in [k]$ there exists $v_i \in \mathrm{span}(w_1, \ldots, w_i)$ s.t. $(v_i)_j \neq 0$ for every $j \in \overline{S}_i$. Assume that there exists v_{i-1} which satisfies the induction hypothesis. (Note that $v_0 = 0$.) Consider the set of q distinct vectors $L_i = \{v_{i-1} + tw_i : t \in \mathbb{F}_q\} \subseteq \mathrm{span}(w_1, \ldots, w_i)$. Observe that for every $j \in S_i$, there exists at most one vector $u_j \in L_i$ satisfying that $(u_j)_j = 0$. Since $n < q$, there must exist $u^* \in L_i$ s.t. $(u^*)_j \neq 0$

for every $j \in S_i$. Moreover, for every $j \in \overline{S_i} \backslash S_i \subseteq \overline{S_{i-1}}$, $(u^*)_j = (v_{i-1})_j \neq 0$. Therefore $(u^*)_j \neq 0$ for every $j \in \overline{S}_i$. By mathematical induction, our claim is true for every $i \in [k]$. Finally observe that v_k is a valid choice of b because $\overline{S}_k = [n]$ and $\text{span}(w_1, \ldots, w_k) = W$. □

Next we present the extractor construction and prove correctness. Let B be the smallest integer greater than 3 such that $\gcd(B, q-1) = 1$. Note that B must be a prime. We can deduce an upper bound on B as follows. Define the primorial function $\nu(\ell)$ as the product of the first ℓ primes. It is known that $\nu(\ell) = e^{(1+o(1))\ell \log(\ell)}$ [104]. Further, it is known that the ℓ'th smallest prime number is at most $O(\ell \log(\ell))$ [105, 106]. Hence, it must be that $B \leq \mu \log q$, for some large enough constant μ. We can thus find such a B efficiently.

For $i \in [2n]$, define $c_i = B(4dn+1) + Bi$. Define the function $h : \mathbb{F}_q^n \to \mathbb{F}_q$ as

$$h(x_1, \ldots, x_n) = \sum_{i=1}^{n} (x_i^{c_{2i-1}} + x_i^{c_{2i}}).$$

Let $\sigma : \mathbb{F}_q \to \{0,1\}^m$ be the mapping from Lemma 9. We now define the non-malleable extractor:

$$\text{NMExt}(x) = \sigma(h(x)).$$

For any $a \in \mathbb{F}_q^n$ and $b \in \mathbb{F}_q^n \setminus \{0^n\}$, define the line $L_{a,b} = \{a + tb : t \in \mathbb{F}_q\}$. We overload notation, and also use $L_{a,b}$ to denote the flat source on this line. We will show that NMExt is a non-malleable extractor against $\text{Poly}_{n,q,d}$ for every skew line source. Theorem 29 then follows using Claim 18.

Lemma 19. *Let $a \in \mathbb{F}_q^n, b \in (\mathbb{F}_q \backslash \{0\})^n$. For every tampering function $P \in \text{Poly}_{n,q,d}$ which is not identity on $L_{a,b}$,*[3]

$$(\text{NMExt}(L_{a,b}), \text{NMExt}(P(L_{a,b}))) \approx_\epsilon (\text{U}_m, \text{NMExt}(P(L_{a,b}))),$$

where $\epsilon = O\left(\frac{2^m d^2 n \log q}{\sqrt{q}}\right)$

[3]That is, there exists $x \in L_{a,b}$ s.t. $P(x) \neq x$.

The following bound is the key ingredient. Indeed, Claim 19 then follows using Lemma 9.

Lemma 20. *Let χ, ϕ be additive characters of $\mathbb{F}_q$ such that χ is non-trivial. Then,*

$$\left|\mathbb{E}[\chi(h(L_{a,b}))\phi(h(P(L_{a,b})))]\right| \leq O((d^2 n \log q)/\sqrt{q}).$$

Let $\chi(y) = e^{2\pi\alpha y/q}$ and $\phi(y) = e^{2\pi\alpha' y/q}$. Since χ is non-trivial, we know that $\alpha \neq 0$. Let $\beta = \alpha'/\alpha$. Define the polynomial

$$g_\beta(x) = h(x) + \beta h(P(x)).$$

We note that

$$|\mathbb{E}[\chi(h(X))\phi(h(P(X)))]| \leq \left|\mathbb{E}\left[e\left(\frac{\alpha g_\beta(X)}{q}\right)\right]\right|.$$

Let $g_{\beta,a,b}(t)$ be the univariate polynomial obtained by restricting $g(x)$ to the line $L_{a,b}$. The following two claims directly yields Claim 20.

Lemma 21. *Suppose for some $a, b \in \mathbb{F}_q^n$, $g_{\beta,a,b}$ is a non-constant polynomial. Then,*

$$\left|\mathbb{E}_{t\sim\mathbb{F}_q}\left[e\left(\frac{\alpha \cdot g_{\beta,a,b}(t)}{q}\right)\right]\right| \leq O((d^2 n \log q)/\sqrt{q}).$$

Lemma 22. *For every $a \in \mathbb{F}_q^n$, $b \in (\mathbb{F}_q \backslash \{0\})^n$, $g_{\beta,a,b}$ is a constant polynomial only if P is identity on $L_{a,b}$.*

Claim 21 is indeed simple to obtain using the Weil bound.

Proof of Claim 21. Follows directly from Theorem 11 using the fact that $deg(g_{\beta,a,b}(t)) \leq O(d^2 n \log q)$. □

Now we prove Claim 22.

Proof of Claim 22. For every $i \in [n]$, define the polynomial $q_i(t) = p_i(a + tb)$. Since $a + tb$ is an

affine function, $deg(q_i) \leq deg(p_i) \leq d$. Let $d_i = deg(q_i)$. For every $i \in [n]$, define

$$w_i(t) = (a_i + tb_i)^{c_{2i-1}} + (a_i + tb_i)^{c_{2i}} + \beta q_i(t)^{c_{2i-1}} + \beta q_i(t)^{c_{2i}}.$$

Recall that

$$g_{\beta,a,b}(t) = \sum_i w_i(t).$$

First we prove that $deg(w_i) \in \{0, c_{2i}d_i, c_{2i}, c_{2i-1}, c_{2i} - 1\}$. Moreover, $deg(w_i) = 0$ if and only if $\beta = -1$ and $q_i(t) = a_i + tb_i$. (In other word, w_i is constant if and only if $\beta = -1$ and $p_i(x) = x_i$ for every $x \in L_{a,b}$.) To prove this statement, first we consider the case $deg(q_i) \geq 2$. Suppose that the leading coefficient in q_i is $s_i \neq 0$. If $\beta \neq 0$, the coefficient of $t^{c_{2i}d_i}$ in w_i is $\beta s_i^{c_{2i}} \neq 0$. Therefore $deg(w_i) = c_{2i}d_i$. If $\beta = 0$, the coefficient of $t^{c_{2i}}$ in w_i is $b_i^{c_{2i}} \neq 0$. Therefore $deg(w_i) = c_{2i}$. Next consider the case $deg(q_i) = 0$. With an argument similar to the case $\beta = 0$, we also have $deg(w_i) = c_{2i}$. Finally consider the case $deg(q_i) = 1$. Suppose $q_i(t) = r_i + ts_i$. Observe that the coefficient of $t^{c_{2i}}$ in w_i is $b_i^{c_{2i}} + \beta s_i^{c_{2i}}$ and the coefficient of $t^{c_{2i}-1}$ in w_i is $c_{2i}(a_i b_i^{c_{2i}-1} + \beta r_i s_i^{c_{2i}-1})$. In this case either $deg(w_i) \in \{c_{2i}, c_{2i} - 1\}$ or

$$b_i^{c_{2i}} = -\beta s_i^{c_{2i}} \text{ and } a_i b_i^{c_{2i}-1} = -\beta r_i s_i^{c_{2i}-1}.$$

The equations hold only when there exists $k \in \mathbb{F}_q$ s.t.

$$r_i = ka_i, s_i = kb_i \text{ and } k^{c_{2i}} = -\beta^{-1}.$$

If such k exists, we can write $w_i(t) = (1 - k^{-B}(a_i + tb_i)^{c_{2i-1}}$. If $\beta = -1$, we have $k = 1$, $w_i(t) = 0$ and $q_i(t) = a_i + tb_i$. If $\beta \neq -1$, then $k \neq 1$, which implies $(1 - k^{-B}) \neq 0$ because $\gcd(B, q-1) = 1$. Therefore w_i contains a monomial of degree c_{2i-1} with coefficient $(1 - k^{-B})b_i^{c_{2i-1}} \neq 0$, and hence $deg(w_i) = c_{2i-1}$.

Now we show that $g_{\beta,a,b}(t)$ is a constant polynomial only if $\beta = -1$ and $q_i(t) = a_i + tb_i$ for every $i \in [n]$. Consider the set of index $I = \{i \in [n] : deg(w_i) > 0\}$. Then for every $i \in I$,

$deg(w_i) \in \{d_i c_{2i}, c_{2i}, c_{2i-1}, c_{2i} - 1\}$ if $d_i > 0$, or $deg(w_i) \in \{c_{2i}, c_{2i-1}, c_{2i} - 1\}$ if $d_i = 0$. By Lemma 10, for every pair $i, j \in I$ s.t. $i \neq j$, we have $deg(w_i) \neq deg(w_j)$. Therefore $deg(g_{\beta,a,b}) > 0$ if I is non-empty. If $g_{\beta,a,b}$ is a constant polynomial, it must be the case that $deg(w_i) = 0$ for every $i \in [n]$. This only happens when $\beta = -1$ and $q_i(t) = a_i + tb_i$ for every $i \in [n]$, i.e. $\beta = -1$ and $P(x) = x$ for every $x \in L_{a,b}$. Claim 22 then follows directly.

□

Finally we prove Theorem 29 formally.

Theorem 30 (Theorem 29, restated)**.** *There exists a constant $C > 0$ such that for every integers n, m, d, any $\epsilon > 0$, any prime q such that $q > Cn^2d^4m2^{2m} \cdot \log(nd/\epsilon)$, any skew affine source $X \in \mathbb{F}_q^n$ of dimension ≥ 1 and any tampering function $f \in \mathrm{Poly}_{n,q,d}$, there exists a distribution D_f on $\{0,1\}^m \cup \{\mathtt{same}\}$ that is independent of X, such that*

$$|(\mathrm{NMExt}(X), \mathrm{NMExt}(f(X))) - (\mathrm{U}_m, \mathrm{copy}(D_f, \mathrm{U}_m))| \leq \epsilon.$$

Proof. By Claim 18, there exists a distribution A on $\mathbb{F}_q^n$ and vector b such that $X = \sum_a \Pr[A = a] \cdot L_{a,b}$. Define $I = \{a \in \mathbb{F}_q^n : f \text{ is identity on } L_{a,b}\}$. For every $a \in I$, define $(D_f)_a = \mathtt{same}$. For every $a \notin I$ define $(D_f)_a = \mathrm{NMExt}(f(L_{a,b}))$. Then we claim that $D_f = \sum_a \Pr[A = a] \cdot (D_f)_a$ satisfies the requirement:

$$\begin{aligned}
&|\mathrm{NMExt}(X), \mathrm{NMExt}(f(X)) - \mathrm{U}_m, \mathrm{copy}(D_f, \mathrm{U}_m)| \\
&\leq \sum_a \Pr[A = a] \cdot |\mathrm{NMExt}(L_{a,b}), \mathrm{NMExt}(f(L_{a,b})) - \mathrm{U}_m, \mathrm{copy}((D_f)_a, \mathrm{U}_m)| \\
&= \sum_{a \in I} \Pr[A = a] \cdot |\mathrm{NMExt}(L_{a,b}), \mathrm{NMExt}(L_{a,b}) - \mathrm{U}_m, \mathrm{U}_m| \\
&+ \sum_{a \notin I} \Pr[A = a] \cdot |\mathrm{NMExt}(L_{a,b}), \mathrm{NMExt}(f(L_{a,b})) - \mathrm{U}_m, \mathrm{NMExt}(f(L_{a,b}))| \\
&\leq \sum_{a \in I} \Pr[A = a] \cdot \epsilon + \sum_{a \notin I} \Pr[A = a] \cdot \epsilon \\
&= \epsilon
\end{aligned}$$

The first inequality is by the convexity of statistical distance, and the second inequality is by Claim 19. □

8.3 Efficient sampling

Recall that to construct efficient non-malleable codes using the connection established in [11], we need to efficiently sample from the pre-image of any given output of the non-malleable extractor constructed in the previous section. (We discuss this connection in Section 2.3.) In this section we show how to construct such a sampler for the non-malleable extractor constructed in Theorem 29. Note that Theorem 8 uses the same non-malleable extractors.

Theorem 31. *Let* $\mathrm{NMExt} : \mathbb{F}_q^n \to \{0,1\}^m$ *be the non-malleable extractor against* $\mathcal{F}_{n,q,d}$ *tampering in Theorem 29. Then there exists a randomized algorithm* $\overline{\mathrm{nmExt}^{-1}}$ *such that for every* $z \in \{0,1\}^m$ *the distribution of* $\overline{\mathrm{nmExt}^{-1}}(z)$ *is* ϵ*-close to uniform distribution on* $\mathrm{NMExt}^{-1}(z)$*. The running time of* $\overline{\mathrm{nmExt}^{-1}}$ *is bounded by* $\mathrm{poly}(n, d, \log q, \log(1/\epsilon))$.

Our starting point to prove Theorem 31 is a sampling algorithm from [37], which has running time $O(d^{n^{O(n)}}(\log q)^{O(1)})$ and error $O(d^{O(n^n)}/\sqrt{q})$. We will show how to modify this algorithm and get an improved running time of $\mathrm{poly}(n, d, \log q, \log(1/\epsilon))$ for arbitrarily small error ϵ.

Let NMExt be the non-malleable extractor from Theorem 29. Recall that $\mathrm{NMExt} = \sigma \circ h$ where $\sigma : \mathbb{F}_q \to \{0,1\}^m$ is defined as $\sigma(x) = x \pmod{2^m}$ and $h : \mathbb{F}_q^n \to \mathbb{F}_q$ is a multivariate polynomial of degree d over $\mathbb{F}_q$. Given $z \in \{0,1\}^m$, the pre-image of z under NMExt is

$$\mathrm{NMExt}^{-1}(z) = \bigcup_{y \in \sigma^{-1}(z)} h^{-1}(y),$$

and our goal is to sample from $\mathrm{NMExt}^{-1}(z)$ almost uniformly. The sampling algorithm in [37] is based on the following rejection sampling strategy.

Let $M \geq \max_y |h^{-1}(y)|$.

1. Sample $y \in \sigma^{-1}(z)$ uniformly at random.

2. Compute $|h^{-1}(y)|$ (approximately), and accept y with probability $|h^{-1}(y)|/M$. If y is rejected, go back to step 1.

3. Output an (almost) uniform sample from $h^{-1}(y)$.

In [37], the second step is achieved by an algorithm from [103] that has running time $O(d^{n^{O(n)}}(k \log q)^{O(1)})$.

The third step is based on the following algorithm in [102].

Lemma 23 ([102])**.** *Let q be a sufficiently large prime, $f \in \mathbb{F}_q[x_1, \ldots, x_n]$ be polynomials of total degree bounded by d, and each polynomial has at most ℓ monomials. Let $S \subseteq \mathbb{F}_q^n$ be the set of common zeroes of f. There exists a randomized algorithm which takes f as input (as a list of monomials) and outputs a random value $X \in \mathbb{F}_q^n$ such that the distribution of X is $O(d^{O(1)}/q)$-close to uniform distribution on S. The worst-case running time of this algorithm is* $\mathrm{poly}(\log q, d, n, \ell)$.

Thus the bottleneck in achieving a polynomial time sampling algorithm is Step (2) which takes time that is doubly exponential in n. We get around this difficulty as follows: first note that the rejection sampling in Step (2) is to ensure that the subset $h^{-1}(y)$ is selected with probability proportional to $|h^{-1}(y)|$. Our crucial observation is that the algorithm in Claim 23 is actually a rejection sampling which accepts an output with probability proportional to $|h^{-1}(y)|$ in each round. Therefore we can actually combine the rejection sampling in Step 2 and 3, and bypass the computation of $|h^{-1}(y)|$.

First we explain the relation between the algorithm in Claim 23 and rejection sampling. A naive way to sample from the variety $h^{-1}(y)$ is to repeatedly sample a point $x \in \mathbb{F}_q^n$ and verify if $h(x) = y$. However, the success probability of the naive rejection sampling is only $|h^{-1}(y)|/q^n$, which is too small. The idea in [102] is that the space $\mathbb{F}_q^n$ can be split into lines, and the variety S is split into many "slices" by these lines. The naive rejection sampling is equivalent to first sampling a line and then sampling a point from this line. Since each line has q points, the probability of a certain point in the variety being chosen is still $1/q^{n-1} \cdot 1/q$. However, if we choose a *good direction* to split the space, each slice of the variety only has at most d points where $d \ll q$, and these points can be enumerated efficiently. Therefore instead of sampling every point in this

subspace with equal probability we can sample only from the *slice of variety* instead. This allows us to increase the accepting probability in each round to $|h^{-1}(y)|/dq^{n-1}$, which is high enough and still proportional to $|h^{-1}(y)|$. With the ideas above we get the following lemma.

Lemma 24. *Let $h : \mathbb{F}_q^n \to \mathbb{F}_q$ be a n-variate polynomial of degree $d < q/2$ with ℓ monomials, and $\sigma : \mathbb{F}_q \to \{0,1\}^m$ be any function. Suppose we have access to an oracle Samp_σ which takes input z and outputs a sample from $\sigma^{-1}(z)$ uniformly at random. Then for every $\epsilon > 0$, there exists a randomized algorithm A such that for every $z \in \{0,1\}^m$, the algorithm either outputs a uniformly random sample from $(\sigma \circ h)^{-1}(z)$ or output $\perp$. The probability that the algorithm outputs $\perp$ is at most ϵ.*

Moreover, the expected running time of A on z is $T \cdot \mathrm{poly}(\log q, n, d, \ell)$ plus T oracle calls to Samp_σ, where

$$T = O\left(\frac{q^{n-1} \cdot d \cdot \left|\sigma^{-1}(z)\right|}{\left|(\sigma \circ h)^{-1}(z)\right|} \log(1/\epsilon)\right).$$

Before we formally prove Theorem 24, first we show how to prove Theorem 31 based on Theorem 24. The following corollary shows that the algorithm in Theorem 24 is efficient when $\sigma \circ h$ is an "extractor for uniform distribution" and σ does not concentrate on certain output.

Corollary 9. *Suppose that $\sigma(h(\mathrm{U}_{\mathbb{F}_q^n})) \approx_{1/2^{m+1}} \mathrm{U}_m$, and $|\sigma^{-1}(z)| \le Cq/2^m$ for every z. Then the running time of the algorithm in Theorem 24 is $C \log(1/\epsilon)\,\mathrm{poly}(n, \ell, \log q, d)$.*

Proof. The number of rounds of rejection sampling in the algorithm from Theorem 24 is $T = O\left(\frac{q^{n-1} \cdot d \cdot \left|\sigma^{-1}(z)\right|}{\left|(\sigma \circ h)^{-1}(z)\right|} \log(1/\epsilon)\right)$.

Observe that

$$\left|(\sigma \circ h)^{-1}(z)\right| = q^n \cdot \Pr[\sigma(h(\mathrm{U}_{\mathbb{F}_q^n})) = z] \ge q^n \cdot (1/2^m - 1/2^{m+1}) = q^n/2^{m+1}.$$

Plugging this in, and the upper on $\sigma^{-1}(z)$, we have $T = O(d \log(1/\epsilon))$. The corollary now follows directly from Theorem 24. □

Proof of Theorem 31. To prove Theorem 31 we only need to show that our non-malleable extrac-

tor satisfies the condition in Theorem 9. The fact that $\sigma(h(\mathrm{U}_{\mathbb{F}_q^n}))$ is close to U_m follows from Theorem 29, and the second condition is also true because $\sigma(x) = x \mod 2^m$, which satisfies $|\sigma^{-1}(z)| \leq \lceil q/2^m \rceil$ for every $z \in \{0,1\}^m$. □

We now prove Theorem 24. First we need the following lemma which is analogous to Proposition 4.3 in [102]. Note that we slightly tweak the lemma to make the sampling algorithm able to handle arbitrarily small error. The lemma says a random direction is a good direction to split the space with high probability.

Lemma 25. *Let $h : \mathbb{F}_q^n \to \mathbb{F}_q$ be a n-variate polynomial of degree at most d, and let $b = (b_1, \ldots, b_n)$ be uniformly random samples from $\mathbb{F}_q$. Then with probability at least $1 - d/q$, $h_{a,b}(t) = h(a_1 + b_1 t, \ldots, a_n + b_n t)$ is a non-constant polynomial of t for every $a = (a_1, \ldots, a_n) \in \mathbb{F}_q^n$.*

Proof. Let g be the highest-degree homogeneous part of h. Then observe that $h_{a,b}(t)$ has degree at most d, and its coefficient of t^d equals to $g(b_1, \ldots, b_n)$. By Lemma 11, the probability that $g(b_1, \ldots, b_n)$ is non-zero is at least $1 - d/q$. Therefore with probability $1 - d/q$ over b, $h_{a,b}(t)$ has degree exactly d for every $a \in \mathbb{F}_q^n$. □

Proof of Theorem 24. In algorithm A, first we repeatedly sample $b \in \mathbb{F}_q^n$ uniformly at random until we find b which satisfies the condition in Claim 25. If we fail to find such b in $\log(1/\epsilon) + 1$ rounds, abort and output $\perp$. Then repeat the following steps for at most T rounds:
Sample $y \in \sigma^{-1}(z)$ with oracle Samp_σ, and sample $a = (a_1, \ldots, a_n)$ uniformly at random. Compute the restriction of $h(x) = y$ on the line $L_{a,b} = \{(a_1 + b_1 t, \ldots, a_n + b_n t) : b \in \mathbb{F}_q\}$, i.e. $h_{a,b}(t) = y$ where $h_{a,b}(t) = h(a_1 + b_1 t, \ldots, a_n + b_n t)$. Note that $h_{a,b}$ is a non-constant polynomial of degree at most d. Then we run Berlekamp-Rabin algorithm [107] to enumerate all the roots of $h_{a,b}$ in $\mathbb{F}_q$, denoted by $t_1, \ldots, t_k$ where $k \leq d$. Now pick a number $i \in [d]$ uniformly at random. If $i \leq k$, the algorithm succeeds, and we will return $(a_1 + b_1 t_i, \ldots, a_n + b_n t_i)$. Otherwise sample y and a again and repeat. If no value is returned after all T rounds, return $\perp$.

To prove the correctness of A, first we compute the distribution $A(z)$ conditioned on that the algorithm succeeds. Observe that $A(z)$ never returns an element which is not in $(\sigma \circ h)^{-1}(z)$. Moreover, for every $v \in (\sigma \circ h)^{-1}(z)$, in each round the probability that $A(z)$ outputs v is

$$\frac{1}{|\sigma^{-1}(z)|} \cdot \frac{1}{q^{n-1}} \cdot \frac{1}{d}.$$

The first factor is the probability that $y = h(v)$, the second factor is the probability that $L_{a,b} \ni v$, and the third factor is the probability that v is chosen from the list of roots of $h_{a,b}$. Since this formula does not depend on v, we can conclude that $A(z)$ is a uniform distribution on $(\sigma \circ h)^{-1}(z)$, conditioned on $A(z) \neq \perp$.

Now we compute the probability that A fails. Assuming $q \geq 2d$, the probability that we fail to find a b satisfying the condition in Claim 25 in $log(1/\epsilon)+1$ rounds is at most $(d/q)^{\log(1/\epsilon)+1} \leq \epsilon/2$. If we find such b successfully, observe that A successfully returns a sample with probability

$$p = \frac{|(\sigma \circ h)^{-1}(z)|}{|\sigma^{-1}(z)| \cdot q^{n-1} \cdot d}$$

in one round. Now define

$$T = \frac{C \log(1/\epsilon)}{p},$$

for a large enough constant C. Then the probability that A does not output any element after T rounds is at most $(1-p)^T < \epsilon/2$. Therefore $\Pr_A[A(z) = \perp] \leq \epsilon$.

Finally we analyze the running time of A. Finding a vector b which satisfies Claim 25 (or abort and output $\perp$) takes at most $\log(1/\epsilon)\,\mathrm{poly}(n, \ell, \log q, d)$ steps. After finding b, we run at most T rounds of rejection sampling, where in each round we first make an oracle call to Samp_σ, sample a and compute the polynomial $h_{a,b}$ which takes $\mathrm{poly}(n, \ell, \log q, d)$ steps, and run Berlekamp-Rabin which takes expected $\mathrm{poly}(n, \ell, \log q, d)$ steps. Therefore the total expected running time is as claimed. □

Remark 6. While we only show the expected running time in Theorem 24, it is possible to bound

the worst-case running time by introducing a small error to the output distribution as follows. In each of the T rounds, we are running Berlekamp-Rabin algorithm to factorize a polynomial of degree at most d. Recall that in Berlekamp-Rabin algorithm, we are repeatedly trying to factorize a polynomial into two non-trivial factors. Moreover, each attempt of factoring succeeds with probability at least $1/2$. To factorize T polynomials of degree d, we need at most Td successful attempts. Note that the probability that there are less than Td success in the first $7Td$ attempts are at most $(1/2)^T < \epsilon$. Therefore, we can force the algorithm to terminate and output $\perp$ after $6Td$ unsuccessful attempts of Berlekamp-Rabin. This ensures that the worst-case running time is still $T \cdot \mathrm{poly}(n, \ell, \log q, d)$. Besides, since the *time-out* event happens with probability at most ϵ, the output distribution is still ϵ-close to uniform distribution on $(\sigma \circ h)^{-1}(z)$.

Chapter 9: Small Circuit Tampering

Perhaps the most natural class of tampering functions, is tampering by polynomial size circuits. Unfortunately, a simple argument shows that any non-malleable code resilient to polynomial size circuit tampering cannot be evaluated in polynomial time (see Section 3.3 for details). The obvious next best thing would be a non-malleable code computable in polynomial time that is resilient to *bounded* polynomial size circuit tampering, tampering by circuits of size at most n^c where c is a constant fixed a priori.

We know efficient non-malleable codes exist for tampering by bounded polynomial size circuits in the CRS model [11, 19] as well as under cryptographic assumptions [20, 21]. However, the latter require assumptions that we currently only know how to instantiate with random oracles and additionally fall short of achieving statistical security guarantees.

In this work, we show that seed-extending PRGs for nondeterministic circuits of size n^c, where c is a constant, suffice to construct explicit[1] codes for bounded polynomial size circuits. Before explaining what this object is, we remark that they are implied by an assumption introduced in the context of derandomizing AM: there is a language that can be computed in exponential deterministic time that requires exponential size nondeterministic circuits.

Assumption 1 (E is hard for exponential size nondeterministic circuits)**.** There is a language $L \in \mathsf{E} = \mathsf{DTIME}(2^{O(n)})$ and a constant γ such that for sufficiently large n nondeterministic circuits of size $2^{\gamma n}$ fail to compute the characteristic function of L on inputs of length n.

Informally, this says that non-uniformity and non-determinism do not always imply significant speed-ups.

[1] We say a code, or family of codes, is explicit if encoding and decoding can be performed in uniform polynomial time.

As mentioned, this assumption and less robust variants been in the complexity and derandomization literature for some time. [108, 109, 110, 111, 112, 113, 114, 115, 116, 117, 118] While this assumption may not be as well understood as we might like, we would like to note that is a *worst-case* assumption on a *class* of languages (that admits complete problems). We would also like to highlight that it is orthogonal to standard cryptographic assumptions such as one-way functions for all that we know, so it may hold even if cryptography does not exist (in some sense $\mathsf{P} = \mathsf{NP}$ only bolsters confidence that E is hard for exponential size nondeterministic circuits).

We can now state our main theorem of this chapter.

Theorem 32. *If E is hard for exponential size nondeterministic circuits, then for every constant c, there is an explicit n^{-c}-non-malleable code resilient to tampering by n^c-size circuits.*

On the feasibility of explicit codes from minimal assumptions. It is known that explicit non-malleable codes for circuits of size $O(n^c)$ imply explicit languages that are hard on average for circuits of size $O(n^c)$. For this reason, it is unlikely to construct explicit codes for such a tampering class. Yet, one might still hope to construct codes by simply assuming minimal circuit lower bounds (simply assuming there exists a language computable in time n^d, for some $d > c$, that is hard on average for $O(n^c)$-size circuits). Unfortunately, in Chapter 10 we will see a barrier to proving such a theorem. In particular, the results their rule out non-malleable code constructions where the *security proof* reduces a successful tampering attack to (breaking) such an assumption makes black box use of the tampering attack. While, of course, non-black-box methods may be possible, one could also take this result to mean that stronger assumptions are needed (given our current toolkit).

In more detail, our present result skirts this lower because the techniques of Chapter 10 critically require that the reduction is no more complex than the tampering class. Here, our tampering class consists of small *deterministic* circuits, but our ingredients require hardness for small *nondeterministic* circuits.

9.1 Technical Overview

We begin by introducing seed-extending PRGs for non-deterministic circuits. We then follow our technical presentation and describe how to construct seedless non-malleable "extractors" for bounded polynomial size circuit tampering and uniform sources.[2] While this doesn't yield a good randomness extractor, it does effectively capture the core non-malleability of our construction. We conclude by sketching how we leverage this concept to a non-malleable code.

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

Seed-extending pseudorandom generators. A *pseudorandom generator (PRG) for nondeterministic circuits of size $s(n)$*, $\mathsf{prg} : \{0,1\}^\ell \to \{0,1\}^n$, allows one to extend a short random seed into a long string that is indistinguishable from random to nondeterministic circuits of size $s(n)$. More precisely, for every nondeterministic circuit, C, of size at most $s(n)$,

$$|\Pr[C(\mathsf{prg}(\mathcal{U}_\ell)) = 1] - \Pr[C(\mathcal{U}_n) = 1]| \leq \frac{1}{s(n)},$$

where $\mathcal{U}_m$ denotes a random variable uniformly distributed over $\{0,1\}^m$.

We are interested in such PRGs that additionally remain secure when given the seed. In particular, we will use "seed-extending" PRGs.[3] A PRG, $\mathsf{prg} : \{0,1\}^\ell \to \{0,1\}^n$, is said to be seed-extending if $\mathsf{prg}(s) = (s, \mathsf{prg}'(s))$ (where prg' is the function corresponding to the $n - \ell$ bit suffix). This particular name was introduced by Kinne et al. in the context of derandomizing randomized algorithms on random inputs. [119, 120] They showed that PRG constructions based

[2] In fact, the construction sketched here can handle sources of the form S, X where S is a truly random seed and X has high min-entropy. This corresponds to a seeded extractor where the seed and source can be jointly tampered. However, because our extractor still requires X to have very high entropy, we do not formalize this.

[3] Elsewhere, such objects have been called *strong* PRGs (borrowing from the Extractor literature) or *systematic* PRGs (borrowing from the coding theory literature).

on Nisan and Wigderson's seminal construction [119] can be made seed-extending. Consequently, many constructions of PRGs for nondeterministic circuits can be made seed extending.

Theorem 33 ([119, 121, 113, 115, 116, 65])**.** *If* E *is hard for exponential size nondeterministic circuits, then for every constant* $c > 1$ *there exists a constant* $\alpha > 1$ *such that for every sufficiently large* n*, and every* r *such that* $\alpha \log n \leq \ell \leq n$ *there is a PRG,* $\mathsf{prg} : \{0,1\}^{\ell} \to \{0,1\}^{n}$*, for nondeterministic circuits of size* n^{c}*.*

We remark that both our requirements that prg is (a) secure against non-deterministic circuits of size n^c and (b) seed-extending each necessitate that prg cannot be evaluated in time less than n^c. As such, no cryptographic PRG meets these criteria.

Our proofs follow the approach of Applebaum et al. [65] and reduce to PRG security by constructing constant round private coin interactive proofs with a small non-uniform verifier sufficient gap between completeness (probability of accepting pseudorandom inputs) and soundness (probability of accepting random inputs). By the classical results [63, 64], constant-round interactive proofs can be compiled into AM protocols[4], without blowing up the size of the verifier (Arthur) too much. From there, standard techniques ($\mathsf{AM}/\mathrm{poly} \subseteq \mathsf{NP}/\mathrm{poly}$) yield a small nondeterministic circuit with the same promise.

Seedless Non-Malleable Extractor. Our construction of a (relaxed) seedless non-malleable extractor for uniform sources and n^c-size circuit tampering is exceedingly simple. Recall that a relaxed seedless non-malleable extractor (Def. 10) is a deterministic function NMExt such that for any n^c size circuit, C without fixed points we have

$$\mathrm{NMExt}(s, x), \mathrm{NMExt}(C(s, x)) \approx \mathcal{U}, \mathrm{NMExt}(C(s, x)).$$

Let 2NMExt be a two-source non-malleable extractor (NME). Our seedless non-malleable ex-

[4] AM protocols consist of a uniformly random message, followed by a response from Merlin, at which point Arthur must, deterministically, output based on the transcript.

tractor, NMExt, is defined as

$$\text{NMExt} : (s, x) \mapsto (\mathsf{prg}(s), x)$$

where prg is a *seed-extending PRG for nondeterministic circuits of size n^d* for some constant $d > c$.

At a very high level, our approach is to fool the non-malleable extractor experiment with our PRG. The fact that the pseudorandom distribution can be compressed to small seed is going to allow us to translate efficient circuit tampering. In particular, by conditioning on the *tampered* seed our small circuit tampering is effectively split-state. Because the seed is short, conditioning on the seed won't impact the entropy of our sources too much in the truly random case. However, the argument is a bit nuanced so we go through it in more detail below.

Fix a small circuit tampering C and consider a uniformly random (s, x). For $C(s, x) = (\tilde{s}, \tilde{x})$. Let g, f be the functions corresponding to the $\tilde{s}$ and $\tilde{x}$ portions of C output, respectively ($g(s, x) = \tilde{s}$ and $f(s, x) = \tilde{x}$).

Now, we would like to invoke the security of the PRG to claim that the outcome of non-malleability two-source non-malleability experiment $(\text{2NMExt}(\mathsf{prg}(s), x), \text{2NMExt}(\mathsf{prg}(g(s, x)), f(s, x))$ is statistically close the game where $\mathsf{prg}(s)$ is chosen uniformly at random instead of pseudorandomly.

As mentioned before, our intuition for why the tampering is effectively independent on s and x, is because s is short. In particular, if x is chosen at random conditioned on $s, \tilde{s}$ (in particular, such that $g(s, x) = \tilde{s}$) then (a) s and x are tampered independently, and (b) x retains has high entropy.

So, in the two-source NME game (with sources $(s, y), x$) we have on the one hand that if (s, y) is truly random, then conditioning on $s, \tilde{s}$ yields independent sources with high min-entropy. On the other hand, tampering $(s, y), x$ in either the pseudorandom or random experiment corresponds to producing tampered sources $(\tilde{s}, \tilde{y}) = \mathsf{prg}(g(s, x)) = \mathsf{prg}(\tilde{s})$ which is independent of x, and $\tilde{x} = f(s, x) = f_s(x)$ which is independent of s, y (again when we condition on s). Thus, we have split state-tampering. We are close to being able to apply our conclusion but we have a few obstacles:

- In the PRG game we get $\mathsf{prg}(s)$, but we need to evaluate prg on a correlated input. Because the seed is given explicitly in the distinguishing game, we can only guarantee the output is pseudorandom to adversaries who *cannot* evaluate the PRG.

- Our PRG is secure against boolean distinguishers (single bit output), the output of the two-source NME game is potentially (hopefully) much longer.

To get around all of these things at once we leverage two things in combination. Firstly, every two-source NME is secure even if one of the sources is output jointly in the NME experiment. In other words, for independent (A, B) of sufficient min-entropy, and any g', f' such that one does not contain fixed points,

$$A, \mathrm{NMExt}(A, B), \mathrm{NMExt}(g'(A), f'(B) \approx A, \mathcal{U}, \mathrm{NMExt}(g'(A), f'(B)).$$

Secondly, we use the fact that our PRG is secure against nondeterministic circuits, and hence constant round private coin interactive proofs with efficient verifiers.

In brief, we use our unbounded (honest) prover Merlin to expand the tampered seed. Then we apply a standard trick (dating back to GNI) to efficiently distinguish long outputs with a prover: Arthur completes simulating the experiment flips a coin: if heads he gives Merlin the output of the real experiment and if tails he gives Merlin the output of the ideal experiment (where the output of the extractor on the non-tampered sources is replaced with independent uniform bits). Merlin's maximal advantage in guessing Arthur's bit tightly corresponds to the distance between these distributions.

In the random case, Merlin's view is identically distributed to the outcome of the strong NME experiment (real or ideal depending on Arthur's coin). Because the y and x have sufficient entropy after conditioning to make the tampering split state, we are guaranteed that these distributions are statistically close and Merlin effectively has no advantage (i.e. soundness is very close to $1/2$.)

On the other hand if there is too much error in the pseudorandom case, then we have an (honest) Merlin strategy that can distinguish with noticeable probability (i.e. completeness is bounded away

from a $1/2$ by an inverse polynomial).

See Section 9.2 for the full details.

Extending to Non-Malleable Codes. Given the non-malleable extract, a natural approach to constructing a non-malleable code is to follow the framework laid out by Cheraghchi and Guruswami [11]. They showed efficiently inverting a seedless non-malleable extractor, sampling a uniformly random preimage, suffices to construct non-malleable codes. Unfortunately, there are a number of obstacles to this approach here:

1. It is not clear how to efficiently invert. A natural approach is sample a uniform seed $\mathsf{prg}(s)$ and then invert the two-source extractor conditioned on the left source/state being $\mathsf{prg}(s)$. Unfortunately, known two-source non-malleable extractors are incredibly complex and their inverters often more so. So, we aren't sure if this is possible using current techniques.

2. Above we constructed a (relaxed) seedless non-malleable extractor for uniform sources, while Cheraghchi and Guruswami's transformation requires the non-relaxed version. Cheraghchi and Guruswami showed that relaxed two-source non-malleable extractors for sufficiently low min-entropy imply the non-relaxed variant. While their ideas can be ported to our setting, we would need our seedless non-malleable extractor to work for *recognizable sources* (meeting some min-entropy budget). Recognizable sources are random variables uniformly distributed over the satisfying assignments to a small circuit. We aren't sure how to extend our construction handle such sources without weakening our assumption (to E is hard for exponential size nondeterministic circuits with gates that compute SAT).

3. Finally, our extractor has inverse-polynomial error. Cheraghchi and Guruswami's transformation (provided an efficient inverter) has a 2^k-loss when a code for k-bit messages is desired. This means Black-box impossibilities of Applebaum et al. [65] suggest that new ideas are needed to get negligible error for even normal (not non-malleable) extractors for recognizable sources.

Instead, we take a different approach and use a strong variant of leakage-resilient split-state non-malleable codes with a special encoding property.

Replay the non-malleable extractor argument above, but this time imagine our pseudorandom source, y, x, as instead a (half-pseudorandom) candidate leakage-resilient split-state codeword. The y state is public (and possibly pseudorandom), and x will be sampled privately by Arthur (conditioned on y). (We will sketch how Arthur can sample it later.)

Notice that in our argument above, any adversarial Merlin strategy yields a tampering attack that is nearly split state. The left tampering, corresponding to the adversarial Merlin strategy, only depends on a ℓ-bits of leakage from Arthur's private state, x, (and the ℓ-bit seed s, but this is easily resolved by no longer including s with y as the left state). This is exactly the tampering model that leakage-resilient split-state NMCs can handle!

Our argument relied on the fact that two-source extractor was strong. In the regime of split-state NMC the analogous property is termed "augmented" (due to [32]), where the simulator is required to jointly simulate the result of tampering with one of the codeword states. For the argument above to go through, we require that the leakage can be simulated in addition to one of the states and the outcome of the tampering experiment. We then show that the leakage-resilient split-state to split-state reduction of Ball, Guo, and Wichs [3] yields such a code, provided the underlying code is an augmented split-state NMC.

Next, we note that it's not obvious how to apply the usual definition of non-malleability in this setting where for every tampering function τ there is guaranteed to be a simulator Sim_τ. This is because the tampering function in our proof will depends on Merlin's strategy, but we need Arthur to sample the strategy himself. Instead, we use a simple alternative definition of non-malleability due to Dziembowski et al. [7] that simply requires that the outcome of the tampering experiment for any two messages is indistinguishable, provided the experiment did not output either message. This allows Arthur to sample a private coin at the beginning and play out either experiment as before. Dziembowksi et al. [7] showed that this definition is equivalent to the original for large enough message domains, and we observe that this equivalence extends to the augmented setting.

Finally, we also need to show how to encode efficiently (and allow Arthur) to sample codewords. We follow the (failed) paradigm sketched above for the non-malleable extractor. We sample a seed, expand it to $\mathsf{prg}(s)$ and then "complete" the split-state codeword with x sampled from the correct conditional distribution (encodings of the message m with left codeword state $\mathsf{prg}(s)$). The comparatively simple split-state non-malleable code of Aggarwal, Dodis, and Lovett [62] allows one to efficiently "complete" codewords in this manner because the inner code is simply an inner product. We observe that the compiler of Ball, Guo, and Wichs [3] additionally preserves this ability to complete codewords.

See Section 9.3 for details on the leakage-resilient split-state code and Section 9.4 for details on the NMC.

9.2 Relaxed seedless non-malleable extractor for small circuit tampering

We begin by presenting the simplest variant of our result, a seedless non-malleable "extractor." This proof captures our central idea and is enough to build a non-malleable code for small circuit tampering that can encode a few bits if we encode by rejection sampling. However, in later sections we will extend this basic argument to construct efficient non-malleable codes for longer messages.

Lemma 26. *For any polynomial $s(n)$, there exists a polynomial $s'(n) > s(n)$ such that the following is true. Let $\ell(n') = O(\log n)$ be the function from Theorem 33 for $\mathsf{prg} : \{0,1\}^{\ell(n')} \to \{0,1\}^{n'}$, where $n' = \ell(n') + n$.*

If $\text{2NMExt} : \{0,1\}^{2n} \to \{0,1\}^m$ is a strong two-source non-malleable extractor with error $\delta(n) < (s'(n))^2/16$ for independent sources of length n with min-entropy deficiency $d(n) = \omega(\log n)$ computable in time $o(s(n))$, and $\mathsf{prg} : \{0,1\}^{\ell(n')} \to \{0,1\}^{n'}$ is a seed-extending PRG for nondeterministic circuits of size $s'(n)$, then the construction, $\text{NMExt} : \{0,1\}^{\ell \times n} \to \{0,1\}^m$, in Figure 9.1 is a relaxed seedless non-malleable extractor for uniform sources, (S, X) with respect to $\mathsf{SIZE}[s(n)]$-tampering and error $4/s'(n)$.

Proof. Let $1/s'(n)$ be denote the security of the PRG and let $\epsilon = 4/s'(n)$. (And let δ the secu-

Let $s(n)$,$s'(n)$ c be as in Lemma 26.
Let 2NMExt be a strong two-source non-malleable extractor with error $\delta(n) < (s'(n))^2/16$ for independent sources of length n with min-entropy deficiency $d(n) = \omega(\log n)$, computable in time $o(s(n))$.
Let prg be a PRG for nondeterministic circuits of size $O(s'(n))$.

$$\text{NMExt} : (s, x) \mapsto \text{2NMExt}(\mathsf{prg}(s), x)$$

Figure 9.1: (RELAXED) NON-MALLEABLE EXTRACTOR FOR POLYSIZE CIRCUIT TAMPERING AND UNIFORM SOURCES

rity/error of 2NMExt). Suppose for the sake of contradiction that there exists a successful tampering function, $\tau : (s, x) \mapsto (\tilde{s}, \tilde{x})$ in $\mathsf{SIZE}[s(n)]$

with no fixed points. We will define f to denote the function that computes $(s, x) \mapsto \tilde{x}$ according to τ, and g to denote the function that computes $(s, x) \mapsto \tilde{s}$ according to τ. In other words, $\tau(s, x) = g(s, x), f(s, x)$ and moreover, for each (s, x) either $g(s, x) \neq s$ or $f(s, x) \neq x$.

Now, our assumption on τ (and hence f, g) can be restated as

$$\Delta(\text{2NMExt}(\mathsf{prg}(S), X), \text{2NMExt}(\mathsf{prg}(g(S, X)), f(S, X)); \mathcal{U}_m, \text{2NMExt}(\mathsf{prg}(g(S, X)), f(S, X))) \geq \epsilon. \tag{9.1}$$

We will use this assumption to distinguish the seed-extending PRG, prg, from the uniform distribution via an interactive proof. In more detail, recall that the guarantee of $\mathsf{prg} : \{0,1\}^\ell \to \{0,1\}^{\ell+n}$ says that for any non-deterministic circuit, C, of size $s'(n)$,

$$\Delta(C(\mathsf{prg}(\mathcal{U}_\ell)); C(\mathcal{U}_n)) < 1/s'(n).$$

We show that there exists a circuit C of size at most $s'(n)$ that does not obey this inequality. We do this by following the approach of [65] and constructing a private coin, constant round interactive proof protocol (see Figure 9.2) where Arthur is a circuit of size at most $O(s(n))$ for a promise problem, $\Pi = (\Pi_Y, \Pi_N)$ where Π_Y is $\epsilon/2$-dense under prg and Π_N is $(1 - \epsilon/4)$-dense under the uniform distribution. By standard transformations [64, 63] (Lemma 3), this results in a nondeterministic circuit of size $s'(n)$ that decides the same problem, and hence breaks the PRG.

Looking ahead, Assumption (9.1) above on *malleability* of the resulting extractor when provided pseudorandom inputs will enable us to prove the protocol is complete, i.e. Arthur accepts pseudorandom inputs with high probability. Soundness, i.e. Arthur rejects random inputs with high probability, will ultimately follow from security of the 2-source non-malleable extractor. Furthermore, what ultimately will enable our soundness argument to go through is the fact that to achieve completeness Arthur needs to communicate very little about each sample X and thus X remains entropic, even after conditioning on this communication. We use a standard private coin technique, where Arthur forces Merlin to guess between two samplable distributions [122] to handle the fact that our extractor has relatively long outputs (even though our hardness assumption only holds for boolean distinguishers in a relatively high error regime).

Claim 21. There exists a set Π_Y such that

1. Π_Y is noticeably dense in prg: $\Pr_{s \xleftarrow{u} \{0,1\}^\ell}[\mathsf{prg}(s) \in \Pi_Y] \geq \epsilon/2$
2. Arthur accepts inputs in Π_Y with probability $> \frac{1+\epsilon/2}{2}$ when playing with (honest) Merlin (as prescribed in Figure 9.2).

Proof. If the protocol in Figure 9.2 is given inputs from $\mathsf{prg}(S)$ (where $S \equiv \mathcal{U}_\ell$), then observe that if Arthur chooses $b = 1$, it follows that his final message is sampled as follows:

$$(z, \tilde{z}) \sim \text{2NMExt}(\mathsf{prg}(S), X), \text{2NMExt}(\mathsf{prg}(g(S, X)), f(S, X)).$$

On the other hand, if $b = 0$, Arthur's final message is sampled according to:

$$(z, \tilde{z}) \sim \mathcal{U}_m, \text{2NMExt}(\mathsf{prg}(g(S, X)), f(S, X)).$$

By our assumption, these two distributions are ϵ-far from each other.

By Proposition 3, this implies there exists a set Π_Y such that for any $(s, y) \in \Pi_Y$ these distributions are $\epsilon/2$-far, and moreover $\Pr[\mathsf{prg}(S) \in \Pi_Y] \geq \epsilon/2$.

Let 2NMExt be a strong two-source non-malleable extractor with error $\delta(n)$ for independent sources of length n with min-entropy deficiency $d(n) = \omega(n)$, computable in time $o(s(n))$. Let prg be a seed-extending PRG for nondeterministic circuits of size $s'(n)$.
Recall that f, g correspond to the tampering attack.
Our protocol aims to accept strings from $G(\mathcal{U}_\ell)$ when Merlin plays according to below (completeness) and reject strings from $\mathcal{U}_n$ regardless of the strategy Merlin utilizes (soundness). Because we can amplify by repetition, it suffices for there to be small gap between the two.

On input (s, y)**,**

Arthur Sample $x \leftarrow \mathcal{U}_n$. Send Merlin $\tilde{s} = g(s, x)$.

Merlin If $(s, y) = \mathsf{prg}(s)$, respond $\tilde{y}$ such that $(\tilde{s}, \tilde{y}) = \mathsf{prg}(\tilde{s})$. Otherwise, respond arbitrary $\tilde{y}$.

Arthur Sample a random coin $b \leftarrow \mathcal{U}$ and set $\tilde{z} = \text{2NMExt}((\tilde{s}, \tilde{y}), \tilde{x})$ where $\tilde{x} = f(s, x)$.

- *If* $b = 0$*:* Sample $z \leftarrow \mathcal{U}_m$ and send $z, \tilde{z}$.
- *Else if* $b = 1$*:* Sample $z \leftarrow \text{2NMExt}((s, y), x)$ and send $z, \tilde{z}$.

Merlin (Guess Arthur's bit.) If

$$\Pr_{\mathcal{U}_m, \mathcal{U}_n}[(\mathcal{U}_m, \text{2NMExt}(\tilde{y}, f(s, \mathcal{U}_n))) = (z, \tilde{z}) | g(s, \mathcal{U}_n) = \tilde{s}]$$

is upper bounded by

$$\Pr_{\mathcal{U}_m, \mathcal{U}_n}[(\text{2NMExt}((s, y), \mathcal{U}_n), \text{2NMExt}((\tilde{s}, \tilde{y}), f(s, \mathcal{U}_n))) = (z, \tilde{z}) | g(s, \mathcal{U}_n) = \tilde{s}],$$

set $b' = 1$. Otherwise, set $b' = 0$. Respond b'.

Arthur Accept if $b = b'$, and reject otherwise.

Figure 9.2: INTERACTIVE PROOF FOR DISTINGUISHING prg FROM UNIFORMLY RANDOM BITS

So by Proposition 2.7, Thus for any $\mathsf{prg}(s) \in \Pi_Y$, $b' = b$, Merlin guesses correctly with probability $\geq \frac{1+\epsilon/2}{2}$. □

Claim 22. There exists a set Π_N such that

1. Π_N is large: $\Pr_{(s,y) \xleftarrow{u} \{0,1\}^{\ell+n}}[(s,y) \in \Pi_N] \geq 1 - 4\delta/\epsilon$
2. Arthur accepts inputs in Π_N with probability $\leq \frac{1+\epsilon/4}{2}$ when playing with (honest) Merlin (as prescribed in Figure 9.2).

Proof. As in Claim 9.2, we will analyze the view of Merlin (up to guessing) on a random input and deduce that their exists a large Π_N which Arthur rejects with probability close to 1/2. The difference from there, is here Merlin can behave arbitrarily.

We get around this by observing that Arthur accepts if and only if Merlin guesses his bit, b, correctly ($b' = b$). It follows by Proposition 2.7 that there is an optimal (for any specific input, not just with respect to uniform inputs) Merlin strategy, M^*, that chooses messages to maximize the distance between his view when Arthur chooses $b = 0$ versus his view when $b = 1$. By the optimality of such a strategy, it suffices to consider just this M^*.

So, suppose the protocol in Figure 9.2 is given uniformly random inputs $(s,x) \leftarrow \mathcal{U}_{\ell+n}$. Fix an optimal strategy, M^*. In particular, let $G^* : (s,y,\tilde{s}) \mapsto \tilde{y}$ be the function that given the transcript thus far, outputs Merlin's first message.

Now, note that if conditioned on $s, \tilde{s}$, then $G^*(s,y,\tilde{s}) = \tilde{y}$ is independent of x (as is $\tilde{s}$). And similarly, $\tilde{x} = f(s,x)$ is independent of (s,y). In otherwords, we can sample $(s,y,x,\tilde{s},\tilde{y},\tilde{x})$ identically as follows:

1. Sample s uniformly at random and $\tilde{s} = f(s,x)$ for uniformly random x. (This is identically distributed to Figure 9.2.)
2. Sample y uniformly at random, and sample x uniformly at random, conditioned on $f(s,x) = \tilde{x}$. Note that y and x each have min-entropy $n - \ell$, conditioned on $s, \tilde{s}$. Let Y, $X_{s,\tilde{s}}$ denote these random variables.

3. Apply the tampering:

   - $\tau_L^{\tilde{s}} : (s, y) \mapsto \tilde{s}, \tilde{y}$ where $\tilde{y} = G^*(s, y, \tilde{s})$
   - $\tau_R^s : x \mapsto \tilde{x} = f(s, x)$

   Thus, conditioned on $s, \tilde{s}, \tau_L, \tau_R$ is a split-state tampering. Moreover, because τ has no fixed points, either $f(s, x) \neq x$ or $g(s, x) \neq s$. So, either $\tau_L^{\tilde{s}}$ or τ_R^s contains fixed points.

Thus, for any (valid) fixed choice of $s, \tilde{s}, Y, X_{s,\tilde{s}}$ each have min-entropy $n - \ell$, and $(\tau_L^{\tilde{s}}, \tau_R^s)$ is a split-state tampering function with no fixed points.

Thus, conditioned on $s, \tilde{s}$ and Arthur's coin $b = 0$, Merlin's view is simply

$$T_0^{s,\tilde{s}} \equiv (s, y), \mathcal{U}, \text{2NMExt}(\tau_L^{\tilde{s}}(s, y), \tau_R^s(x)).$$

On the other hand, if Arthur's coin is $b = 1$, Merlin's view is

$$T_1^{s,\tilde{s}} \equiv (s, y), \text{2NMExt}((s, y), x), \text{2NMExt}(\tau_L^{\tilde{s}}(s, y), \tau_R^s(x)).$$

Because 2NMExt is a strong two-source non-malleable extractor with error δ for source with min-entropy deficiency $O(\log n)$, we have that for any (worst-case) choice of $s, \tilde{s}$,

$$\Delta(T_0^{s,\tilde{s}}; T_1^{s,\tilde{s}}) \leq \delta.$$

Thus, by Proposition 2 there exists a set Π_N such that $\Pr_{(s,y) \xleftarrow{u} \{0,1\}^{\ell+n}}[(s, y) \in \Pi_N] \geq 1 - 4\delta/\epsilon$ and for any $(s, y) \in \Pi_N$, Arthur accepts with probability at most $\frac{1+\epsilon/4}{2}$.

It follows from Proposition 2.7 that for any strategy of Merlin and any input $(s, y) \in \Pi_N$, $\Pr[b' = b] \leq \frac{1+\epsilon/4}{2}$. □

We conclude from Claim 9.2 and Claim 9.2, that there is a constant round IP protocol where Arthur can be represented by circuit of size $O(s(n))$ that recognizes $\Pi = (\Pi_Y, \Pi_N)$ with completeness/soundness gap $\epsilon/2$. By Lemma 3, this implies the existence of an $s'(n)$-size nonde-

terministic circuit,C, that decides the promise problem, Π. Because Π_Y is $\epsilon/2$-dense under prg (i.e. $\Pr_s[\mathsf{prg}(s) \in \Pi_Y]$) and Π_N is $1-4\delta/\epsilon$ dense under the uniform distribution (i.e. $\Pr_z[z \in \Pi_Y] \leq 4\delta/\epsilon$). The nondeterministic circuit C can distinguish with advantage $|\epsilon/2-4\delta/\epsilon| \geq \epsilon/4 = 1/s'(n)$. So, our initial assumption must be false. □

9.3 Augmented Leakage-Resilient Split-State

In this section we observe that the leakage-resilient compiler of Chapter 4 preserves two properties of an underlying non-malleable code that are of interest to us:

- **Augmented split-state non-malleability:** there exists a simulator which can simulate the joint distribution of the left (or right) codeword states in addition to the outcome of non-malleability experiment.

  Because in the leaky split-state tampering setting the tampered left (or right) codeword additionally depends on the leakage, we also require the simulator to output this as well.

- **Special encoding:** there exists a special encoding procedure that given a desired left codeword state (or right) and message can output a valid encoding. Importantly, the special encoder is given uniform left codeword states, the output is identically distributed to encodings of the message.

We conclude the section by observing that the definition and equivalence of alternative-non-malleability (Definition 7 and Lemma 2) extends to augmented non-malleability.

9.3.1 Augmenting the construction of Chapter 4

Here we sketch how to prove that the construction from Chapter 4 preserves augmented non-malleability. In particular, we show how to extend the analysis of [3].

Lemma 27. *For any $\alpha \in (0, 1/4)$ there exists a setting of parameters such that the following holds.*

If $(\mathrm{E}', \mathrm{D}')$ is an efficient ϵ-augmented-split-state non-malleable code, then $(\mathrm{Enc} \circ \mathrm{E}', \mathrm{D}' \circ \mathrm{Dec})$ is an efficient α-leakage-resilient 2ϵ-augmented split-state non-malleable code.

Sketch. In Chapter 4 we observed that for any leaky split-state tampering strategy and any message (corresponding to a split-state codeword), the joint distribution, Z, of the leakage-resilient encryption keys, transcript used to specify Alice and Bob's ultimate tampering, and the tampered leakage-resilient encryption keys was statistically close to its distribution of these items when the leaky-split state tampering strategy was applied to encodings of the all zero , Z^0. (In other words, this distribution can be simulated independently of the input. Notably, this means the leakage can be simulated.) Moreover, conditioned on these items, Alice's and Bob's tampering is independent. So the result of the leaky split-state experiment with respect to $(\mathrm{Enc}, \mathrm{Dec})$ was statistically close to some distribution $(g_L^{Z^0}, g_R^{Z^0})$ over split-state tampering functions.

We simply observe that after conditioning on some choice z from the simulated distribution Z^0 and the local randomness sampling the split-state tampering, the left codeword is simply a function of the left input which we can get from the underlying augmented split-state simulator (and the corresponding leakage, or transcript, is simply a part of z). So, we can simulate as follows:

1. Sample $z^0 = (\mathsf{key}_L, \mathsf{key}_R, \mathsf{trans}, \tilde{\mathsf{key}}_L, \tilde{\mathsf{key}}_R) \leftarrow Z^0$, where Z^0 denotes the random variable corresponding to the distribution of leakage-resilient encryption keys, transcript used to specify Alice and Bob's ultimate tampering, and the tampered leakage-resilient encryption keys when $0^k, 0^k$ is encoded in the non-malleability game.

2. Sample u_L, y_L, u_R, y_R uniformly at random, and derive the resulting split-state tampering functions $g_L^{z_0,u_L,y_L}, g_R^{z_0,u_R,y_R}$. Additionally, let z_L denote $\mathsf{Ext}(u_L; 2\mathsf{Ext}(\mathsf{key}_L, y_L)$.

3. Invoke the simulator for the underlying split-state non-malleable code with respect to our tampering functions $g_L^{z_0,u_L,y_L}, g_R^{z_0,u_R,y_R}, (c_L, s) \leftarrow \mathsf{Sim}'(g_L^{z_0,u_L,y_L}, g_R^{z_0,u_R,y_R})$ (where $s \in \{0,1\}^k \cup \{\mathtt{same}\}$).

4. Output $(u_L, y_L, z_L + c_L), \mathsf{trans}, s$.

□

Now, we conclude by recalling that the construction of [62] (Theorem 5) is an augmented split-state non-malleable code with special encoding.[5]

Lemma 28 ([62])**.** *There exist efficient* $\exp(-n^{\Omega(1)})$*-augmented-split-state non-malleable codes with special encoding.*

Our main theorem of this section is a corollary of Lemma 13 and Lemma 28.

Theorem 34. *For any constant* $\alpha \in (0, 1/4)$*, there exist efficient* α*-leakage-resilient* $\exp(-n^{\Omega})$*-augmented-split-state non-malleable codes with special encoding.*

9.3.2 Augmenting alternate-non-malleability

In this section, we observe that the alternative definition of non-malleability of [7] (Def. 32) can be extended to handle this notion of augmented non-malleability. As before these definitions are equivalent up to an additive factor of 2^{-k} in the security parameter.

We give a definition for the specific setting of leakage-resilient split-state.

Definition 32 (Alternate Definition of Augmented Non-Malleability)**.** Let $\mathcal{F}$ be a family of α-leaky split-state tampering functions. We say that a coding scheme $(\mathrm{Enc}, \mathrm{Dec})$ is an α-leakage-resilient augmented-split-state ϵ-alternative-non-malleable if for any $m_0, m_1 \in \{0,1\}^k$ and any $f \in \mathcal{F}$, we have:

$$\mathrm{AltANM}^{f}_{m_0,m_1}(1) \approx_{\epsilon} \mathrm{AltANM}^{f}_{m_0,m_1}(1)$$

where we define the two experiments by

$$\mathrm{AltANM}^{f}_{m_0,m_1}(b) := \left\{ \begin{array}{c} {}_{L}, c_R \leftarrow \mathrm{Enc}(m_b), \tilde{c}_L, \tilde{c}_R \leftarrow f(c_L, c_R), \tilde{m} = \mathrm{Dec}(\tilde{c}) \\ \text{Output } (c_L, \texttt{same}) \text{ if } \tilde{m} \in \{m_0, m_1\}, \text{ and } (c_L, \tilde{m}) \text{ otherwise.} \end{array} \right\}$$

The following lemma follows from the analysis given in [7].

[5]The special encoding is not noted explicitly in [62] but is easy to observe because to encode a message x, first it is encoded as $y \in \mathbb{F}_p$ (via an affine evasive encoding scheme), and then the encoder simply chooses A, B in $\mathbb{F}_p^n$ uniformly at random such that $\langle A, B \rangle = y$.

Lemma 29. *Let* (Enc, Dec) *be a coding scheme for k-bit messages.*

If (Enc, Dec) *is an α-leakage-resilient augmented-split-state ϵ-non-malleable code, then it is an α-leakage-resilient augmented-split-state 2ϵ-alternate-non-malleable code.*

If (Enc, Dec) *is an α-leakage-resilient augmented-split-state ϵ-alternate-non-malleable code, then it is an α-leakage-resilient augmented-split-state $(\epsilon + 2^{-k})$-alternate-non-malleable code.*

9.4 A Non-Malleable Code for Small Circuit Tampering

In this section, we prove our main technical lemma. The main result of this chapter, Theorem 32 is a corollary of this Lemma in concert with Lemma 29 and Theorem 34.

Lemma 30. *For any polynomial $s(n)$, there exists a polynomial $s'(n) > s(n)$ such that the following is true. Let $\ell(n) = O(\log n)$ be the function from Theorem 33 for* $\mathsf{prg} : \{0,1\}^{\ell(n)} \to \{0,1\}^n$.

If $\mathsf{alrssEnc} : \{0,1\}^k \to \{0,1\}^{2n}, \mathsf{alrssDec} : \{0,1\}^{2n} \to \{0,1\}^k$ *is an augmented $\alpha(n)$-leakage-resilient two-source δ-non-malleable code with special encoding, computable in time $o(s(n))$, and* $\mathsf{prg} : \{0,1\}^{\ell(n)} \to \{0,1\}^n$ *is a seed-extending PRG for nondeterministic circuits of size $O(s(n)^c)$ such that $\ell(n) \leq \alpha(n)$ and $\delta < (s'(n))^2/32$, then the construction,* (E, D) *in Figure 9.3 is a $4/s'(n)$-alternate-non-malleable code resilient to* $\mathsf{SIZE}[s(n)]$*-tampering with error $4/s'(n)$.*

Proof. Let $\epsilon(n) = 4/s'(n)$ (the target error of our non-malleable code). Recall taht $1/s'(n)$ is the advantage bound of the PRG, prg. And $(\mathsf{alrssEnc}, \mathsf{alrssDec})$ is δ-non-malleable (with additional properties).

For the sake contradiction, assume (E, D) does not satisfy ϵ-alternate-non-malleability: namely, there exists $m_0, m_1 \in \{0,1\}^k$ and tampering function τ of size $s(n)$ such that

$$\mathrm{AltNM}^{\tau,\mathrm{E},\mathrm{D}}_{m_0,m_1}(0) \not\approx_{4/\epsilon} \mathrm{AltNM}^{\tau,\mathrm{E},\mathrm{D}}_{m_0,m_1}(1)$$

As before, we will use this fact (as well as the security of the underlying leakage-resilient augmented-split-state non-malleable code) to break the pseudorandomness guarantee of prg by

Let $(\mathsf{alrssEnc}, \mathsf{alrssDec})$ be an augmented leakage-resilient two-source non-malleable code with special encoding. Recall that special encoding means that $\mathsf{alrssEnc}$ that takes a state y as input, in addition to the message m, and outputs $\mathsf{alrssEnc}'(m, y) = (y, X)$ with the property that $\mathsf{alrssEnc}(\cdot, \mathcal{U}), \mathsf{alrssDec})$ is an augmented leakage-resilient two-source non-malleable code.
Let prg be a PRG for nondeterministic circuits of size $O(s(n))$.

Encoding (E) : On input m, do the following

Sample $s \leftarrow \mathcal{U}_\ell$. Sample $(\mathsf{prg}(s), x) \leftarrow \mathsf{alrssEnc}^*(m; \mathsf{prg}(s))$.

Output $\mathrm{E}(m) = (s, x)$.

Decoding (D) : On input $(\tilde{s}, \tilde{x})$, do the following

Compute $\tilde{m} = \mathsf{alrssDec}(\mathsf{prg}(\tilde{s}), \tilde{x})$.

Output $\mathrm{D}(\tilde{s}, \tilde{x}) = \tilde{m}$.

Figure 9.3: NON-MALLEABLE CODE FOR POLYSIZE CIRCUIT TAMPERING

designing a constant-round private coin interactive proof that distinguishes with some non-trivial soundness/completeness gap.

For any $\tau : (s, x) \mapsto (\tilde{s}, \tilde{x})$ in $\mathsf{SIZE}^{\Sigma_k}[s(n)]$. Define f to denote the function that computes $(s, x) \mapsto \tilde{x}$ according to τ, and g to denote the function that computes $(s, x) \mapsto \tilde{s}$ according to τ. In other words, $\tau(s, x) = g(s, x), f(s, x)$.

Claim 23. There exists a set Π_Y such that

1. Π_Y is noticeably dense in prg: $\Pr_{s \xleftarrow{u} \{0,1\}^\ell}[\mathsf{prg}(s) \in \Pi_Y] \geq \epsilon/2$
2. Arthur accepts inputs in Π_Y with probability $> \frac{1+\epsilon/2}{2}$ when playing with (honest) Merlin (as prescribed in Figure 9.2).

Proof. If the protocol in Figure 9.2 is given inputs from $\mathsf{prg}(S) = (S, \mathsf{prg}'(S))$ (where $S \equiv \mathcal{U}_\ell$), then upon the choice of $b = 1$, Arthur's final message is exactly that of the alternate-non-malleability game :

$$z \sim \mathrm{AltNM}^{\tau}_{m_0, m_1}(1).$$

Recall that $(\mathsf{alrssEnc}, \mathsf{alrssDec})$ is an augmented leakage-resilient split-state non-malleable code with special encoding, $\mathsf{alrssEnc}'$. Define $\mathsf{alrssEnc}''$ to be the $\mathsf{alrssEnc}'$ that just outputs the right state, i.e. if $\mathsf{alrssEnc}' : (m, y; r) \mapsto (y, x)$ then $\mathsf{alrssEnc}'' : (m, y; r) \mapsto x$.
Recall that prg is a PRG for nondeterministic circuits of size $O(s(n))$. Finally, recall that f, g correspond to the tampering attack.
Our protocol aims to accept strings from $\mathcal{U}_\ell, G(\mathcal{U}_\ell)$ when Merlin plays according to below (completeness) and reject strings from $\mathcal{U}_{\ell+n}$ regardless of the strategy Merlin utilizes (soundness). Because we can amplify by repetition, it suffices for there to be small gap between the two.
Hardcoded into Arthur as non-uniform advice are f, g and m_0, m_1.

On input s, y:

Arthur Sample coin $b \leftarrow \mathcal{U}$. Sample encoding $(y, x) \leftarrow \mathsf{alrssEnc}'(m_b, y)$. Send Merlin $\tilde{s} = g(s, x)$.

Merlin If $(s, y) = \mathsf{prg}(s)$, respond $\tilde{y}$ such that $(\tilde{s}, \tilde{y}) = \mathsf{prg}(\tilde{s})$. Otherwise, respond arbitrary $\tilde{y}$.

Arthur Set $z' = \mathsf{alrssDec}(\tilde{y}, \tilde{x})$ where $\tilde{x} = f(s, x)$. If $z' \in \{m_0, m_1\}$, set $z = \texttt{same}$. Otherwise, set $z = z'$. Send z to merlin.

Merlin (Guess Arthur's bit.) If

$$\Pr[\mathsf{alrssDec}(\tilde{y}, f(s, \mathsf{alrssEnc}''(m_0, y))) = z | g(s, \mathsf{alrssEnc}'(m_0, y)) = s']$$

is upper bounded by

$$\Pr[\mathsf{alrssDec}(\tilde{y}, f(s, \mathsf{alrssEnc}''(m_1, y))) = z | g(s, \mathsf{alrssEnc}'(m_1, y)) = s']$$

set $b' = 1$. Otherwise, set $b' = 0$. Respond b'.

Arthur Accept if $b = b'$, and reject otherwise.

Figure 9.4: INTERACTIVE PROOF FOR DISTINGUISHING prg FROM UNIFORMLY RANDOM BITS

Similarly, if $b = 0$, Arthur's final message is sampled according to:

$$(z, \tilde{z}) \sim \mathrm{AltNM}^{\tau}_{m_0,m_1}(0).$$

By our assumption, these two distributions are ϵ-far from each other.

By Proposition 3, this implies there exists a set Π_Y such that for any $(s, y) \in \Pi_Y$ these distributions are $\epsilon/2$-far, and moreover $\Pr[\mathsf{prg}(S) \in \Pi_Y] \geq \epsilon/2$.

By Proposition 2.7, for any $(s, y) \in \Pi_Y$ Merlin guesses b correctly and Arthur accepts with probability $\geq \frac{1+\epsilon/2}{2}$. □

Claim 24. There exists a set Π_N such that

1. Π_N is large: $\Pr_{(s,y)\xleftarrow{u}\{0,1\}^{\ell+n}}[(s, y) \in \Pi_N] \geq 1 - 8\delta/\epsilon$
2. Arthur accepts inputs in Π_N with probability $\leq \frac{1+\epsilon/4}{2}$ when playing with (honest) Merlin (as prescribed in Figure 9.2).

Proof. Soundness follows from first observing that any Merlin strategy corresponds to some α-leaky split-state tampering on the augmented-leakage resilient split state-code. We conclude soundness because Merlin's view is that of the alternate leakage-resilient augmented-split-state game. As with we the case of the non-malleable extractor, we use the existence the optimality of some optimal strategy M^* (who, for any input (s, y), chooses messages to maximize the distance of his view when Arthur chooses $b = 0$ versus his view when Arthur chooses $b = 1$) to apply the Markov argument to a single distribution.

Fix an optimal Merlin strategy M^* as described above and assume s, y are uniformly distributed. We make some observations about the protocol in this case:

1. **Well-formed augmented leakage-resilient split-state encodings.**

   Uniform $y \sim \mathcal{U}$ means our leakage-resilient augmented-split-state codewords are properly distributed, namely for $b = 0, 1$ it is the case that $\mathsf{alrssEnc}'(m_b, \mathcal{U}) \equiv \mathsf{alrssEnc}(m_b)$.

Moreover, s is independent of the split-state codeword (x, y) sampled by Arthur at the beginning.

2. **ℓ-leaky split-state tampering.**

   Arthur's first message to Merlin, corresponding to the random variable $\tilde{s} = g(s, x)$, can be viewed as ℓ-bits of leakage from the right codeword state (to the left tampering function).

   Thus, we have $\tilde{x} = f(s, x)$ and $\tilde{y} = M^*(s, y, g(s, x))$ which for any fixed choice of s is an ℓ-leaky split-state tampering, Π^s. Thus when s is random, Π^s is a distribution over ℓ-leaky split-state tampering functions.

3. **Merlin's view is identical to augmented alternate-non-malleable game.**

   Recall that Merlin's view corresponds to the variables $(s, y, \tilde{s}, z) = \mathsf{View}^{M^*}(b)$, where b is Arthur's initial coin. Observe that $(y, \tilde{s}, z)$ is sampled identically to $\mathrm{AltANM}^{\Pi^s, \mathsf{alrssEnc}, \mathsf{alrssDec}}(b)$, where b is Arthur's initial coin toss. And s is independent of the initial encoding in the AltANM game, which has worst case guarantees that apply to Π^s for any choice of s.

Putting these observations together, we have by that, because $(\mathsf{alrssEnc}, \mathsf{alrssDec})$ is an ℓ-leakage-resilient δ-augmented-split-state non-malleable code,

$$\mathsf{View}^{M^*}(0) \approx_{2\delta} \mathsf{View}^{M^*}(1).$$

Observe that if there existed a strategy M' and input (s, y) such that the distance between the view of M' on $b = 0$ vs $b = 1$ was greater than that of $M*$, this would contradict the optimality of M^*. Thus, by Proposition 2 there exists a set, Π_N such that $\Pr_{(s,y) \xleftarrow{u} \{0,1\}^{\ell+n}}[(s, y) \in \Pi_N] \geq 1 - 8\delta/\epsilon$ and for each $(s, y) \in \Pi_N$ and any Merlin strategy M', the view when $b = 0$ is $\epsilon/4$-far from the view when $b = 1$.

Thus, by Proposition 2.7, this means for any $(s, y) \in \Pi_N$, any Merlin strategy outputs b' such that $b' = b$ with probability at most $\frac{1+\epsilon/4}{2}$. □

We conclude from Claim 9.4 and Claim 9.4, that there is a constant round IP protocol where Arthur can be represented by circuit of size $O(s(n))$ that recognizes $\Pi = (\Pi_Y, \Pi_N)$ with completeness/soundness gap $\epsilon/2$. By Lemma 3, this implies the existence of an $s'(n)$-size nondeterministic circuit,C, that decides the promise problem, Π. Because Π_Y is $\epsilon/2$-dense under prg (i.e. $\Pr_s[\mathsf{prg}(s) \in \Pi_Y]$) and Π_N is $1-8\delta/\epsilon$-dense under the uniform distribution (i.e. $\Pr_z[z \in \Pi_Y] \leq 4\delta/\epsilon$). The nondeterministic circuit C can distinguish with advantage $|\epsilon/2-8\delta/\epsilon| \geq \epsilon/4 = 1/s'(n)$. So, our initial assumption must be false. □

Chapter 10: Black-Box Barriers

In the previous chapter we presented a non-malleable code for tampering by circuits of bounded polynomial size, assuming E requires nondeterministic circuits of exponential size. However, it is not clear if this assumption is necessary. From Section 3.3, we know that this NMC implies lower bounds on polynomial size circuits. In this chapter, we investigate whether the necessary circuit lower bounds on the tampering class suffice for constructing non-malleable codes. Our specific focus is on classes just beyond the reach of current circuit lower bound techniques, such as NC^1.

In Chapters 5-8 we presented unconditional constructions of non-malleable codes for a variety of circuit classes for which strong unconditional lower bounds are known. In fact, the construction of Chapter 7 remains secure for circuit depths as large as $\Theta(\log(n)/\log\log(n))$. Moreover, due to the impossibility of efficient NMC for all of P, extending their result to obtain unconditional NMC for circuits with asymptotically larger depth would require separating P from NC^1, a problem that is well out of reach with current complexity-theoretic techniques. However, rather than ruling out such constructions entirely, in this regime we ask what are the minimal assumptions necessary for achieving non-malleable codes for NC^1, as well as other classes $\mathcal{F}$ that are believed to be strictly contained in P.

The above question was partially addressed by Ball et al. [22, 20] in their recent work, where they presented a general framework for construction of computationally-secure NMC for various classes $\mathcal{F}$ in the CRS model and under cryptographic assumptions. Instantiating their framework for NC^1 yields a computational, CRS-model construction of 1-bit NMC for NC^1, assuming there is a distributional problem that is hard for NC^1, but easy for P. Moreover, such distributional problems for NC^1 can be based on worst-case assumptions.[1]

[1] Assuming $\oplus\mathsf{L}/\mathrm{poly} \not\subseteq \mathsf{NC}^1$ yields a distributional problem since randomized encodings for $\oplus\mathsf{L}/\mathrm{poly}$ are known to exist [123, 124, 125, 126].

In this chapter, our focus is on whether 1-bit non-malleable codes for NC^1 in the standard (no-CRS) model can be constructed from the assumption that there are distributional problems that are hard for NC^1 but easy for P. Recall that this assumption is minimal, since the decoding function of a 1-bit non-malleable code for NC^1 w.r.t. the distribution of random encodings of 1 bit messages yields such a distributional problem.

We provide a negative answer, showing that, under black-box reductions (restricting use of the tampering function in the security proof to be black-box), this is impossible.

Specifically, we define a notion of black-box reductions for the setting of 1-bit non-malleable codes (E, D) against a complexity class $\mathcal{F}$ to a distributional problem (Ψ, L) that is hard for $\mathcal{F}$. This type of reduction is required to use the "adversary"—i.e. the tampering function in our setting—in a black-box manner, but is not restricted in its use of the underlying assumptions. To motivate our new notion, we begin by recalling the notions of reductions in complexity theory and cryptography, and how they are used.

Reductions in Cryptography. Reductions in cryptography are exactly like reductions in complexity theory. For example, the seminal result of [127] proves by reduction that *breaking a pseudorandom function* (Problem 1) is as hard as *breaking a pseudorandom generator* (Problem 2). In order to prove this, they present a reduction R such that that given an algorithm A that breaks the constructed pseudorandom function, R^A breaks the underlying pseudorandom generator. Note that since R only has oracle access to A, so R does not care how A works, as long as it exhibits input-output behavior that qualifies it as a valid distinguisher between a pseudorandom and random function. Thus, R is black-box. Further, R is only useful if it is polynomial time, since otherwise R can break the pseudorandom generator on its own. Furthermore, we again want to use the existence of the reduction to draw conclusions about the security relationship between the pseudorandom function and the pseudorandom generator. Here we want to show that if there exists a polynomial-time algorithm that breaks the pseudorandom function, then there exists a polynomial-time algorithm that breaks the pseudorandom generator. Therefore, we want it to be the case that

whenever A is polynomial time, R^A is also polynomial time. This trivially holds, as before, since P is closed under composition. However, in the following we will consider cases where this type of closure does not necessarily hold. For example, when A and R are in NC^1, R^A may no longer be in NC^1 (in fact R^A could have depth up to $\log^2(n)$). We therefore need to include a notion of closure under composition as one of the requirements of a black-box reduction in our setting.

A fine-grained setting: Security reductions for non-malleable codes. What would a security reduction in the setting of non-malleable codes look like? In this case, we want to show that *breaking the non-malleable code* (Problem 1) is as hard as *breaking distributional problem* (Ψ, L) (Problem 2). Here, an algorithm that breaks the non-malleable code simply consists of a *tampering function* f. A reduction R is provided black-box access to the tampering function f and must use it to break the distributional problem (Ψ, L). First, note that since we assume R is black-box, R is only allowed to use f as a subroutine (gives it inputs and obtains its output), regardless of *how* f performs its computation. Thus, as in all the cases discussed above, we require that R^f break the distributional problem (Ψ, L), even in the case that f is not contained in $\mathcal{F}$. Note that the distributional problem (Ψ, L) is *easy* for polynomial-time. Therefore, for the reduction to be non-trivial, R must be in a complexity class that does not contain P. Indeed, since we only assume that (Ψ, L) is hard for $\mathcal{F}$, R must be contained in $\mathcal{F}$ in order for us to draw any conclusions (otherwise, we cannot rule out the possibility that R simply ignores its oracle and solves (Ψ, L) on its own). Furthermore, as discussed above, the point of the reduction is to be able to conclude that if there is a tampering function f in $\mathcal{F}$ that breaks the non-malleable code, then there exists an algorithm in $\mathcal{F}$ that breaks the distributional problem (Ψ, L). Therefore, it is not enough that $R \in \mathcal{F}$, and we actually need that whenever $f \in \mathcal{F}$, $R^f \in \mathcal{F}$. We will then use the fact that R^f breaks (Ψ, L) and is used to obtain a contradiction to the hardness of (Ψ, L) for $\mathcal{F}$.

Overall, at a high level (skipping some technical details), we require two properties of a black-box reduction R from (E, D) to (Ψ, L):

- If the tampering function f succeeds in breaking the non-malleable code, the reduction, R^f,

should succeed, regardless of whether $f \in \mathcal{F}$. This represents the fact that R uses f in a black-box manner.

- For any $f \in \mathcal{F}$, R^f must also be in $\mathcal{F}$, and in particular, R itself must be in $\mathcal{F}$. This represents the fact that the black-box reduction R should allow one to obtain a contradiction to the assumption that (Ψ, L) is hard for $\mathcal{F}$, in the case that (E, D) is malleable by $\mathcal{F}$.

Note that for arbitrary classes $\mathcal{F}$ (unlike the usual polynomial-time adversaries typically used in cryptography), the fact that $R \in \mathcal{F}$ and $f \in \mathcal{F}$ does not necessarily imply that $R^f \in \mathcal{F}$. This introduces some additional complexity in our definitions and also requires us to restrict our end results to classes $\mathcal{F}$ that behave appropriately under composition.

We present general impossibility results for constructing 1-bit non-malleable codes for a class $\mathcal{F}$ from a distributional problem that is hard for $\mathcal{F}$ but easy for P. We present three types of results: results ruling out *security parameter preserving* reductions for tampering class $\mathcal{F}$ that behave nicely under composition; results ruling out *"approximate" security parameter preserving* reductions for tampering class $\mathcal{F}$ with slightly stronger compositional properties; and results ruling out *non-security parameter preserving* reductions for tampering class $\mathcal{F}$ that are fully closed under composition. See Definitions 35, 36 and Lemmas 31, 32, 33 for the formal statements.

Briefly, security parameter preserving reductions have the property that the reduction only queries the adversary (in our case the tampering function) on the same security parameter that it receives as input. The notion of "approximate" security parameter preserving reductions is new to this work. Such reductions are parameterized by polynomial functions $\ell(\cdot), u(\cdot)$ and on input security parameter n, the reduction may query the adversary on any security parameter in the range $\ell(n)$ to $u(n)$. This notion is somewhat less restrictive than a security parameter preserving reduction. Finally, in a non-security parameter preserving reduction, the reduction receives security parameter n as input and may query the adversary on arbitrary security parameter n'. Note that $n'(n)$ must be in $O(n^c)$ for some constant c, since the reduction must be polynomial time. This notion allows us to rule out the most general type of black-box reduction discussed above.

We can instantiate the tampering class $\mathcal{F}$ from our generic lemma statements with various

classes of interest. Our results on security parameter preserving and approximate security parameter preserving reductions apply to the class NC^1 as a special case. Our result ruling out non-security parameter preserving reductions applies to the class (non-uniform) NC as a special case. See Corollaries 10, 11, 12 for the formal statements. As the proofs are already quite involved, we make the simplifying assumption of deterministic decoding and perfect correctness. However, this is not inherent to the proof and we expect the results to extend to coding schemes with imperfect correctness and randomized decoding.

Do reductions for NMC take the above form? So far, in the non-malleable codes setting, results have either been *unconditional* (e.g. [42, 34]) or have been based on polynomial-hardness assumptions (e.g. [31, 32]). The results that are based on polynomial-hardness assumptions have all used black-box security reductions, in the standard polynomial-time sense [128]. Our notion is new since it captures a fine-grained setting where the underlying distributional problem is, in fact, *easy* for polynomial-time algorithms. As discussed above, this is the minimal computational hardness assumption necessary to construct non-malleable codes for classes $\mathcal{F}$ for which we cannot prove unconditionally that $\mathsf{P} \not\subseteq \mathcal{F}$. While this type of reduction implicitly arises in the work of [22], our work is the first to formally define and explore this notion of fine-grained black-box reductions in a cryptographic setting.

10.1 Technical Overview

We begin by describing our proof showing the impossibility of a black-box, security-parameter preserving reduction, from NMC against the tampering class NC^1, to a distributional problem that is hard for NC^1. The proof for approximately security parameter preserving reductions is essentially the same, and so we subsequently describe the extension to impossibility of a black-box, *non*-security-parameter preserving reduction, from non-malleable codes against the tampering class NC, to a distributional problem that is hard for NC.

Our proof proceeds via the meta-reduction technique. Specifically, consider a black-box re-

duction R, reducing the security of a single-bit non-malleable code against NC^1 to a distributional problem that is hard for NC^1. The form that this reduction takes, is that it submits codewords c to the tampering function f and gets back (tampered) codewords y as responses. The main idea is to begin with a tampering function f, which is not in NC^1. This tampering function receives a codeword c, decodes it to obtain the bit b and then submits a randomly generated encoding of the bit $1-b$. In the proof, we assume the existence of a reduction R such that R^f breaks the underlying distributional problem (this follows from the definition of a black-box reduction). We then switch from f to a tampering function f' that *is* in NC^1, which behaves as follows: Upon receiving a codeword c, f simply responds with a (hardcoded) random codeword c' that encodes a random bit, independent of the bit that is obtained when the decoding algorithm is applied to c. This switch is desirable, since then $R^{f'}$ will be in NC^1 (note that we are guaranteed that $R^{f'}$ is in NC^1, since one of the properties of R is that whenever the tampering function f' is in NC^1, then $R^{f'}$ must also be in NC^1). It remains, however, to show that $R^{f'}$ succeeds in breaking the underlying distributional problem, which then implies that the underlying distributional problem is not hard for NC^1. In order to ensure this, we use a hybrid argument, where responses to queries from R are switched one by one, from responses according to f to responses according to f'. In each step, we must show that the reduction remains successful in breaking the underlying distributional problem. Importantly, in the i-th hybrid, the first $i-1$ responses are answered according to f, the i-th response and on are answered according to f'. Since R is in NC^1, we argue that if R can distinguish the $(i-1)$-st and i-th hybrids, then we obtain a *tampering attack in* NC^1 *on the non-malleable code*. To do this, we construct a tampering function that hardwires the input to R, the transcript (queries and responses) and entire state of R for the first $i-1$ queries made from R to f, the i-th query along with the value b that it decodes to, and the responses to the queries $i+1$ and on. Then, given an input codeword c', the tampering function inserts this value as the response to the i-th query, runs the reduction R from this point on (given the state of R at the point of the response to the i-th query) and responds with the random hardwired queries upon any future queries from R. If R distinguishes between Hybrids $i-1$ and i, then the above yields a distinguisher between randomly

generated encodings of the bit b, versus randomly generated encodings of a random bit. It is not hard to see that such a distinguisher immediately yields a tampering attack, since it can be used to predict the underlying encoded value and the tampering attack can then replace the codeword with an encoding of a bit which is the opposite of the predicted bit.

In the above, note that it is crucial that the reduction is security parameter preserving. Indeed, if R queries codewords c' that are very short (say length $\log^2(n)$) then we can no longer use R to obtain a valid tampering function against the non-malleable code. This is because R has size $\text{poly}(n)$ and depth $\log(n)$, which is not in NC^1 relative to input length $\log^2(n)$. To deal with this problem, we take advantage of the fact that R must be successful even for tampering functions f that work only for very sparse input lengths $\{1, 2, 2^2, 2^{2^2}, \ldots\}$. In this way, we can essentially guarantee that the reduction queries at most a single input length ℓ which is greater than $\log(n)$ and at most $\text{poly}(n)$. We now consider two cases: Either for this input length ℓ it is the case that NC circuits can distinguish encodings of 0 and 1 with probability at least $3/4$, or for this input length ℓ it is the case that NC circuits can distinguish encodings of 0 and 1 with probability at most $1 - 1/\text{poly}(n)$. If we are in the first case, then we can actually honestly run the attack using a NC circuit (in this case we just use the distinguisher to guess the value of the encoding and succeed with probability $3/4$). If we are in the second case, then we can use Impagliazzo's Hard Core Set [66] to find a set of encodings such that a NC circuit can distinguish random encodings of 0 and 1 from this set with probability at most $1/2 + 1/\text{poly}(n)$. In this case, we modify the tampering function to hardcode random encodings *from the hard core set* and return these in response to the queries from R. Note that to obtain contradiction to the security of the constructed non-malleable code, we now require that when the reduction R is composed with any tampering circuit in NC, then the composed circuit is still in NC. This property holds for NC, but not NC^1, which is why our result on ruling out non-security preserving reductions holds only for NC.

10.2 Preliminaries

We begin by present definitions of functionalities Unroll and Replace which will then allow us to define the appropriate notions of composition and closure for function classes for which our methods hold. We conclude this section with the formal definitions of black box reductions.

Definition 33 (Unroll functionality.)**.** Let $F := \{f_n\}_{n=1}^{\infty} \in \mathcal{F}$, where $f_n : \{0,1\}^n \to \{0,1\}$ and $G = \{g_m\}_{m=1}^{\infty} \in \mathcal{G}$, where $g_m : \{0,1\}^m \to \{0,1\}^m$, be function families. Also let t, p be polynomials. Let $m \in \text{poly}(n)$. Let F^G denote families functions $f_n : \{0,1\}^n \to \{0,1\} \in F$ which contains at most $t(n)$ oracle gates computing $g_m : \{0,1\}^m \to \{0,1\}^m \in G$ and get string of length $p(n)$ as non-uniform advice. On an n-bit input, consider the DAG whose left side consists of the circuit of f_n and whose right side consists of circuits $g_{n_1}, \ldots, g_{n_{t(n)}}$. The values of wires going from the left to the right correspond to (a topological ordering of) the oracle queries $x_1, \ldots, x_{t(n)}$ of lengths $n_1, \ldots, n_{t(n)}$, made in each of the $t(n)$ queries. For $i \in [t(n)]$, circuit g_{n_i} takes as input x_i and returns y_i. The values of wires going from the right to the left correspond to the responses $y_1, \ldots, y_{t(n)}$. We say that this DAG, denoted $\mathsf{Unroll}(F^G)$, is an unrolling of $F^G(x)$.

Definition 34 (Replace Functionality.)**.** Consider replacing each g_{n_i}, $i \in [t(n)]$, in $\mathsf{Unroll}(F^G)$ with a circuit g'_{n_i} that takes input $(x_1, \ldots, x_i)$ and produces output y_i. This is denoted by

$$\mathsf{Replace}(\mathsf{Unroll}(F^G), g'_{n_1}, \ldots, g'_{n_{t(n)}})$$

.

Definition 35 ($(\mathcal{G}, t, \ell, u)$-closure of $\mathcal{F}$)**.** Let $F := \{f_n\}_{n=1}^{\infty} \in \mathcal{F}$, where $f_n : \{0,1\}^n \to \{0,1\}$ and $G = \{g_m\}_{m=1}^{\infty} \in \mathcal{G}$, where $g_m : \{0,1\}^m \to \{0,1\}^m$, be function families. Also let t, ℓ, u be polynomials, and $\ell(n) \leq m \leq u(n)$. Let $f_n{}^{g_m}$ denote function $f_n : \{0,1\}^n \to \{0,1\}$ which has access to the output of $g_m : \{0,1\}^m \to \{0,1\}^m$ on at most $t(n)$ inputs of its choice.

We say that $\mathcal{F}$ is $(\mathcal{G}, t, \ell, u)$-closed under compositions if for every $F \in \mathcal{F}$ such that for all

$G \in \mathcal{G}$, $\mathsf{Unroll}(F^G) \in \mathcal{F}$, we have that for all $G' \in \mathcal{G}$ and all $g'_{n_1}, \ldots, g'_{n_{t(n)}} \in G'$,

$$\mathsf{Replace}(\mathsf{Unroll}(F^G), g'_{n_1}, \ldots, g'_{n_{t(n)}}) \in \mathcal{F}$$

.

Definition 36 ($(\mathcal{G}, t)$-closure of $\mathcal{F}$ under Strong Composition)**.** Let $F := \{f_n\}_{n=1}^{\infty} \in \mathcal{F}$, where $f_n : \{0,1\}^n \to \{0,1\}$ and $G = \{g_m\}_{m=1}^{\infty} \in \mathcal{G}$, where $g_m : \{0,1\}^m \to \{0,1\}^m$, be function families. Also let t, p be polynomials. Let $m \in \text{poly}(n)$. Let F^G denote families functions $f_n : \{0,1\}^n \to \{0,1\} \in F$ which contains at most $t(n)$ oracle gates computing $g_m : \{0,1\}^m \to \{0,1\}^m \in G$.

We say that $\mathcal{F}$ is $(\mathcal{G}, t)$-closed under compositions if for every $F \in \mathcal{F}$ we have that for all $G, G' \in \mathcal{G}$ and all $g'_1, \ldots, g'_{t(n)} \in G'$, $\mathsf{Replace}(\mathsf{Unroll}(F^G), g'_1, \ldots, g'_{t(n)}) \in \mathcal{F}$.

10.2.1 Black Box Reductions

Definition 37 (Black-Box-Reduction)**.** We say R is an (F, ϵ, δ)*-black-box reduction from a (single bit) non-malleable code,* $(\mathrm{E}, \mathrm{D}) = \{(\mathrm{E}_n, \mathrm{D}_n)\}_{n=1}^{\infty}$*, to a distributional problem,* $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^{\infty}$, if the following hold:

1. For every set of circuits $\{f_n\}_{n=1}^{\infty}$ parameterized by input length n such that f_n achieves $\epsilon(n)$-malleability, for non-negligible ϵ, i.e.

$$\Pr_{b \xleftarrow{u} \{0,1\}}[\mathrm{D}_n(f_n(\mathrm{E}_n(b))) = 1 - b] > \frac{1}{2} + \epsilon(n),$$

   then R^f solves $\{(\Psi_n, L_n)\}_{n=1}^{\infty}$ with advantage $\delta(n)$, where δ is non-negligible. I.e.

$$\Pr_{x \leftarrow \Psi_n}[L_n(x) = R^{\{f_k\}_{k=1}^{\infty}}(x)] > \frac{1}{2} + \delta(n).$$

2. If $\{f_n\}_{n=1}^{\infty} \in F$, then $R^{\{f_k\}_{k=1}^{\infty}}(x) \in F$.

We say a reduction R is *length-preserving* if R, on input of length n is only allowed to make

queries to oracles with security parameter n. Namely,

$$\Pr_{x \leftarrow \Psi_n}[L_n(x) = R^{f_n}(x)] > \frac{1}{2} + \delta(n).$$

We say a reduction R is *approximately length-preserving* if there are polynomials $p(\cdot), q(\cdot)$ such that R, on input of length n is only allowed to make queries to oracles with security parameter $k \in [p(n), q(n)]$. Namely,

$$\Pr_{x \leftarrow \Psi_n}[L_n(x) = R^{\{f_k\}_{k=p(n)}^{q(n)}}(x)] > \frac{1}{2} + \delta(n).$$

We say a reduction is in NC^1 if it can be written as a family of circuits of $O(\log n)$-depth, poly(n)-size.

10.3 On NMC from Average-Case Hardness via BB Reductions

A crucial component of our impossibility result will be a lookup circuit that responds to queries submitted by the reduction with hardwired responses. However, we need the lookup circuit to maintain consistency: If the reduction queries the same query multiple times, the same response should be given each time. Such a lookup circuit is trivial to implement with polynomial-size circuits. However, in our case, we require that this lookup circuit is implementable in NC^1. In the following, we first formally define such a lookup circuit and then prove that it is implementable in NC^1.

Definition 38 (Look-Up Circuit.)**.** A $(\ell(n), p(n))$ lookup circuit consists of $\ell(n)$ hardwired values of length $p(n)$ bits, denoted $y_1, \ldots, y_{\ell(n)}$. The lookup circuit receives as input $x_1, \ldots, x_{\ell(n)}$, where each x_i has length $p(n)$ bits. The circuit outputs $\ell(n)$ number of $p(n)$-bit strings: $y_{i_1}, \ldots, y_{i_{\ell(n)}}$, where for $j \in [\ell(n)]$, i_j is set to the first index $k \in [\ell(n)]$ such that $x_j = x_k$. For example, on input $x_1, x_2, x_3, x_4, \ldots$, where $x_1 = x_3$ and $x_2 = x_4$, the circuit outputs y_1, y_2, y_1, y_2.

Proposition 7. *For $p(n)$, $\ell(n) = O(n^c)$ for some fixed constant c, there exist polynomial size look-up circuits of depth $O(\log n)$.*

Sketch. The inputs, $x_1, \ldots, x_{\ell-1}$, can be put in sorted order via a circuit of size $O(n^c \log n)$ and depth $O(\log n)$ [129]. Then each sorted x_i can determine if it is the first of that value (if $x_1, \ldots, x_{\ell-1}$ are in sorted order then x_j is determining that there does not exist $x_i = x_j$ such that $i < j$), by comparing only to one neighboring value. This can be done in parallel. Finally, compare x_ℓ to all x_i that pass this test in parallel. If there is such an x_i such that $x_i = x_\ell$, the circuit will output y_i. Otherwise, the circuit will output y_ℓ. □

We now present the central technical lemma of the section.

Lemma 31. *Assume that $\mathcal{F}$ is $(\mathcal{F}, t, p(n), p(n))$-closed under composition (see Definition 35), and contains $(t(n), p(n))$ look-up circuits for polynomials $t(\cdot)$, $p(\cdot)$.[2] If there is an $(\mathcal{F}, 1/2, \delta(n))$-black-box-reduction making $t(n)$ security parameter-preserving queries from a (single bit) non-malleable code for $\mathcal{F}$, $(\mathsf{E}, \mathsf{D}) = \{(\mathsf{E}_n, \mathsf{D}_n)\}_{n=1}^{\infty}$, to a distributional problem, $(\Psi, L) = \{(= \Psi_n, L_n)\}_{n=1}^{\infty}$, then one of the following must hold:*

1. *(E, D) is $\frac{\delta(n)}{2t(n)}$-malleable by $\mathcal{F}$.*
2. *(Ψ, L) is $(\delta(n)/2)$-easy for $\mathcal{F}$.*

Moreover, if (E, D) is efficient, then it suffices that $\mathcal{F}$ contains such look-up circuits generated in uniform polynomial time.

Proof. Let R be such a security parameter-preserving $(\mathcal{F}, 1/2, \delta(n)$-reduction, for a non-malleable code (E, D) and distributional problem (Ψ, L). Moreover, for security parameter n, let $p(n)$ be the length of the codeword generated by E, where $p(\cdot)$ is a polynomial.

Consider the following tampering functions $\{f_{p(n)}\}_{p(n)}$ whose behavior on a given codeword c is defined as follows (where H is a random function $H : \{0,1\}^{p(n)} \to \{0,1\}^*$ and $H(c)$ is the

[2] $p(n)$ corresponds to the length of the codeword outputted by E_n.

randomness used by encoding algorithm):

$$f_{p(n)}(c) := \begin{cases} \mathsf{E}_n(1; H(c)) & \text{if } \mathsf{D}_n(c) = 0 \\ \mathsf{E}_n(0; H(c)) & \text{if } \mathsf{D}_n(c) = 1 \end{cases}$$

Since, NMC are perfectly correct, we have (for any choice of H)

$$\Pr_{b \xleftarrow{u} \{0,1\}} [\mathsf{D}_n(f_{p(n)}(\mathsf{E}(b))) = 1 - b] = 1.$$

Therefore, by our assumption on R we have that for all n,

$$\Pr_{x \leftarrow \Psi_n} [L_n(x) = R^{f_{p(n)}}(x)] \geq \frac{1}{2} + \delta(n).$$

Now, for the j-th oracle query, we define $f'^{,j}_{p(n)}$, a stateful simulation of the output of the tampering function $f_{p(n)}$ on the j-th query. Each $f'^{,j}_{p(n)}$ is a $(j, p(n))$ lookup circuit (with j number of inputs/outputs of length $p(n)$) that hardcodes a random codeword (sampled from $\mathsf{E}(b)$ where b is uniform) as the y_j value.

By our assumption on $\mathcal{F}$ (and R), we have that $\mathsf{Replace}(\mathsf{Unroll}(R^{f_{p(n)}}), f'^{,1}_{p(n)}, \ldots, f'^{,t(n)}_{p(n)}) \in \mathcal{F}$. We will abuse notation and denote the resulting circuit by $R^{f'_{p(n)}}$. So, it suffices to show that the behavior of $R^{f'_{p(n)}}(x)$ is close that of $R^{f^H_{p(n)}}(x)$, for any x, which will imply that $R^{f'_{p(n)}}(x) \in \mathcal{F}$ breaks the distributional problem w.h.p., since $R^{f^H_{p(n)}}(x)$ does. More accurately, if (E, D) is $\frac{\delta(n)}{2t(n)}$-non-malleable by $\mathcal{F}$, then we will show that

$$\forall n \in \mathbb{N}, \forall x \in \{0,1\}^n, \Delta(R^{f'_n}(x); R^{f_n}(x)) \leq \delta(n)/2.$$

By the above, this then implies that (Ψ, L) is $(\delta(n)/2)$-easy for $\mathcal{F}$.

To show that the outputs of $R^{f'_{p(n)}}(x)$ and $R^{f^H_{p(n)}}(x)$ are close, we will use a hybrid argument, reducing to the $\frac{\delta(n)}{2t(n)}$-non-malleablity of (E, D) at every step.

In the i-th hybrid, the function $f^{(i),j}_{p(n)}$ responding to the j-th query is a $(j, p(n))$ look-up circuit

that hardcodes values $y^i_1, \ldots, y^i_j$. For $k \in [t = t(n)]$, the y^i_k values are sampled as follows: For $k \in [t - i]$, y^i_k is sampled as by $f^H_{p(n)}$. For $k > t - i$, y^i_k is a random encoding of a random bit. The concatenation of the t circuits for each query is denoted by $f^{(i)}_{p(n)}$. Clearly, $f^{(0)}_{p(n)} \equiv f_{p(n)}$ and $f^{(t)}_{p(n)} \equiv f'_{p(n)}$.

We will show that for all $x \in \{0,1\}^n$ (and any fixing of random coins r for R) $\Delta(R^{f^{(i)}_{p(n)}}(x); R^{f^{(i-1)}_{p(n)}}(x)) \leq \frac{\delta(n)}{2t(n)}$ (for $i \in [t(n)]$), which proves the claim above. ($R^{f^{(0)}_{p(n)}}(x)$ has advantage $\delta(n)$ and in each of the subsequent $t(n)$ hybrids we lose at most an $\epsilon(n)$ factor.)

Suppose not, then there exists an x (and random coins r, if R is randomized) such that R's behavior differs with respect to $f^{(i)}_{p(n)}$ and $f^{(i-1)}_{p(n)}$: $|\Pr[R^{f^{(i)}_{p(n)}}(x) = 1] - \Pr[R^{f^{(i-1)}_{p(n)}}(x) = 1]| \geq \frac{\delta(n)}{2t(n)}$.

Note that for fixed random function H (that generates the random coins used to sample the y_j values) $f^{(i)}_{p(n)}$ and $f^{(i-1)}_{p(n)}$ differ solely on the response to $(t - i)$-th query. So, fix x, H and all but the $(t - i)$-th value y^i_{t-i} and "hardcode" all other y_k values in both cases. The reason that we can hardcode the y_j values except for the $(t-i)$-th response is the following: Clearly, up to the $(t-i)$-th query, the responses can be fully hardcoded since x is fixed and so all the queries and responses can also be fixed. The y_j values hardcoded in the $(t - i + 1)$-st lookup circuit and on can also be fixed, since in both $f^{(i)}_{p(n)}$ and $f^{(i-1)}_{p(n)}$, the $(t-i+1)$-st value of y_j and on is a random codeword, that does not depend on the value encoded in the query submitted by the reduction. Let $s_{H,x}$ denote the value encoded in the $(t - i)$-th query in this hardcoded variant of the hybrid. Note that the value of $s_{H,x}$ is also fixed.

1. In $R^{f^{(i-1)}_{p(n)}}(x)$ all values up to the $(t - i)$-th response are hardcoded. The $(t - i)$-th response, which will be a random encoding of bit $1 - s_{H,x}$, is not hardcoded. All the other responses are computed by lookup circuits with hardwired y_j values.

2. In $R^{f^{(i)}_{p(n)}}(x)$, all values up to the $(t - i)$-th response are hardcoded. The $(t - i)$-th response, which will be a random encoding of a random bit, is not hardcoded. All the other responses are computed by lookup circuits with hardwired y_j values.

Thus, we will treat the above as a new function $R'_{H,x}(\cdot)$ that takes as input just the response to

the $(t-i)$-th query and returns some value. Note that $R'_{H,x}(\cdot)$ is in $\mathcal{F}$, since it can be viewed as the circuit $R^{f^{(i)}_{p(n)}}$, with queries/responses to $f^{(i),j}$, $j \in [t-i-1]$ hardcoded, the $(t-i)$-th query hardcoded, the $(t-i)$-th value y^i_{t-i} as the input to the circuit, and for $j > t-i$, the $f^{(i),j}$ functions as lookup circuits contained in $\mathcal{F}$. Moreover, by the above, $R'_{H,x}(\cdot)$ distinguishes random codewords that encode the bit $1-s_{H,x}$ from random codewords that encode a random bit with advantage $\epsilon(n)$. Specifically,

$$\Pr[R'_{H,x}(c) = 1 \mid c \leftarrow \mathsf{E}_n(1-s_{H,x})] - \Pr[R'_{H,x}(c) = 1 \mid c \leftarrow \mathsf{E}_n(b), b \leftarrow \{0,1\}] \geq \frac{\delta(n)}{2t(n)}.$$

By standard manipulation, the above is equivalent to:

$$\frac{1}{2} \cdot \Pr[R'_{H,x}(c) = 1 \mid c \leftarrow \mathsf{E}_n(1-s_{H,x})] + \frac{1}{2} \cdot \Pr[R'_{H,x}(c) = 0 \mid c \leftarrow \mathsf{E}_n(s_{H,x})] \geq \frac{1}{2} + \frac{\delta(n)}{2t(n)}.$$

This implies that we can use $R'_{H,x}$ to construct a distribution over tampering functions in $\mathcal{F}$ that successfully break (E, D). Details follow.

Let $c_{s_{H,x}}$ be a codeword encoding bit $s_{H,x}$ and let $c_{1-s_{H,x}}$ be a codeword encoding bit $1-s_{H,x}$. Define $\hat{f}_{H,x}$ as follows: $\hat{f}_{H,x}$ hardcodes $c_{s_{H,x}}$ and $c_{1-s_{H,x}}$. On input (codeword) c,

- If $R'_{H,x}(c) = 1$, output $c_{s_{H,x}}$;
- Otherwise, output $c_{1-s_{H,x}}$.

We now analyze

$$\Pr_{b \leftarrow \{0,1\}}[\mathsf{D}_n(\hat{f}_{H,x}(\mathsf{E}_n(b))) = 1-b].$$

$$\begin{aligned}\Pr_{b\leftarrow\{0,1\}}[\mathsf{D}(\hat{f}_{H,x}(\mathsf{E}(b))) = 1-b] &= \Pr[b = 1-s_{H,x}]\cdot\Pr[R'_{H,x}(c)=1 \mid c\leftarrow \mathsf{E}_n(1-s_{H,x})] \\ &\quad + \Pr[b = s_{H,x}]\cdot\Pr[R'_{H,x}(c)=0 \mid c\leftarrow \mathsf{E}_n(s_{H,x})] \\ &= \frac{1}{2}\cdot\Pr[R'_{H,x}(c)=1 \mid c\leftarrow \mathsf{E}_n(1-s_{H,x})] \\ &\quad + \frac{1}{2}\cdot\Pr[R'_{H,x}(c)=0 \mid c\leftarrow \mathsf{E}_n(s_{H,x})] \\ &\geq \frac{1}{2}+\frac{\delta(n)}{2t(n)}.\end{aligned}$$

But, the above implies that (E,D) is $\frac{\delta(n)}{2t(n)}$-malleable for $\mathcal{F}$.

Therefore, we conclude that either (E,D) is $\frac{\delta(n)}{2t(n)}$-malleable for $\mathcal{F}$ or the distributional problem, $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^{\infty}$ is $(\delta(n)/2)$-easy for $\mathcal{F}$. □

The following corollary holds since NC^1 is $(\mathsf{NC}^1, t, p(n), p(n))$-closed under composition (for all polynomials $p(\cdot)$), and NC^1 contains $(t(n), p(n))$ lookup circuits for any polynomials $t(\cdot), p(\cdot)$.

Corollary 10. *If there is an* $(\mathsf{NC}^1, 1/2, \delta(n))$*-black-box-reduction making* $t(n)$ *security parameter preserving queries from a (single bit) non-malleable code for* NC^1*,* $(\mathsf{E},\mathsf{D}) = \{(\mathsf{E}_n,\mathsf{D}_n)\}_{n=1}^{\infty}$*, to a distributional problem,* $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^{\infty}$*, then one of the following must hold:*

1. (E,D) *is* $\frac{\delta(n)}{2t(n)}$*-malleable by* NC^1*.*
2. (Ψ, L) *is* $(\delta(n)/2)$*-easy for* NC^1*.*

Note 1. The proof of Lemma 31 (as well as the other proofs in this section), does not extend to cases in which the reduction R is outside in the class of tampering functions $\mathcal{F}$. Specifically, in the hybrid arguments, we require that $R'_{H,x}(\cdot)$ is in $\mathcal{F}$. In particular, our proof approach does not extend to proving impossibility of constructing a (single bit) non-malleable code for $\mathcal{F}$, from a distributional problem, (Ψ, L) that is hard for some larger class $\mathcal{F}$. E.g. our techniques do not allow us to rule out constructions of non-malleable codes for NC^1 from a distributional problem that is hard for NC^2. Our techniques also do not rule out constructions of non-malleable codes for $\mathcal{F}$ from

an "incompressibility"-type assumption, such as those used in the recent work of [20]. Briefly, if a function ψ is incompressible by circuit class $\mathcal{F}$, it means that for $t \ll n$, for any *computationally unbounded* Boolean function $D : \{0,1\}^t \to \{0,1\}$ and any $F : \{0,1\}^n \to \{0,1\}^t \in \mathcal{F}$, the output of $D \circ F(x_1, \ldots, x_n)$ is uncorrelated with $\psi(x_1, \ldots, x_n)$ (over uniform choice of $x_1, \ldots, x_n$). In our case, this would mean that the reduction R is allowed oracle access to a computationally unbounded Boolean function D, since the hardness assumption would still be broken by the reduction as long as $R \in \mathcal{F}$ and the query made to D has length $t \ll n$. Since R composed with D is clearly outside the tampering class $\mathcal{F}$, our proof approach does not apply in the incompressibility setting.

Note 2. We can extend Lemma 31 to rule out $(u(n), \ell(n))$-*approximately* security parameter preserving reductions by allowing our reduction access to a greater range of inefficient/simulated tampering functions (defined in the same manner as above): $\{f_k\}_{k=\ell(n)}^{u(n)}$ and $\{f'_k\}_{k=\ell(n)}^{u(n)}$. In this case, we can, WLOG, conflate the security parameter queried to the oracle with the length of the query made to the oracle. However, we now require for our proof that $\mathcal{F}$ is $(\mathcal{F}, t, \ell, u)$-closed under composition and contains look-up circuits with $t(n)$ inputs, consisting of $\ell(n)$ to $u(n)$ number of bits, for polynomials $t(\cdot), \ell(\cdot), u(\cdot)$.

Lemma 32. *Assume $\mathcal{F}$ is $(\mathcal{F}, t, \ell, u)$-closed under composition (see Definition 35) and contains $(t(n), p(n))$ look-up circuits for polynomials $t(\cdot)$, $p(\cdot)$. If there is an $(\mathcal{F}, 1/2, \delta(n))$-black-box-reduction making $t(n)$ number of $(\ell(n), u(n))$-approximately length preserving queries, from a (single bit) non-malleable code for $\mathcal{F}$, $(\mathsf{E}, \mathsf{D}) = \{(\mathsf{E}_n, \mathsf{D}_n)\}_{n=1}^{\infty}$, to a distributional problem, $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^{\infty}$, then one of the following must hold:*

1. *(E, D) is $\frac{\delta(n)}{2t(n)}$-malleable by $\mathcal{F}$.*
2. *(Ψ, L) is $(\delta(n)/2)$-easy for $\mathcal{F}$.*

Moreover, if (E, D) is efficient, then for the conclusion to hold it suffices that $\mathcal{F}$ contains such look-up circuits generated that are generated uniform polynomial time.

The following corollary holds since NC^1 is $(\mathsf{NC}^1, t, \ell, u)$-closed under composition, where

$\ell(n) = n^\gamma$, for any constant $\gamma \leq 1$, $u(n) = n^c$, for any constant $c \geq 1$ and NC^1 contains look-up circuits with $t(n)$ number of inputs of length $\ell(n)$ to $u(n)$-bits for polynomials $t(\cdot), \ell(\cdot), u(\cdot)$.

Corollary 11. *Fix constants $\gamma \leq 1$, $c \geq 1$. If there is an $(\mathsf{NC}^1, 1/2, \delta(n))$-black-box-reduction making $t(n)$ (n^γ, n^c)-approximately length preserving queries from a (single bit) non-malleable code for NC^1, $(\mathsf{E}, \mathsf{D}) = \{(\mathsf{E}_n, \mathsf{D}_n)\}_{n=1}^\infty$, to a distributional problem, $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^\infty$, then one of the following must hold:*

1. *(E, D) is $\frac{\delta(n)}{2t(n)}$-malleable by NC^1.*
2. *(Ψ, L) is $(\delta(n)/2)$-easy for NC^1.*

We extend to non-security parameter preserving reductions, but require a stronger compositional property for the tampering class $\mathcal{F}$. As for approximate security parameter preserving reductions, WLOG we may conflate the security parameter queried to the oracle with the length of the query made to the oracle.

Lemma 33. *Let $\mathcal{F}$ be closed under strong composition (see Definition 36) and contain $(t(n), u(n))$ lookup circuits. If for every non-negligible ϵ, there is an $(\mathcal{F}, \epsilon, \delta(n))$-black-box-reduction (for some non-negligble δ) making $t(n)$ queries from an (single bit) $\epsilon(n)$-non-malleable code for $\mathcal{F}$, $(\mathsf{E}, \mathsf{D}) = \{(\mathsf{E}_n, \mathsf{D}_n)\}_{n=1}^\infty$, to a distributional problem, $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^\infty$, then (Ψ, L) is not $(\delta(n) - t(n) \cdot \epsilon(n))$-hard for $\mathcal{F}$.*

Proof. Let $\mathcal{S} := \{1, 2^1, 2^{2^1}, 2^{2^{2^1}}, \ldots\}$. Let $\epsilon(n)$ be the following non-negligible function:

$$\epsilon(n) := \begin{cases} \frac{1}{4} & \text{if } n \in \mathcal{S} \\ 0 & \text{if } n \notin \mathcal{S} \end{cases}$$

Assume there is some reduction R that succeeds with non-negligible probability $\delta := \delta(n)$ for this ϵ. Since δ is non-negligible, there must be an infinite set $\mathcal{S}'$ such that $\delta(n) \geq 1/n^c$ for some constant c and for all $n \in \mathcal{S}'$.

WLOG, we may assume that the reduction R, on input of length n, queries at most a single input length $\ell(n) \in \omega(\log(n))$, whereas all other queries are of input length $O(\log(n))$ (since we may assume the oracle simply returns strings of all 0's on any input of length $k \notin \mathcal{S}$). Additionally, we may assume that (1) $\ell(n)$ is polynomial in n (since otherwise the reduction does not have time to even write down the query) and (2) for any $k \in \mathbb{N}$, the size of the set $\ell^{-1}(k) \cap \mathcal{S}'$ is finite (otherwise we can hardcode all possible query/responses for a particular input length k into the reduction–which is constant size since k is constant–and obtain a circuit that breaks the underlying hard problem on an infinite number of input lengths). Moreover, we assume WLOG that $\ell(n) < n$, since otherwise our previous proof holds.

Since by assumption $\mathcal{F}$ is HCS-amenable, we can apply Impagliazzo's hard-core set theorem to adversaries in $\mathcal{F}$. Specifically, for random codewords $c \leftarrow \mathsf{E}_{\ell(n)}(b), b \leftarrow \{0,1\}$ of length $\ell = \ell(n)$ s.t. $\ell(n) < n$, there are two possible cases:

1. For infinitely many $n \in \mathcal{S}'$ (this set of values is denoted by $\mathcal{S}'' \subseteq \mathcal{S}'$), there is some adversary in $\mathcal{F} := \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ that outputs $\mathsf{D}_{\ell(n)}(c)$ with probability at least $3/4$[3].

2. For infinitely many $n \in \mathcal{S}'$ (this set of values is denoted by $\mathcal{S}'' \subseteq \mathcal{S}'$), there is some hardcore set $\mathcal{H}$ of size at least $\epsilon'(n) \cdot 2^{\ell}$, where $\epsilon'(n) = \frac{1}{2 \cdot n^c \cdot t(n)}$ such that every adversary in $\mathcal{F} := \{\mathcal{F}_n\}_{n\in\mathbb{N}}$ outputs $\mathsf{D}_{\ell(n)}(c)$ with probability at most $1/2 + \epsilon'(n)$, when c is chosen at random from $\mathcal{H}$[4].

In Case 1, we set the tampering function $\{f_k\}_k$ to use the circuit described above to decode a random codeword with prob $3/4$ and then chooses a random encoding of 0 or 1 appropriately. Additionally, f_k only responds if $k \in \mathcal{S}$. Clearly, f_k succeeds with non-negligible probability ϵ. Since the ϵ function remains the same, we know that δ and $\ell, \mathcal{S}, \mathcal{S}'$ remain the same.

In this case, as in the previous proof, we can switch to a simulated tampering function Sim, which responds with $f_{\ell(n)}$ on query input length $\ell(n)$ and hardcodes all responses for all possible

[3]Note that $\mathsf{D}_{\ell(n)}(c)$ takes inputs of length $\ell(n)$, whereas $\mathcal{F}_n$ takes inputs of length n. We can easily resolve this discrepancy by padding inputs of length $\ell(n)$ up to n.

[4]Again, the input c to $\mathsf{D}_{\ell(n)}$ has length $\ell(n)$ while $\mathcal{F}_n$ takes inputs of length n. As above, we resolve the discrepancy by padding inputs of length $\ell(n)$ up to n.

queries R makes to f_k with input lengths $k = k(n) \in O(\log(n))$.

Note that since we are in Case 1, for infinitely many input lengths–input lengths $n \in \mathcal{S}''$–to R, R^{Sim}, is a circuit in $\mathcal{F}_n$, since $\mathcal{F}_n$ strongly composes. Additionally, the behavior of R^{Sim} is identical to the behavior of $R^{\{f_k\}_k}$. Moreover, since f_k succeeds with non-negligible ϵ, by assumption on R, it means that for all $n \in \mathcal{S}'$, $R^{f_{\ell(n)}}$ agrees with (Ψ, L) with probability $1/2 + 1/n^c$. But then we must have that for infinitely many $n \in \mathcal{S}'$–input lengths $n \in \mathcal{S}''$–R^{Sim} agrees with (Ψ, L) with probability $1/2 + 1/n^c$ and $R^{\mathsf{Sim}} \in \mathcal{F}_n$. So (Ψ, L) is $(\delta'(n))$-easy for $\mathcal{F}$, where

$$\delta'(n) := \begin{cases} \frac{1}{n^c} & \text{if } n \in \mathcal{S}'' \\ 0 & \text{if } n \notin \mathcal{S}'' \end{cases}$$

In Case 2, we set the tampering function $\{f_k\}_k$ to decode the query submitted by the reduction R and respond with a random encoding from the hardcore set described above (if it exists), which decodes to 0 or 1 as appropriate. Specifically, the hardcore set $\mathcal{H}$ is defined as follows: f_k sets n^* to be equal to the lexicographically first element in the (finite) set $\ell^{-1}(k) \cap \mathcal{S}''$[5], and chooses the lexicographically first set $\mathcal{H}$ of size $\epsilon'(n^*) \cdot 2^{\ell(n^*)} = \epsilon'(n^*) \cdot 2^k$ for which every adversary in $\mathcal{F}_n$ outputs $\mathsf{D}_{\ell(n^*)}(c)$ with probability at most $1/2 + \epsilon'(n^*)$, when c is chosen at random from $\mathcal{H}$. If $\ell^{-1}(k) \cap \mathcal{S}' = \emptyset$ or there is no such hardcore set $\mathcal{H}$, then f_k applies the trivial breaking strategy described above (decoding the input and responding with a random encoding of 0 or 1 as appropriate). Moreover, f_k responds only if $k \in \mathcal{S}$. Since the ϵ function remains the same in this case as well, the δ function also remains the same. Thus, for $n \in \mathcal{S}'$, $R^{f_{\ell(n)}}$ must still agree with (Ψ, L) with probability $1/2 + 1/n^c$.

In this case, as in the previous proof, we can switch to a simulated tampering function Sim that does not decode but rather chooses a random codeword from the hardcode set $\mathcal{H}$ (which again we can hardcode in using lookup circuits as before). Moreover, for queries R makes to Sim with input lengths $k = k(n) \in O(\log(n))$, all responses for all possible queries c are hardcoded into Sim. Now, for infinitely many $n \in \mathcal{S}'$–input lengths $n \in \mathcal{S}''$–R's behavior should be $t(n) \cdot \epsilon'(n)$-close

[5]Note that it is finite since $\ell^{-1}(k) \cap \mathcal{S}'$ is finite and $\mathcal{S}'' \subseteq \mathcal{S}'$.

when interacting with $\{f_k\}_k$ versus Sim, since otherwise in each hybrid step we can construct a distinguishing circuit in $\mathcal{F}_n$ (as in the previous proof) contradicting the guaranteed hardness of the hardcore set. Finally, we must argue that for infinitely many $n \in \mathcal{S}'$–input lengths $n \in \mathcal{S}''$–R composed with Sim is in the class $\mathcal{F}$. But due to the fact that $\mathcal{F}$ is $(\mathcal{F}, t)$-closed under strong composition, this occurs whenever the reduction is instantiated with security parameter $n \in \mathcal{S}''$, where n is the lexicographically first element in the set $\ell^{-1}(\ell(n)) \cap \mathcal{S}''$. Since n is always contained in $\ell^{-1}(\ell(n))$, since the size of $\ell^{-1}(\ell(n)) \cap \mathcal{S}'$ is finite and since the size of $\mathcal{S}''$ is infinite, there will be infinitely many $n \in \mathcal{S}''$ for which this event occurs. Thus, for infinitely many $n \in \mathcal{S}''$ (denote this set of values by $\widetilde{\mathcal{S}}$, $R^{\{f_k\}_k}$ agrees with (Ψ, L) with probability $1/2 + 1/n^c$ and R^{Sim} is $t(n) \cdot \epsilon'(n) \leq 1/2n^c$-close to $R^{\{f_k\}_k}$. So we conclude that (Ψ, L) is $(\delta'(n))$-easy for $\mathcal{F}$, where

$$\delta'(n) := \begin{cases} \frac{1}{2n^c} & \text{if } n \in \widetilde{\mathcal{S}} \\ 0 & \text{if } n \notin \widetilde{\mathcal{S}} \end{cases}$$

□

The following corollary holds since NC is (NC, t)-closed under strong composition and Impagliazzo's HCS holds for NC.

Corollary 12. *If for every non-negligible $\epsilon = \epsilon(\cdot)$, there is an $(\mathsf{nu}-\mathsf{NC}, \epsilon, \delta)$-black-box-reduction, for some non-negligible $\delta = \delta(\cdot)$, making $t(n)$ queries from a (single bit) non-malleable code for $\mathsf{nu}-\mathsf{NC}$, $(\mathsf{E}, \mathsf{D}) = \{(\mathsf{E}_n, \mathsf{D}_n)\}_{n=1}^{\infty}$, to a distributional problem, $(\Psi, L) = \{(\Psi_n, L_n)\}_{n=1}^{\infty}$, then (Ψ, L) is $(\delta'(n))$-easy for NC, for some non-negligible $\delta' = \delta'(\cdot)$.*

Conclusion

In this thesis, we have seen explicit constructions of non-malleable codes for a variety of computational classes. For classes where average-case unconditional lower bounds are known, we have seen that unconditionally secure non-malleable codes exist. For polynomial size circuits tampering, non-deterministic circuit lower bounds imply explicit non-malleable codes for polynomial size circuit tampering. And yet many questions remain.

Improving parameters. For many of our classes, we should be able to improve parameters. For example, we know ϵ-non-malleable codes *exist* for tampering of locality $n - \log(1/\epsilon)$ (we know they do not exist for locality $n - \log(2/\epsilon)$, but are unsure about locality between these bounds). On the other hand, we do not know even an inefficient construction that is secure for locality $\Omega(n/\log n)$. Can we achieve non-malleability against decision trees of depth n^γ for $\gamma \geq 1/4$? Similarly, can the analysis in Chapter 7 be tightened to match the parameters in [130, 98]?

In this work we did not focus on rate (so long as codeword were polynomial in the message length), but there is much room for improvement. Similarly, the error of our conditional construction for polynomial size circuit tampering is not even negligible. Ideally, we would like a code that is efficient in the *security parameter*, in the language of this thesis this means Encoding and Decoding can be performed in time $\text{poly}(\log(1/\epsilon))$ (the construction in Chapter 9 is $\text{poly}(1/\epsilon)$)). Can the error be improved? Black box impossibility results of Applebaum et al. [65] suggest this may be difficult without different assumptions.

Unconditional constructions for other classes. There are a variety of other classes for which we know strong lower bounds, but do not yet know unconditionally secure codes. To name a few: small formulas, low degree polynomials over *small* fields, branching programs with small width.

Towards a more unified theory of non-malleability? We opened this thesis with a discussion of the feasibility of proving a hardness vs. non-malleability type result. We gave both semi-generic constructions from nondeterministic circuit lower bounds (polynomial size circuits in Chapter 9) and also show obstacles to proving such a theorem from the sufficient assumption of average-case lower bounds against the class. However, it remains unclear what a minimal assumption indeed would be for non-malleability. We observe that non-malleability for small circuit tampering admits an MA proof. However, it seems likely that there is a test of this property with lower complexity.

Additionally, the separation between circuit lower bounds and non-malleability, Section 5.5, is a bit underwhelming if only for the simple reason that lower bounds typically consider boolean functions, but our classes have output equal to the input length. A relevant notion seems to be hardness of compression, but it remains to be seen if this connection can be made tight.

There are also a host of internal questions about the theory of non-malleability. For example, regarding (explicit) closure: Given an explicit NMC for $\mathcal{F}$ and an explicit NMC for $\mathcal{G}$, can the codes be generically compiled to yield a code against $\mathcal{F} \cup \mathcal{G}$ (provided one exists[6])? Similarly, given explicit NMCs for $\mathcal{F}$ and $\mathcal{G}$ can we generically compile them to get an NMC for $\mathcal{F} \circ \mathcal{G}$ (provided such a code exists)?

Do combiners exist for non-malleable codes? In particular, given a few candidate non-malleable codes for a class $\mathcal{F}$, can I combine them to get a single explicit code for the class provided that most are good.

Is domain expansion possible for natural classes? For example, given an NMC that can encode a single bit, can I get a code for even just two bits (again provided such a code exists[7]). Rate compilers [131, 32, 16], suggest that this is possible for specific classes in a certain sense.

Similarly, is hardness amplification possible? Namely, given an explicit ϵ-NMC can we generically compile it into, say, an ϵ^2-NMC (provided one exists[8])?

[6]Without this proviso, it is easy to construct counter-examples. For example $\mathcal{F}$ if n is odd and impossible if n is even, where as $\mathcal{G}$ is defined oppositely.

[7]We know from Section 3.4 that at most one bit can be protected from tampering attacks that may change up to $n - 1$ bits arbitrarily.

[8]Again, the results of Section 5.5 tell us that this is not always possible if the explicit condition is relaxed

For all of these questions, even results for specific classes would be very interesting. (Note that in all of these examples, there are uninteresting "compilers" that ignore the input codes and simply output a code with the desired properties. But, of course, this does not capture what we would like.)

New frontiers? Stepping back, non-malleable codes have a clear utility in protecting memory against tampering attacks. Moreover, we have seen in this thesis and elsewhere remarkable connections to combinatorics, pseudorandomness, complexity, and cryptography. Just like theory of error correcting codes has found broad applicability well beyond its original use-case, we believe that novel applications for these remarkable objects exist in both the theory and practice of computer science.

References

[1] M. Ball, D. Dachman-Soled, M. Kulkarni, and T. Malkin, "Limits to non-malleability," in *ITCS*, ser. LIPIcs, vol. 151, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2020, 80:1–80:32.

[2] M. Ball, D. Dachman-Soled, M. Kulkarni, and T. Malkin, "Non-malleable codes for bounded depth, bounded fan-in circuits," in *TCC*, 2016.

[3] M. Ball, S. Guo, and D. Wichs, "Non-malleable codes for decision trees," in *CRYPTO (1)*, ser. Lecture Notes in Computer Science, vol. 11692, Springer, 2019, pp. 413–434.

[4] M. Ball, D. Dachman-Soled, S. Guo, T. Malkin, and L.-Y. Tan, "Non-malleable codes for small-depth circuits," in *2018 IEEE 59th Annual Symposium on Foundations of Computer Science (FOCS)*, IEEE, 2018, pp. 826–837.

[5] M. Ball, E. Chattopadhyay, J. Liao, T. Malkin, and L. Tan, "Non-malleability against polynomial tampering," in *CRYPTO (3)*, ser. Lecture Notes in Computer Science, vol. 12172, Springer, 2020, pp. 97–126.

[6] M. Ball, D. Dachman-Soled, and J. Loss, "Explicit non-malleable codes for polynomial size circuit tampering," (unpublished manuscript).

[7] S. Dziembowski, K. Pietrzak, and D. Wichs, "Non-malleable codes," in *Innovations in Computer Science - ICS 2010, Tsinghua University, Beijing, China, January 5-7, 2010. Proceedings*, A. C. Yao, Ed., Tsinghua University Press, 2010, pp. 434–452.

[8] D. Dolev, C. Dwork, and M. Naor, "Non-malleable cryptography (extended abstract)," in *STOC*, ACM, 1991, pp. 542–552.

[9] E. Biham, "New types of cryptanalytic attacks using related keys (extended abstract)," in *EUROCRYPT*, ser. Lecture Notes in Computer Science, vol. 765, Springer, 1993, pp. 398–409.

[10] R. Gennaro, A. Lysyanskaya, T. Malkin, S. Micali, and T. Rabin, "Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering," in *TCC*, ser. Lecture Notes in Computer Science, vol. 2951, Springer, 2004, pp. 258–277.

[11] M. Cheraghchi and V. Guruswami, "Non-malleable coding against bit-wise and split-state tampering," in *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, 2014, pp. 440–464.

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

[13] D. Aggarwal, S. Dziembowski, T. Kazana, and M. Obremski, "Leakage-resilient non-malleable codes," in *TCC (1)*, ser. Lecture Notes in Computer Science, vol. 9014, Springer, 2015, pp. 398–426.

[14] E. Chattopadhyay and X. Li, "Non-malleable codes, extractors and secret sharing for interleaved tampering and composition of tampering," Cryptology ePrint Archive, Report 2018/1069, 2018., Tech. Rep., 2019.

[15] ——, "Non-malleable codes and extractors for small-depth circuits, and affine functions," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, ACM, 2017, pp. 1171–1184.

[16] D. Gupta, H. K. Maji, and M. Wang, "Explicit rate-1 non-malleable codes for local tampering," in *CRYPTO (1)*, ser. Lecture Notes in Computer Science, vol. 11692, Springer, 2019, pp. 435–466.

[17] F. Lin, M. Cheraghchi, V. Guruswami, R. Safavi-Naini, and H. Wang, "Non-malleable secret sharing against affine tampering," *arXiv preprint arXiv:1902.06195 (appeared at ITC 2020)*, 2019.

[18] M. Cheraghchi and V. Guruswami, "Capacity of non-malleable codes," in *Innovations in Theoretical Computer Science, ITCS'14, Princeton, NJ, USA, January 12-14, 2014*, 2014, pp. 155–168.

[19] S. Faust, P. Mukherjee, D. Venturi, and D. Wichs, "Efficient non-malleable codes and key-derivation for poly-size tampering circuits," in *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, 2014, pp. 111–128.

[20] M. Ball, D. Dachman-Soled, M. Kulkarni, H. Lin, and T. Malkin, "Non-malleable codes against bounded polynomial time tampering," in *EUROCRYPT (1)*, ser. Lecture Notes in Computer Science, vol. 11476, Springer, 2019, pp. 501–530.

[21] D. Dachman-Soled, I. Komargodski, and R. Pass, "Non-malleable codes for bounded polynomial depth tampering," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 776, 2020.

[22] M. Ball, D. Dachman-Soled, M. Kulkarni, and T. Malkin, "Non-malleable codes from average-case hardness: AC0, decision trees, and streaming space-bounded tampering," in *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, J. B. Nielsen and V. Rijmen, Eds., ser. Lecture Notes in Computer Science, vol. 10822, Springer, 2018, pp. 618–650.

[23] A. A. Razborov, "Lower bounds on the size of bounded depth circuits over a complete basis with logical addition," *Mathematical Notes of the Academy of Sciences of the USSR*, vol. 41, no. 4, pp. 333–338, 1987.

[24] R. Smolensky, "Algebraic methods in the theory of lower bounds for boolean circuit complexity," in *Proceedings of the nineteenth annual ACM symposium on Theory of computing*, 1987, pp. 77–82.

[25] ——, "On representations by low-degree polynomials," in *Proceedings of 1993 IEEE 34th Annual Foundations of Computer Science*, IEEE, 1993, pp. 130–138.

[26] R. Zippel, "Probabilistic algorithms for sparse polynomials," in *International Symposium on Symbolic and Algebraic Manipulation*, Springer, 1979, pp. 216–226.

[27] S. Faust, P. Mukherjee, J. B. Nielsen, and D. Venturi, "Continuous non-malleable codes," in *Theory of Cryptography - 11th Theory of Cryptography Conference, TCC 2014, San Diego, CA, USA, February 24-26, 2014. Proceedings*, 2014, pp. 465–488.

[28] D. Dachman-Soled, F. Liu, E. Shi, and H. Zhou, "Locally decodable and updatable non-malleable codes and their applications," in *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, 2015, pp. 427–450.

[29] E. Chattopadhyay, V. Goyal, and X. Li, "Non-malleable extractors and codes, with their many tampered extensions," in *Proceedings of the 48th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2016, Cambridge, MA, USA, June 18-21, 2016*, 2016, pp. 285–298.

[30] N. Chandran, V. Goyal, P. Mukherjee, O. Pandey, and J. Upadhyay, "Block-wise non-malleable codes," in *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, 2016, 31:1–31:14.

[31] F. Liu and A. Lysyanskaya, "Tamper and leakage resilience in the split-state model," in *Advances in Cryptology - CRYPTO 2012 - 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, August 19-23, 2012. Proceedings*, 2012, pp. 517–532.

[32] D. Aggarwal, S. Agrawal, D. Gupta, H. K. Maji, O. Pandey, and M. Prabhakaran, "Optimal computational split-state non-malleable codes," in *TCC (A2)*, ser. Lecture Notes in Computer Science, vol. 9563, Springer, 2016, pp. 393–417.

[33] A. Kiayias, F. Liu, and Y. Tselekounis, "Practical non-malleable codes from l-more extractable hash functions," in *Proceedings of the 2016 ACM SIGSAC Conference on Com-*

puter and Communications Security, Vienna, Austria, October 24-28, 2016, 2016, pp. 1317–1328.

[34] D. Aggarwal, Y. Dodis, and S. Lovett, "Non-malleable codes from additive combinatorics," in *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, D. B. Shmoys, Ed., ACM, 2014, pp. 774–783.

[35] X. Li, "Improved non-malleable extractors, non-malleable codes and independent source extractors," in *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017*, H. Hatami, P. McKenzie, and V. King, Eds., ACM, 2017, pp. 1144–1156.

[36] D. Aggarwal and M. Obremski, "A constant-rate non-malleable code in the split-state model," *IACR Cryptology ePrint Archive*, vol. 2019, p. 1299, 2019.

[37] E. Chattopadhyay and D. Zuckerman, "Non-malleable codes against constant split-state tampering," in *Proceedings of the 55th Annual IEEE Symposium on Foundations of Computer Science*, 2014, pp. 306–315.

[38] B. Kanukurthi, L. Obbattu, and S. Sekar, *Four-state non-malleable codes with explicit constant rate*, Theory of Cryptography - 15th Theory of Cryptography Conference, TCC 2017, Baltimore, MD, USA, November 12-15, 2017, to appear, 2014.

[39] S. Agrawal, D. Gupta, H. K. Maji, O. Pandey, and M. Prabhakaran, "Explicit non-malleable codes against bit-wise tampering and permutations," in *Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I*, 2015, pp. 538–557.

[40] M. Cheraghchi and V. Guruswami, "Capacity of non-malleable codes," *IEEE Trans. Information Theory*, vol. 62, no. 3, pp. 1097–1118, 2016.

[41] S. Faust, K. Hostáková, P. Mukherjee, and D. Venturi, "Non-malleable codes for space-bounded tampering," in *Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II*, 2017, pp. 95–126.

[42] S. Dziembowski, K. Pietrzak, and D. Wichs, "Non-malleable codes," *J. ACM*, vol. 65, no. 4, 20:1–20:32, Apr. 2018, Extended abstract appeared in Innovations in Computer Science (ICS) 2010.

[43] S. Dziembowski, T. Kazana, and M. Obremski, "Non-malleable codes from two-source extractors," in *CRYPTO (2)*, ser. Lecture Notes in Computer Science, vol. 8043, Springer, 2013, pp. 239–257.

[44] D. Aggarwal, Y. Dodis, T. Kazana, and M. Obremski, "Non-malleable reductions and applications," in *Proceedings of the Forty-Seventh Annual ACM on Symposium on Theory of Computing, STOC 2015, Portland, OR, USA, June 14-17, 2015*, R. A. Servedio and R. Rubinfeld, Eds., ACM, 2015, pp. 459–468.

[45] N. Nisan and D. Zuckerman, "Randomness is linear in space," *J. Comput. Syst. Sci.*, vol. 52, no. 1, pp. 43–52, 1996.

[46] V. Guruswami, C. Umans, and S. P. Vadhan, "Unbalanced expanders and randomness extractors from Parvaresh–Vardy codes," *J. ACM*, vol. 56, no. 4, 2009.

[47] Z. Dvir, S. Kopparty, S. Saraf, and M. Sudan, "Extensions to the method of multiplicities, with applications to Kakeya sets and mergers," in *Proceedings of the 50th Annual IEEE Symposium on Foundations of Computer Science*, 2009, pp. 181–190.

[48] B. Chor and O. Goldreich, "Unbiased bits from sources of weak randomness and probabilistic communication complexity," vol. 17, no. 2, pp. 230–261, 1988.

[49] J. Bourgain, "More on the sum-product phenomenon in prime fields and its applications," *International Journal of Number Theory*, vol. 01, no. 01, pp. 1–32, 2005. eprint: `http://www.worldscientific.com/doi/pdf/10.1142/S1793042105000108`.

[50] B. Barak, R. Impagliazzo, and A. Wigderson, "Extracting randomness using few independent sources," *SIAM J. Comput.*, vol. 36, no. 4, pp. 1095–1118, Dec. 2006.

[51] E. Chattopadhyay and D. Zuckerman, "Explicit two-source extractors and resilient functions," *Annals of Mathematics*, vol. 189, no. 3, pp. 653–705, 2019.

[52] J. Bourgain, "On the construction of affine extractors," *GAFA Geometric And Functional Analysis*, vol. 17, no. 1, pp. 33–57, 2007.

[53] A. Gabizon and R. Raz, "Deterministic extractors for affine sources over large fields," *Combinatorica*, vol. 28, no. 4, pp. 415–440, 2008.

[54] Z. Dvir, "Extractors for varieties," *Computational complexity*, vol. 21, no. 4, pp. 515–572, 2012.

[55] B. Chor, O. Goldreich, J. Hasted, J. Freidmann, S. Rudich, and R. Smolensky, "The bit extraction problem or t-resilient functions," in *IEEE Symposium on Foundations of Computer Science*, 1985, pp. 396–407.

[56] D. Zuckerman, "Linear degree extractors and the inapproximability of max clique and chromatic number," in *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, 2006, pp. 681–690.

[57] A. Ta-Shma and D. Zuckerman, "Extractor codes," *IEEE Transactions on Information Theory*, vol. 50, no. 12, pp. 3015–3025, 2004.

[58] C. Bennett, G. Brassard, and J.-M. Robert, "Privacy amplification by public discussion," vol. 17, pp. 210–229, 1988.

[59] C.-J. Lu, "Hyper-encryption against space-bounded adversaries from on-line strong extractors," in *Annual International Cryptology Conference*, Springer, 2002, pp. 257–271.

[60] Y. Dodis and D. Wichs, "Non-malleable extractors and symmetric key cryptography from weak secrets," in *STOC*, 2009, pp. 601–610.

[61] E. Chattopadhyay, V. Goyal, and X. Li, "Non-malleable extractors and codes, with their many tampered extensions," in *STOC*, 2016.

[62] D. Aggarwal, Y. Dodis, and S. Lovett, "Non-malleable codes from additive combinatorics," *SIAM J. Comput.*, vol. 47, no. 2, pp. 524–546, 2018.

[63] S. Goldwasser and M. Sipser, "Private coins versus public coins in interactive proof systems," in *STOC*, ACM, 1986, pp. 59–68.

[64] L. Babai and S. Moran, "Arthur-merlin games: A randomized proof system, and a hierarchy of complexity classes," *J. Comput. Syst. Sci.*, vol. 36, no. 2, pp. 254–276, 1988.

[65] B. Applebaum, S. Artemenko, R. Shaltiel, and G. Yang, "Incompressible functions, relative-error extractors, and the power of nondeterministic reductions," *Comput. Complex.*, vol. 25, no. 2, pp. 349–418, 2016.

[66] R. Impagliazzo, "Hard-core distributions for somewhat hard problems," in *FOCS*, IEEE Computer Society, 1995, pp. 538–545.

[67] S. G. Choi, D. Dachman-Soled, T. Malkin, and H. Wee, "Black-box construction of a non-malleable encryption scheme from any semantically secure one," in *Theory of Cryptography, Fifth Theory of Cryptography Conference, TCC 2008, New York, USA, March 19-21, 2008.*, R. Canetti, Ed., ser. Lecture Notes in Computer Science, vol. 4948, Springer, 2008, pp. 427–444.

[68] ——, "A black-box construction of non-malleable encryption from semantically secure encryption," *IACR Cryptology ePrint Archive*, vol. 2016, p. 720, 2016.

[69] S. E. Decatur, O. Goldreich, and D. Ron, "Computational sample complexity," in *COLT*, ACM, 1997, pp. 130–142.

[70] ——, "A probabilistic error-correcting scheme that provides partial secrecy," in *Computational Complexity and Property Testing*, ser. Lecture Notes in Computer Science, vol. 12050, Springer, 2020, pp. 1–8.

[71] L. Carter and M. N. Wegman, "Universal classes of hash functions (extended abstract)," in *STOC*, ACM, 1977, pp. 106–112.

[72] A. Weil, "On some exponential sums," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 34, no. 5, p. 204, 1948.

[73] A. Rao, "An exposition of Bourgain's 2-source extractor," in *Electronic Colloquium on Computational Complexity (ECCC)*, vol. 14, 2007.

[74] Y. Dodis, X. Li, T. D. Wooley, and D. Zuckerman, "Privacy amplification and nonmalleable extractors via character sums," *SIAM Journal on Computing*, vol. 43, no. 2, pp. 800–830, 2014.

[75] N. Nisan, "Pseudorandom generators for space-bounded computation," *Combinatorica*, vol. 12, no. 4, pp. 449–461, 1992.

[76] L. M. J. Bazzi, "Polylogarithmic independence can fool DNF formulas," *SIAM J. Comput.*, vol. 38, no. 6, pp. 2220–2272, 2009.

[77] A. Razborov, "A simple proof of bazzi's theorem," *ACM Transactions on Computation Theory (TOCT)*, vol. 1, no. 1, p. 3, 2009.

[78] L. Trevisan and T. Xue, "A derandomized switching lemma and an improved derandomization of AC0," in *Proceedings of the 28th Conference on Computational Complexity, CCC 2013, K.lo Alto, California, USA, 5-7 June, 2013*, IEEE Computer Society, 2013, pp. 242–247.

[79] A. Panconesi and A. Srinivasan, "Randomized distributed edge coloring via an extension of the chernoff-hoeffding bounds," *SIAM J. Comput.*, vol. 26, no. 2, pp. 350–368, 1997.

[80] M. Bellare and J. Rompel, "Randomness-efficient oblivious sampling," in *35th Annual Symposium on Foundations of Computer Science, Santa Fe, New Mexico, USA, 20-22 November 1994*, IEEE Computer Society, 1994, pp. 276–287.

[81] J. P. Schmidt, A. Siegel, and A. Srinivasan, "Chernoff-hoeffding bounds for applications with limited independence," *SIAM J. Discrete Math.*, vol. 8, no. 2, pp. 223–250, 1995.

[82] Z. Dvir, A. Gabizon, and A. Wigderson, "Extractors and rank extractors for polynomial sources," *Computational Complexity*, vol. 18, no. 1, pp. 1–58, 2009.

[83] J. T. Schwartz, "Probabilistic algorithms for verification of polynomial identities," in *International Symposium on Symbolic and Algebraic Manipulation*, Springer, 1979, pp. 200–215.

[84] S. P. Vadhan, "A study of statistical zero-knowledge proofs," PhD thesis, Massachusetts Institute of Technology, 1999.

[85] E. Chattopadhyay and X. Li, "Non-malleable extractors and codes in the interleaved split-state model and more," *CoRR*, vol. abs/1804.05228, 2018. arXiv: 1804.05228.

[86] S. Dziembowski, "On forward-secure storage," in *Advances in Cryptology - CRYPTO 2006, 26th Annual International Cryptology Conference, Santa Barbara, California, USA, August 20-24, 2006, Proceedings*, 2006, pp. 251–270.

[87] F. Davì, S. Dziembowski, and D. Venturi, "Leakage-resilient storage," in *SCN*, ser. Lecture Notes in Computer Science, vol. 6280, Springer, 2010, pp. 121–137.

[88] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson, "Multi-prover interactive proofs: How to remove intractability assumptions," in *STOC*, ACM, 1988, pp. 113–131.

[89] H. Buhrman and R. de Wolf, "Complexity measures and decision tree complexity: A survey," *Theor. Comput. Sci.*, vol. 288, no. 1, pp. 21–43, 2002.

[90] M. Ball, D. Dachman-Soled, S. Guo, T. Malkin, and L. Tan, "Non-malleable codes for small-depth circuits," in *FOCS*, IEEE Computer Society, 2018, pp. 826–837.

[91] M. M. Klawe, W. J. Paul, N. Pippenger, and M. Yannakakis, "On monotone formulae with restricted depth (preliminary version)," in *Proceedings of the 16th Annual ACM Symposium on Theory of Computing, April 30 - May 2, 1984, Washington, DC, USA*, R. A. DeMillo, Ed., ACM, 1984, pp. 480–487.

[92] L. G. Valiant, "Exponential lower bounds for restricted monotone circuits," in *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, D. S. Johnson, R. Fagin, M. L. Fredman, D. Harel, R. M. Karp, N. A. Lynch, C. H. Papadimitriou, R. L. Rivest, W. L. Ruzzo, and J. I. Seiferas, Eds., ACM, 1983, pp. 110–117.

[93] M. L. Furst, J. B. Saxe, and M. Sipser, "Parity, circuits, and the polynomial-time hierarchy," *Mathematical Systems Theory*, vol. 17, no. 1, pp. 13–27, 1984.

[94] M. Ajtai, "First-order definability on finite structures," *Ann. Pure Appl. Logic*, vol. 45, no. 3, pp. 211–225, 1989.

[95] A. C. Yao, "Separating the polynomial-time hierarchy by oracles (preliminary version)," in *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, 1985, pp. 1–10.

[96] M. Ajtai and A. Wigderson, "Deterministic simulation of probabilistic constant depth circuits (preliminary version)," in *26th Annual Symposium on Foundations of Computer Science, Portland, Oregon, USA, 21-23 October 1985*, 1985, pp. 11–19.

[97] M. Agrawal, E. Allender, R. Impagliazzo, T. Pitassi, and S. Rudich, "Reducing the complexity of reductions," *Computational Complexity*, vol. 10, no. 2, pp. 117–138, 2001.

[98] R. Impagliazzo, W. Matthews, and R. Paturi, "A satisfiability algorithm for AC0," in *Proceedings of the Twenty-Third Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2012, Kyoto, Japan, January 17-19, 2012*, 2012, pp. 961–972.

[99] P. Gopalan, R. Meka, and O. Reingold, "DNF sparsification and a faster deterministic counting algorithm," *Computational Complexity*, vol. 22, no. 2, pp. 275–310, 2013.

[100] O. Goldreich and A. Wigderson, "On derandomizing algorithms that err extremely rarely," in *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, 2014, pp. 109–118.

[101] M. Ball, E. Chattopadhyay, J. Liao, T. Malkin, and L. Tan, "Non-malleability against polynomial tampering," *IACR Cryptol. ePrint Arch.*, vol. 2020, p. 147, 2020.

[102] M. Cheraghchi and A. Shokrollahi, "Almost-uniform sampling of points on high-dimensional algebraic varieties," in *26th International Symposium on Theoretical Aspects of Computer Science, STACS 2009, February 26-28, 2009, Freiburg, Germany, Proceedings*, 2009, pp. 277–288.

[103] M. A. Huang and Y. Wong, "An algorithm for approximate counting of points on algebraic sets over finite fields," in *ANTS*, ser. Lecture Notes in Computer Science, vol. 1423, Springer, 1998, pp. 514–527.

[104] P. Dusart, "Estimates of some functions over primes without RH," *arXiv preprint arXiv:1002.0442*, 2010.

[105] B. Rosser, "The n-th prime is greater than nlogn," *Proceedings of the London Mathematical Society*, vol. 2, no. 1, pp. 21–44, 1939.

[106] G. Robin, "Permanence de relations de récurrence dans certains développements asymptotiques," *Pub. Inst. Math. Beograd*, vol. 43, no. 57, pp. 17–25, 1988.

[107] M. O. Rabin, "Probabilistic algorithms in finite fields," *SIAM J. Comput.*, vol. 9, no. 2, pp. 273–280, 1980.

[108] B. Barak, S. J. Ong, and S. P. Vadhan, "Derandomization in cryptography," in *CRYPTO*, ser. Lecture Notes in Computer Science, vol. 2729, Springer, 2003, pp. 299–315.

[109] A. Drucker, "Nondeterministic direct product reductions and the success probability of SAT solvers," in *FOCS*, IEEE Computer Society, 2013, pp. 736–745.

[110] U. Feige and C. Lund, "On the hardness of computing the permanent of random matrices," *Comput. Complex.*, vol. 6, no. 2, pp. 101–132, 1997.

[111] O. Goldreich and A. Wigderson, "Derandomization that is rarely wrong from short advice that is typically good," in *RANDOM*, ser. Lecture Notes in Computer Science, vol. 2483, Springer, 2002, pp. 209–223.

[112] D. Gutfreund, R. Shaltiel, and A. Ta-Shma, "Uniform hardness versus randomness tradeoffs for arthur-merlin games," *Comput. Complex.*, vol. 12, no. 3-4, pp. 85–130, 2003.

[113] A. R. Klivans and D. van Melkebeek, "Graph nonisomorphism has subexponential size proofs unless the polynomial-time hierarchy collapses," *SIAM J. Comput.*, vol. 31, no. 5, pp. 1501–1526, 2002.

[114] P. B. Miltersen and N. V. Vinodchandran, "Derandomizing arthur-merlin games using hitting sets," *Comput. Complex.*, vol. 14, no. 3, pp. 256–279, 2005.

[115] R. Shaltiel and C. Umans, "Simple extractors for all min-entropies and a new pseudorandom generator," *J. ACM*, vol. 52, no. 2, pp. 172–216, 2005.

[116] ——, "Pseudorandomness for approximate counting and sampling," *Comput. Complex.*, vol. 15, no. 4, pp. 298–341, 2006.

[117] ——, "Low-end uniform hardness versus randomness tradeoffs for AM," *SIAM J. Comput.*, vol. 39, no. 3, pp. 1006–1037, 2009.

[118] L. Trevisan and S. P. Vadhan, "Extracting randomness from samplable distributions," in *FOCS*, IEEE Computer Society, 2000, pp. 32–42.

[119] J. Kinne, D. van Melkebeek, and R. Shaltiel, "Pseudorandom generators, typically-correct derandomization, and circuit lower bounds," *Comput. Complex.*, vol. 21, no. 1, pp. 3–61, 2012.

[120] F. Li and D. Zuckerman, "Improved extractors for recognizable and algebraic sources," in *APPROX-RANDOM*, ser. LIPIcs, vol. 145, Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2019, 72:1–72:22.

[121] R. Impagliazzo and A. Wigderson, "$P = BPP$ if E requires exponential circuits: Derandomizing the XOR lemma," in *STOC*, ACM, 1997, pp. 220–229.

[122] O. Goldreich, S. Micali, and A. Wigderson, "Proofs that yield nothing but their validity for all languages in NP have zero-knowledge proof systems," *J. ACM*, vol. 38, no. 3, pp. 691–729, 1991.

[123] B. Applebaum, Y. Ishai, and E. Kushilevitz, "Cryptography in nc^0," in *FOCS*, IEEE Computer Society, 2004, pp. 166–175.

[124] N. Bitansky, S. Goldwasser, A. Jain, O. Paneth, V. Vaikuntanathan, and B. Waters, "Time-lock puzzles from randomized encodings," in *ITCS*, ACM, 2016, pp. 345–356.

[125] A. Degwekar, V. Vaikuntanathan, and P. N. Vasudevan, "Fine-grained cryptography," in *CRYPTO (3)*, ser. Lecture Notes in Computer Science, vol. 9816, Springer, 2016, pp. 533–562.

[126] B. Applebaum and P. Raykov, "On the relationship between statistical zero-knowledge and statistical randomized encodings," in *CRYPTO (3)*, ser. Lecture Notes in Computer Science, vol. 9816, Springer, 2016, pp. 449–477.

[127] O. Goldreich, S. Goldwasser, and S. Micali, "How to construct random functions (extended abstract)," in *FOCS*, IEEE Computer Society, 1984, pp. 464–479.

[128] O. Reingold, L. Trevisan, and S. P. Vadhan, "Notions of reducibility between cryptographic primitives," in *TCC*, ser. Lecture Notes in Computer Science, vol. 2951, Springer, 2004, pp. 1–20.

[129] M. Ajtai, J. Komlós, and E. Szemerédi, "An o(n log n) sorting network," in *Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 25-27 April, 1983, Boston, Massachusetts, USA*, D. S. Johnson, R. Fagin, M. L. Fredman, D. Harel, R. M. Karp, N. A. Lynch, C. H. Papadimitriou, R. L. Rivest, W. L. Ruzzo, and J. I. Seiferas, Eds., ACM, 1983, pp. 1–9.

[130] J. Håstad, "On the correlation of parity and small-depth circuits," *SIAM J. Comput.*, vol. 43, no. 5, pp. 1699–1708, 2014.

[131] S. Agrawal, D. Gupta, H. K. Maji, O. Pandey, and M. Prabhakaran, "A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations," in *Theory of Cryptography - 12th Theory of Cryptography Conference, TCC 2015, Warsaw, Poland, March 23-25, 2015, Proceedings, Part I*, 2015, pp. 375–397.