ABSTRACT

Title of dissertation: EXTENDING THE APPLICABILITY OF NON-MALLEABLE CODES

Mukul Ramesh Kulkarni
Doctor of Philosophy, 2019

Dissertation directed by: Professor Dana Dachman-Soled
Department of Electrical and Computer Engineering

Modern cryptographic systems provide provable security guarantees as long as secret keys of the system remain confidential. However, if adversary learns some bits of information about the secret keys the security of the system can be breached. Side-channel attacks (like power analysis, timing analysis etc.) are one of the most effective tools employed by the adversaries to learn information pertaining to cryptographic secret keys.An adversary can also tamper with secret keys (say flip some bits) and observe the modified behavior of the cryptosystem, thereby leaking information about the secret keys. Dziembowski et al. (JACM 2018) defined the notion of non-malleable codes, a tool to protect memory against tampering. Non-malleable codes ensure that, when a codeword (generated by encoding an underlying message) is modified by some tampering function in a given tampering class, if the decoding of tampered codeword is incorrect then the decoded message is independent of the original message.

In this dissertation, we focus on improving different aspects of non-malleable codes. Specifically, (1) we extend the class of tampering functions and present explicit constructions as well as general frameworks for constructing non-malleable codes. While most prior work considered "compartmentalized" tampering functions, which modify parts of the codeword independently, we consider classes of tampering functions which can tamper with the entire codeword but are restricted in computational complexity. The tampering classes studied in this work include complexity classes NC^0, and AC^0. Also, earlier works focused on constructing non-malleable codes from scratch for different tampering classes, in this work we present a general framework for constructing non-malleable codes based on average-case hard problems for specific tampering families, and we instantiate our framework for various tampering classes including AC^0. (2) The locality of code is the number of codeword blocks required to be accessed in order to decode/update a single block in the underlying message. We improve efficiency and usability by studying the optimal locality of non-malleable codes. We show that locally decodable and updatable non-malleable codes cannot have constant locality. We also give a matching upper bound that improves the locality of previous constructions. (3) We investigate a stronger variant of non-malleable codes called continuous non-malleable codes, which are known to be impossible to construct without computational assumptions. We show that setup assumptions such as common reference string (CRS) are also necessary to construct this stronger primitive. We present construction of continuous non-malleable codes in CRS model from weaker computational assumptions than assumptions used in prior work.

EXTENDING THE APPLICABILITY OF NON-MALLEABLE CODES

by

Mukul Ramesh Kulkarni

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2019

Dedication

To my grandparents *Late Dinkar Keshav Raje & Late Pramila Dinkar Raje* and my father *Late Ramesh Dwarkanath Kulkarni*, who I know will be beaming with pride.

Acknowledgments

I am grateful to everybody who have made this thesis possible and to all who have made my graduate experience memorable and worth cherishing.

First and foremost I wish to thank my advisor, Professor Dana Dachman-Soled for trusting my abilities and providing me an invaluable opportunity to work on engaging and intellectually stimulating projects throughout the past five years. Her mentoring has taught me how to be a critical thinker, dedicated researcher, and how to communicate my ideas effectively. Throughout the time we worked together, she has always shared her invaluable insights, shown unwavering support, and encouraged me to produce high quality research. I am honored and consider myself fortunate to work with and learn from her.

I would also like to thank, Professor Tal Malkin. Who has been amazing collaborator, and guide to me. Without her extraordinary ideas and expertise, this thesis would have been a distant dream. Thanks are due to Professor Jonathan Katz, Professor Charalampos (Babis) Papamanthou and Professor Gang Qu for agreeing to serve on my thesis committee and for sparing their invaluable time reviewing the manuscript.

My collaborators Marshall Ball, Aria Shahverdi, Huijing Gong, Professor Huijia (Rachel) Lin, Professor Tudor Dumitras, Dr. Kartik Nayak, Soumya Indela deserve a special acknowledgement since without their hard work and support it would have been incredibly difficult if not impossible to explore different research topics.

I also wish to thank my colleagues at Maryland Cybersecurity Center who enriched my graduate life. Dr. Alex Malozemoff, Dr. Aishwarya Thiryvengadam, and Dr. Daniel Apon deserve a special mention for helping me settle during the initial days. I am thankful to Professor Arkady Yerukhimovich, Professor Seung-Geol Choi, Dr. Xiao Wang, Professor Bhavana Kanukurthi, Dr. Nishanth Chandran, Sruthi Sekar, Sai Lakshmi Bhavana Obbattu, and Dr. Samuel Ranellucci for many fruitful discussions.

I am extremely grateful to Melanie Prange, William (Bill) Churma, Emily Irwin, and Dana Purcell for their continued support and assistance to ensure that I can focus on research without having to worry about any other administrative work.

I thank Professor Ankur Srivastava and other faculty members of ECE department who have always provided their guidance and expertise whenever I requested.

I owe my deepest thanks to my family - my mother and sister who have always believed in me and have sacrificed more than one can imagine to make my dream possible. Words cannot express the gratitude I owe them. My wife, Devika, deserves special thanks for her endless love, company, and patience. I am extremely fortunate to have her in my life. I would also like to thank my brother-in-law Shailesh, and my in-laws for their never ending support and encouragement.

My friends have been a crucial factor in my well being, have looked after me and have accompanied me throughout this long journey which otherwise would have been lonely. I'd like to express my gratitude to Matthew Reed, Helene Nguewou, Vidya Raju, Swaminathan Sankarnarayanan, Ariyan Kabir, Sarah Al-Hussaini, Raji

Samaraweera, Estefany Carrillo, Nidhi Rathi, Akshay & Papeeha Deshpande, Amit & Girija Kulkarni, Gunesh & Priyanka Patil, Tejas & Anuja Kulkarni, Pratiksha & Radhakrishna Tetali, Sujay Bhatt, Siddhi Shinde, Aditya Mishra, Shweta Vinod, Aditi Joshi Bhavthankar, Kiran & Abhitosh Singh for their friendship and support.

I believe all the ups and downs experienced in graduate school will make me a better person, son, and husband.

It is impossible to remember all, and I apologize to those I've inadvertently left out.

Thank you all and thank God!

Table of Contents

Acknowledgements iii

Table of Contents vi

List of Tables x

List of Figures xi

List of Abbreviations xiii

1 Introduction 1
- 1.1 Organization . . . 7
- 1.2 Non-Malleable Codes for Bounded Depth, Bounded Fan-in Circuits . 7
- 1.3 Non-Malleable Codes from Average-Case Hardness: AC0, Decision Trees, and Streaming Space-Bounded Tampering . . . 8
- 1.4 Tight Upper and Lower Bounds for Leakage-Resilient, Locally Decodable and Updatable Non-Malleable Codes . . . 9
- 1.5 Upper and Lower Bounds for Continuous Non-Malleable Codes . . . 12

2 Related Work 15
- 2.1 Non-Malleable Codes . . . 15
- 2.2 Locally Decodable and Updatable Non-Malleable Codes . . . 20
- 2.3 Continuous Non-Malleable Codes . . . 22

3 Preliminaries and Definitions 26
- 3.1 Notation . . . 26
- 3.2 Non-Malleable Codes . . . 28
  - 3.2.1 Non-Malleable Code in CRS Model . . . 31
  - 3.2.2 Medium Non-Malleability . . . 33
  - 3.2.3 Continuous Non-Malleable Codes (CNMC) . . . 36
  - 3.2.4 Locally Decodable and Updatable Non-Malleable Codes (LDUNMC) . . . 38
- 3.3 Boolean Circuits and Related Definitions . . . 42
  - 3.3.1 Local Functions . . . 43
  - 3.3.2 Background on Boolean Analysis . . . 45

3.4 Public Key Encryption Scheme and PRG 47
3.4.1 Pseudorandom Generators of Space-Bounded Computation . . 49
3.5 Non-Interactive Zero Knowledge 50

4 Non-Malleable Codes for Bounded Depth, Bounded Fan-in Circuits 55
4.1 Introduction 55
4.1.1 Technical Overview 61
4.2 Preliminaries 68
4.2.1 Non-Malleable Codes: Alternate Definitions 68
4.2.2 Tampering Families 70
4.2.3 Reconstructable Probabilistic Encoding Scheme 70
4.3 Non-malleable Codes for $\mathrm{Local}^{\ell_i(n)}_{\ell_o(n)}$ 72
4.3.1 Extending to Leaky Local 83
4.4 Extending to $\mathrm{Local}^{m(n)}$ 85
4.5 Achieving Resilience against $o(n/\log n)$ Output Locality 91

5 Non-Malleable Codes from Average-Case Hardness:
AC^0, Decision Trees, and Streaming Space-Bounded Tampering 107
5.1 Introduction 107
5.1.1 Technical Overview 110
5.2 Definitions 121
5.2.1 Incompressible Functions 121
5.2.2 Proof Systems for Circuit SAT 121
5.2.2.1 Circuit SAT proof system for the class $\mathcal{L}(\mathcal{C})$ with prover complexity $\mathcal{D}$ and verifier complexity AC^0 . . . 122
5.2.2.2 Circuit SAT proof system for the class $\mathcal{L}(\mathcal{C})$ with prover complexity $\mathcal{D}$ and streaming verifier. 123
5.2.3 Computational Model for Streaming Adversaries 125
5.3 Generic Construction for One-Bit Messages 129
5.3.1 Proof of Theorem 5.3.1 135
5.3.2 One-Bit NMC for AC^0 and beyond 137
5.3.3 Tampering classes beyond AC^0 149
5.4 Construction for Multi-Bit Messages 152
5.4.1 Generic Analysis 158
5.4.2 Efficient, Multi-Bit NMC for AC^0 160
5.4.3 Tampering with decision trees 170
5.5 One-Bit NMC Against Streaming Adversaries 172
5.5.1 The Hard Distribution D_b (parameter n) 173
5.5.2 Encryption scheme $\mathcal{E} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ (parameter $n' \in \omega(n)$) 173
5.5.3 Weak Encryption Scheme (parameter $n'' \in o(n)$) 174

5.5.4 Non-Interactive Simulatable Proof System (parameter $n'' \in o(n)$) . . . 174
5.5.5 Multi-Bit NMC Against Streaming Adversaries . . . 189
5.6 Figure to explain MPC in head from section 5.5.4 . . . 191

6 Tight Upper and Lower Bounds for Leakage-Resilient, Locally Decodable and Updatable Non-Malleable Codes 193
6.1 Introduction . . . 193
6.1.1 Our Results . . . 196
6.1.2 Our Techniques . . . 198
6.1.2.1 Lower Bound for Deterministic Access Patterns . . . 198
6.1.2.2 Lower Bound for Randomized Access Patterns . . . 203
6.1.2.3 Difficulty of Extending Techniques to the Adaptive Setting . . . 206
6.1.2.4 Upper Bound . . . 207
6.2 Lower Bound . . . 209
6.2.1 Attack Preliminaries . . . 209
6.2.2 Applying the Sunflower Lemma . . . 211
6.2.3 The Compression Functions . . . 212
6.2.4 Closeness of Distributions . . . 212
6.2.5 The Attack . . . 214
6.2.6 Attack Analysis . . . 215
6.3 Extending Lower Bound to Randomized Decode/Update . . . 222
6.3.1 Applying the Sunflower Lemma . . . 223
6.3.2 The Compression Functions . . . 224
6.3.3 Closeness of Distributions . . . 225
6.3.4 The Attack . . . 229
6.3.5 Attack Analysis . . . 231
6.4 Matching Upper Bound . . . 246

7 Upper and Lower Bounds for Continuous Non-Malleable Codes 258
7.1 Introduction . . . 258
7.1.1 Our Results . . . 260
7.1.2 Technical Overview . . . 263
7.2 Preliminaries . . . 270
7.2.1 Randomness Extractors . . . 270
7.2.2 Continuous Non-Malleable Randomness Encoder . . . 270
7.2.3 Falsifiable Assumptions and Black-Box Reductions . . . 272
7.2.4 (Strong) One-Time Signature Schemes . . . 273
7.2.5 Equivocal Commitment Scheme . . . 273
7.2.6 One-to-one Equivocal Commitment . . . 277

7.2.7 Equivocal Commitment (with extra properties) in the CRS model . . . 278
7.3 Impossibility of CNMC with no CRS . . . 281
7.4 2-State CNMC for One-Bit Messages . . . 285
7.4.1 CNM Randomness Encoder to Single-Bit CNMC . . . 286
7.5 4-State CNMC for Multi-Bit Messages . . . 297

8 Conclusion and Future Directions 312
8.1 Future Directions . . . 314

Bibliography 315

List of Tables

List of Figures

3.1 Non-Malleability Experiment $\mathsf{Tamper}^{\Pi,\mathcal{F}}_{A,m}(n)$ 31
3.2 Non-Malleability Experiment $\mathbf{Ideal}_{\mathsf{Sim},m}(n)$ 32
3.3 Strong Non-Malleability Experiment $\mathsf{StrongTamper}^{\Pi,\mathcal{F}}_{A,m}(n)$ 32
3.4 The experiment corresponding to the special predicate g. 34
3.5 Medium Non-Malleability Experiment $\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,m,g}(n)$ 34

4.1 Figure with caption indented 75
4.2 The adversary chooses tampering function $f = (f^{\mathsf{L}}, f^{\mathsf{R}}) \in \mathrm{Local}^{d_o(k)}_{d_i(k)}$ 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 $\mathcal{S}_{\mathsf{L}\to\mathsf{R}}$. 76
4.3 The $(\mathrm{Local}^{d_o{}'}, \mathrm{LL}, \mathsf{negl}(n))$-non-malleable reduction $(\mathsf{E}', \mathsf{D}')$ . . 87
4.4 The $(\mathrm{Local}^{d_o}, \mathrm{SS}, \mathsf{negl}(k))$-non-malleable reduction (E, D) . . . 93

5.1 Non-malleable code $(\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$, secure against $\mathcal{F}$ tampering. 130
5.2 Encoding algorithm with simulated proofs. 131
5.3 Encoding algorithm with simulated proofs and encryptions. 131
5.4 Extracting procedure Ext. 132
5.5 Alternate decoding procedure D', given additional extracted key k as input. 132
5.6 The predicate $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r)$. 132
5.7 Non-malleable code $(\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$, secure against $\mathcal{F}$ tampering. 153
5.8 Encoding algorithm with simulated proofs. 154
5.9 Encoding algorithm with simulated proofs and encryptions. 155
5.10 Extracting procedure Ext. 155

5.11 ALTERNATE DECODING PROCEDURE D', GIVEN ADDITIONAL EXTRACTED KEY $[k^i]_{i\in[m]}$ AS INPUT. 155
5.12 THE PREDICATE $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r)$. 156
5.13 **A pictorial representation of the Prover's output in the NI Simulatable Proof System Π.** Let $\ell = 5$ be the number of parties, and λ be the security parameter. In the q-th iteration, each party P_i for $i \in [\ell]$ has inputs (w_i^q, s_i^q). We encode each s_i^q as $s_i^q||\overline{s}_i^q$, where $\overline{s}_i^q$ is the bit-wise complement of s_i^q. For example 001 is encoded as 001100. For each bit of the encoding of s_i^q, if the bit is 1 then each party P_i places a weak encryption of its view, view_i, in the corresponding slot (represented by filled-in rectangles of various shades of gray in the figure). Otherwise, if the bit is 0 then each party places a weak encryption of all 0's in the corresponding slot, (represented by blank rectangles in the figure). During verification, the verifier checks in the first step that input $s = s_1^q \oplus s_2^q \oplus \ldots \oplus s_\ell^q$. To check the consistency of the views, the verifier selects 2 columns (P_2 and P_4) and 2 rows (s_i^q and s_ℓ^q) at random and does the following: (1) checks that s_i^q and s_ℓ^q are consistent with the values read in the first step (2) runs Rec on the weakly encrypted views and checks that the resulting views are consistent with each other and internally. 192

6.1 Illustration of a Sunflower 200
6.2 Illustration of 3-slice-Merkle Tree: The figure shows a 3-slice Merkle tree with $n = 81$ number of blocks, where each block has size $\mathcal{X}$. The hash function h maps inputs of size $\mathcal{X}$ to outputs of size $\mathcal{X}/3$. As a concrete example, starting from leaf block α_{45}^0, we apply the hash h to obtain the leftmost (0-th) slice of block α_{15}^1. Next, hashing α_{15}^1, we obtain the leftmost (0-th) slice of block α_5^2. Then, hashing α_5^2, we obtain the rightmost (2-nd) slice of block α_1^3. Finally, hashing α_1^3, we obtain the middle (1-st) slice of block α_0^4. Thus, in order to check consistency or update leaf block 45 at the bottom, we need to access only the path to block 45, consisting of $(\alpha_{45}^0, \alpha_{15}^1, \alpha_5^2, \alpha_1^3, \alpha_0^4)$, as described above. Note that no sibling nodes are accessed. 249

7.1 Construction of 2-State, Continuous, Non-Malleable Randomness Encoder. 287
7.2 Decode in Hybrid 1. 288
7.3 Gen and Challenge Codeword generation in Hybrid 2. 289
7.4 Decode in Hybrid 3. 290
7.5 Decode in Hybrid 4. 291
7.6 Construction of 4-state Continuous, Non-Malleable Code. 298
7.7 Algorithm $\mathsf{D}'^3_{\Sigma_{Eq}}$. 300
7.8 Algorithm $\mathsf{D}'^4_{\Sigma_{Eq}}$. 301

List of Abbreviations

NMC	Non-Malleable Codes
CNMC	Continuous Non-Malleable Codes
LDUNMC	Locally Decodable and Updatable Non-Malleable Codes
CRS	Common Reference String
PRG	Pseudo-Random Generator
ECC	Error Correcting Codes
PKE	Public Key Encryption
NMRE	Non-Malleable Randomness Encoder
CNMRE	Continuous Non-Malleable Randomness Encoder
RPE	Reconstructable Probabilistic Encoding
OWF	One Way Function
CRHF	Collision Resistant Hash Function
NIZK	Non Interactive Zero Knowledge

Chapter 1: Introduction

Cryptographic systems play an instrumental role in providing provable security guarantees in situations where sensitive data is handled or transferred. Traditionally, such systems assume that the adversarial entities interact with the system in a black-box manner. Specifically, it is crucial for the security proof that some *secret* internal state of the system cannot be manipulated (or even inferred) by the attacker. Unfortunately, adversarial capabilities in practice often allow much more access. Such attacks, often called *side channel attacks*, aim to recover the internal secret state by analyzing extra information learned via side-channels such as power consumption, timing analysis of certain computations etc. [28, 95]. An adversary can also tamper with secret state (say flip some bits) in order to learn about the secret by studying the modified input-output behavior; such attacks are known as tampering attacks [24, 27, 112, 115].

In their seminal work in 2010, Dziembowski et al. [66] defined the notion of non-malleable codes as an extension of error-correcting codes to achieve tamper resilience. Whereas error-correcting codes provide the guarantee that (if not too many errors occur) the receiver can recover the original message from a corrupted codeword, non-malleable codes are essentially concerned with security. In other words, correct decoding of corrupted codewords is not guaranteed (nor required), but it is instead guaranteed that if the decoding is incorrect then it is *independent* of the original message. Non-malleable codes can be used to encode the secret key in the memory of a device such that a tampering adversary interacting with the device does not learn anything more than the input-output behavior. Unfortunately, it is impossible to construct non-malleable codes secure against arbitrary tampering, since the adversary can always apply the tampering function that decodes the entire codeword to recover the message m and then re-encodes a related message m'. Thus, non-malleable codes are typically constructed against limited classes of tampering functions $\mathcal{F}$.

Somewhat formally, non-malleable codes are defined via a Ideal-Real paradigm definition. In the real world, a challenger creates a codeword c by encoding some underlying message m. The challenger then applies an adversarial tampering function f from a specific tampering class $\mathcal{F}$, on the codeword c, and then decodes the tampered codeword. The adversarial view in the real world comprises of the decoded message obtained after the decoding the tampered codeword. On the other hand, in an ideal world, there exists a simulator which is given the adversarial tampering function f. The simulator then samples the decoded message from a distribution which depends *only* on f. The simulator can also output a special symbol same^*, to indicate that decoding of the tampered codeword matches the original message m (note that, in the ideal world the simulator must indicate this without any knowledge of m). The view of the adversary in the ideal world consists of the simulator's output. The coding scheme is called non-malleable if the adversarial views in the two cases are indistinguishable from each other for all messages, and for all tampering functions in the tampering class $\mathcal{F}$.

Tampering Classes and Complexity Prior work, due to the impossibility result mentioned above, focused on providing resilience against *split-state* tampering functions, where an attacker is allowed to arbitrarily modify two (or more) parts of the secret key independently. In this dissertation we introduce a novel formalization of tampering functions based on well-studied complexity classes which are known to be strictly weaker from the class of all polynomial time algorithms (P). Studying the tampering classes through the lens of complexity theory, allows us to achieve security against attackers with limited computational resources, who can corrupt the codeword in a non split-state manner. The work presented in this dissertation is the first work to specifically investigate tampering functions that correspond to well-studies classes in complexity theory. This research direction has generated significant research interest and has been followed up by several important advances in the study of non-malleable codes [15–18, 36].

We begin, in chapter 4, by constructing explicit, efficient, and unconditionally secure non-malleable codes against a powerful tampering class which includes all bounded-depth circuits with bounded fan-in and unbounded fan-out; the well-studied complexity class NC^0 is a special case of this tampering class. The class of bounded depth circuits is natural both as a complexity class and as a stepping stone towards modeling practical tampering attacks, where an adversary has *limited time or other computational resources to tamper with memory* before the memory gets overwritten and/or refreshed.

The work presented in chapter 4, is based on a paper co-authored with Marshall Ball, Dana Dachman-Soled, and Tal Malkin [17]; which is published in Eurocrypt 2016.

In chapter 5, we extend the previous direction and present general frameworks for constructing non-malleable codes for encoding one and multiple bits against broad classes of tampering functions $\mathcal{F}$ for which average case hardness results are known. Note that one cannot simply use the NMC for encoding single bit, in order to encode multiple bits by encoding each bit individually. This is because, a simple tampering function which re-orders the encodings will reveal a related message after decoding. Our frameworks include both a generic construction, which requires that certain underlying primitives are instantiated in a suitable way, as well as a proof "template." We instantiate our framework for particular tampering classes $\mathcal{F}$ in both the computational and information theoretic setting such as: (1) tampering functions that are represented by polynomial size, constant-depth, *unbounded fan-in and unbounded fan-out* circuits, such tampering functions correspond to AC^0 circuits (2) tampering functions represented by bounded-depth decision trees (3) tampering functions that are represented by read-once, bounded-width branching programs, such tampering functions correspond to the space-bounded, streaming setting.

The work presented in chapter 5, is based on a paper co-authored with Marshall Ball, Dana Dachman-Soled, and Tal Malkin [18]; which is published in Eurocrypt 2018.

Improving Efficiency Standard non-malleable codes are not suitable in settings where, say, an entire database must be protected; because they do not allow for random access. For example, the entire database would have to be re-encoded even if only a single entry is read/written by the user. Dachman-Soled et al. [54] proposed a new notion called locally decodable and updatable non-malleable codes, which informally speaking, provides the security guarantees of a non-malleable code while also allowing for efficient random access. In chapter 6 we analyze the theoretical upper and lower bounds on such constructions. In particular we show that any locally decodable and updatable code must have super-constant locality in order to be secure, as we show an efficient, explicit attack on schemes with lower locality. We also improve the performance of the scheme presented by Dachman-Soled et al. [54] and match the lower bound of super-constant locality.

The work presented in chapter 6, is based on a paper co-authored with Dana Dachman-Soled, and Aria Shahverdi. A preliminary version of this work [53] is published in Public Key Cryptography 2017, and the full version has been accepted for publication in the Journal of Information & Computation.

Tamper Resilience Against Stronger Adversaries Recently, Faust et al. [71] introduced the notion of *continuous* non-malleable codes (CNMC), which provides stronger security guarantees than standard non-malleable codes, by allowing an adversary to tamper with the codeword in a continuous way instead of one-time tampering. They also showed that CNMC with information theoretic security cannot be constructed in 2-split-state tampering model, and presented a construction of the same in CRS (common reference string) model using collision-resistant hash functions (CRHF) and non-interactive zero-knowledge proofs (NIZK).

Since it is known that 2-split-state CNMC imply one-way functions (OWF), OWF is the minimal computational assumption necessary to construct CNMC. The computational assumptions used in prior constructions (NIZK and CRHF) are believed to be stronger than one-way functions. Therefore, an interesting research question is to reduce this gap by constructing 2-split-state CNMC from weaker assumptions.

In chapter 7, we ask if it is possible to construct CNMC from such weaker assumptions. We answer this question by presenting lower as well as upper bounds. Specifically, we show that it is impossible to construct 2-split-state CNMC, with no CRS, for one-bit messages from any falsifiable assumption with black-box reduction, thus establishing the lower bound. Black-box reduction here means, that the security reduction has only input/output (black-box) access to the adversary breaking the security of CNMC. We additionally provide an upper bound by constructing 2-split-state CNMC for one-bit messages, assuming only the existence of a family of injective one way functions, which is believed to be a significantly weaker computational assumption than NIZK. We also present a construction of 4-split-state CNMC for multi-bit messages in CRS model from the same assumptions. Additionally, we present definitions of the following new primitives: 1) *One-to-one commitments*, and 2) *Continuous Non-Malleable Randomness Encoders*, which may be of independent interest.

The work presented in chapter 7, is based on a paper co-authored with Dana

Dachman-Soled [52]; which is published in Public Key Cryptography 2019.

We next give an outline of the dissertation, followed by overview of the chapters presenting technical results.

1.1 Organization

The dissertation is structured as follows: Chapter 2 gives an overview of prior work. It is followed by some preliminaries and standard definitions in chapter 3. Chapter 4, presents the construction of non-malleable codes for bounded depth bounded fan-in circuits, which is followed by presentation of general framework for constructing non-malleable codes from average-case hardness in chapter 5. In chapters 6 and 7 we present the upper and lower bounds on variants of standard non-malleable codes (locally decodable and updatable NMC in chapter 6, and continuous non-malleable codes in chapter 7). We conclude by presenting conclusions in chapter 8

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

In chapter 4, we give a construction of non-malleable codes where the attacker is allowed to tamper with the key arbitrarily in a non split-state manner, but is constrained in the computational complexity of the tampering function. This model is inspired from the need to capture real world attacks where an attacker usually gets chance to tamper with the entire secret key but has limited time to carry out the attack. In this work, we devise explicit, efficient, and unconditionally secure non-malleable codes against a powerful tampering class which includes all bounded-depth circuits with bounded fan-in and unbounded fan-out. This tampering class includes NC^0 . Specifically, we consider the class Local^{d_o}, consisting of all functions $f : \{0,1\}^n \to \{0,1\}^n$ that can be computed with output locality $d_o(n)$, where each output bit depends on at most $d_o(n)$ input bits. Note that this class includes all fan-in-b circuits of depth at most $\log_b d_o$. 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 in chapter 4, our constructions actually hold for an even broader class, that also includes all split state functions, and beyond.

1.3 Non-Malleable Codes from Average-Case Hardness: AC0, Decision Trees, and Streaming Space-Bounded Tampering

In chapter 5,we continue this line of research and consider constructing non-malleable codes against various complexity classes, including: (1) AC^0 tampering, where the tampering function is represented by a polynomial size constant-depth, unbounded fan-in/fan-out circuit, (2) tampering with bounded-depth decision trees, where the tampering function is represented by a decision tree with n variables and depth n^{ε} for $\varepsilon < 1$, (3) streaming tampering with quadratic space, where the tampering function is represented by a read-once, bounded-width ($2^{o(n^2)}$) branching program, (4) small threshold circuits: depth d circuits of majority gates with a quasilinear number of wires, (5) fixed polynomial time tampering: randomized turing machines running in time $O(n^k)$ for any fixed k. Constructing non-malleable codes against a wide array of complexity classes is desirable since in practice, the capabilities of a tampering adversary are uniquely tied to the computational setting under consideration and/or the physical device being used. AC^0 circuits, which are constant-depth circuits, model attackers with limited time, since the propagation delay of a circuit is proportional to the length of the longest path from input to output.

We also present general frameworks for constructing non-malleable codes for encoding one and multi-bits against various tampering classes $\mathcal{F}$ for which average case hardness results are known. Our frameworks (one for single-bit and one for multi-bit) include a generic construction, which requires suitable instantiations of underlying primitives, along with a proof "template." Our frameworks are inspired by the well-known double-encryption paradigm for constructing CCA2-secure public key encryption schemes [99, 102, 111].

1.4 Tight Upper and Lower Bounds for Leakage-Resilient, Locally Decodable and Updatable Non-Malleable Codes

Another aspect of tamper and leakage resilient cryptography is the efficiency of these schemes. The earlier works suffered from the severe drawbacks on this front since those constructions did not support the random access of data., For example, the entire database was needed to be re-encoded even if only a single file was read/written by the user. This is not practical in setting such as cloud based data storage systems where updates to data are often quite small in size compared to overall data stored. The recent work of Dachman-Soled et al [54], addressed this by introducing the locally decodable and updatable non-malleable codes (LDUNMC). These codes allow the user to access only few blocks of data to be accessed (or re-encoded) when the data is read (or updated). These codes therefore are useful in large databases where random access to the data is desired, which is the case with all practical systems.

Locally decodable and updatable non-malleable codes, (LDUNMC) informally speaking, provides the security guarantees of a non-malleable code while also allowing for efficient random access. In more detail, we consider a message $m = m_1, \ldots, m_n$ consisting of n blocks, and an encoding algorithm $\mathsf{E}(m)$ that outputs a codeword $\hat{C} = \hat{c}_1, \ldots, \hat{c}_{\hat{n}}$ consisting of $\hat{n}$ blocks. As introduced by Katz and Trevisan [92], local decodability means that in order to retrieve a single block of the underlying message, one does not need to read through the whole codeword but rather, one can access just a few blocks of the codeword. Similarly, local updatability means that in order to update a single block of the underlying messages, one only needs to update a few blocks of the codeword. Dachman-Soled et al. in [54] considered LDUNMC that are also *leakage-resilient* (LR-LDUNMC), thus allowing for adversaries who continually leak information about D in addition to tampering. The locality achieved by the construction of [54] is $\Theta(\log(n))$, meaning that when encoding messages of length n number of blocks, the decode and update procedures each require access to $\Theta(\log(n))$ number of blocks of the encoding. Thus, when using the encoding scheme of [54] to compile a RAM program into its secure version, the overhead is at least $\Omega(\log(n))$ memory accesses for each read/write access in the underlying program. In practice, such an overhead is often prohibitive. [1]

In chapter 6, we ask whether it is possible to construct leakage-resilient, locally decodable and updatable non-malleable codes (LR-LDUNMC)that achieve significantly better locality. When considering both leakage and tampering attacks (even just a single leakage query followed in a later round by a single tampering query) so-called *rewind attacks* become possible. In a rewind attack, the attacker does the following (1) **leak** information on only a "few" blocks of memory in rounds $1, \ldots, i$; (2) **wait** during rounds $i + 1, \ldots, j$ until these memory locations are (with high probability) modified by the "updater" (the entity that models the honest computation on the data); (3) **re-write** the old information into these memory locations in round $j+1$, with the goal of causing the state of the computation to be *rewound*. Rewind attacks can be thwarted by ensuring that when the old information is written back, it becomes *inconsistent* with other positions of the codeword and an error is detected.

Our results show that any construction of LDUNMC in a threat model which allows for a rewind attack as above will require "high locality." Specifically, we

[1]Although the ORAM scheme used in the compiler also has $\omega(\log(n))$ overhead, in many applications of interest, properties of the specific RAM program can be leveraged so that the overhead of ORAM can be reduced such that it becomes practically feasible. On the other hand, the $\Theta(\log(n))$ overhead of the encoding scheme of [54] is entirely agnostic to the RAM program being run on top and thus, the high overhead would be incurred in all applications.

show tight upper and lower bounds: (1) Every such construction will require super-constant locality, moreover; (2) Super-constant locality is sufficient for achieving constructions in the same threat model as [54] (which, as discussed, allows for rewind attacks). In this work, we assume that the decode and update procedures are *non-adaptive* in the sense that the next block of the codeword accessed during decode/update does not depend on the contents of the previous blocks accessed. For non-adaptive decode and update we consider both deterministic as well as randomized access patterns, for more details refer chapter 6.

1.5 Upper and Lower Bounds for Continuous Non-Malleable Codes

In chapter 7, we study *continuous non-malleable codes* which is a stronger variant of non-malleable codes.

Standard non-malleable codes achieve security only against one-time tampering. This means that in applications, the non-malleable encoding of a secret key must be continually decoded and re-encoded, each time the device is run, incurring overhead in computation and in generation of randomness for re-encoding. This motivated a stronger notion of non-malleable codes, known as continuous non-malleable codes (CNMC), introduced by Faust et al. [71]. This definition allows many-time tampering, which means that the adversary can continuously tamper with the codeword and observe the effects of the tampering. Due to known impossibility results, there must also be a "self-destruct" mechanism. This means that if, upon decode, the device detects an error, then a "self-destruct" mechanism, which erases the secret key, is triggered, rendering the device useless.

The notion of CNMC with respect to a tampering class $\mathcal{F}$ is as follows: Given a coding scheme $\Pi = (\mathsf{E}, \mathsf{D})$, where E is the encoding function and D is the decoding function, the adversary gets to interact with an oracle $\mathcal{O}_{\Pi}(C)$, parameterized by Π and an encoding of a message m, $C \leftarrow \mathsf{E}(m)$. We refer to the encoding C as the "challenge" encoding. In each round, the adversary gets to submit a tampering function $f \in \mathcal{F}$. The oracle evaluates $C' = f(C)$. If $\mathsf{D}(C') = \perp$, the oracle outputs $\perp$ and a "self-destruct" occurs, aborting the experiment. Otherwise, if $C' = C$, the oracle outputs a special message "same." Otherwise, the oracle outputs C'. We emphasize that the entire tampered codeword is returned to the adversary in this case. A CNMC is secure if for every pair of messages m_0, m_1, the adversary's view in the above game is computationally indistinguishable when the message is m_0 or m_1.

The original CNMC paper of [71] showed an information-theoretic impossibility result for 2-split-state CNMC. This shows that the CNMC setting is distinguished from other NMC settings, since information-theoretic (unconditional) security is impossible. Prior work [71] constructed 2-split-state CNMC in the *CRS model* using collision-resistant hash functions and NIZK. On the other hand, CNMC's imply commitment schemes, which in turn imply OWF. It remains to determine where CNMC lies in terms of complexity assumptions and what are the minimal computational assumptions needed to achieve CNMC.

In chapter 7, we present upper and lower bounds for CNMC in the 2-split-state model. First, we show that with no CRS, single-bit CNMC in the 2-split-state model (with a black-box security proof) is impossible to construct from any falsifiable assumption. On the other hand, in the CRS model, we show how to achieve single-bit CNMC in the 2-split-state model from injective one-way functions.

Chapter 2: Related Work

In this chapter we give an overview of some related prior work.

2.1 Non-Malleable Codes

Study of non-malleable cryptography was initiated by Dolev, Dwork and Naor in [59]. It has been studied in both computational and information theoretic setting. Error-correcting codes along with early works on tamper resilience [77,87,88], motivated the study of non-malleable codes.

Dziembowski, Pietrzak and Wichs [66] introduced and formalized the notion of non-malleable codes. Non-malleable codes have also been studied in both the computational and information-theoretic setting. In their seminal work [66] also introduced the split-state tampering function class, which allows the adversarial tampering function to tamper with different parts of the codeword independently. They showed the existence of such codes for the class of all bit-wise independent tampering functions (which can be viewed as split state with n parts).

Liu and Lysyanskaya [100], presented an efficient construction of computationally secure non-malleable codes for split state functions, in the CRS model. Their construction also provided security against continual split-state leakage.

Further studies on non-malleable codes for split state classes focused on improvements in terms of achieving information-theoretic security, gain stronger tamper resilience by reducing the number of states, and by improving the efficiency by increasing the rate. Information theoretic non-malleable code for 1-bit messages against 2 state split state functions, were constructed in [64], which was followed by Aggarwal et.al [4], who used results from additive combinatorics to construct information-theoretic non-malleable code for k-bit messages. Aggarwal et.al [3] presented an elegant framework for proving results about non-malleable codes via composable non-malleable reductions.

Studies improving the efficiency of non-malleable codes include [4, 39], which presented constructions for a constant (> 2) number of states with constant rate. For computationally secure 2-split-state non-malleable codes [2] improved the rate to $1-o(1)$ (which is optimal). Since then a long line of research [2,10,38,90,97,98] has improved the state-of-the-art. Recently, Kanukurthi et.al [91], gave a construction for 3-split-state non-malleable codes with rate $\frac{1}{3}$.

Beyond Split-State Tampering While most of the research on non-malleable codes prior to this work focused on split-state tampering we note few exceptions here. Agrawal et.al. [8] gave a construction of non-malleable codes against tampering functions which permute individual bits of the codeword (in addition to bit-wise independent tampering). Chabanne et.al [29] considered subclass of linear tampering functions.

Prior to this work, other studies presented some existential [66], randomized constructions (or efficient constructions in CRS model) of non-malleable codes [42, 73] for more general tampering classes.

For example, non-malleable codes secure against any 'small-enough' tampering family ($< 2^{2^n}$) were proved to exist in [66].

Faust et.al [73] constructed information theoretically secure, efficient non-malleable codes in CRS model, for tampering function families $\mathcal{F}$ with size $|\mathcal{F}| \leq 2^{\mathsf{poly}(n)}$, where n is the length of codeword. Their construction was based on t-wise independent hashing for t proportional to $\log |\mathcal{F}|$. Note that the construction of [73] does not achieve security against the tampering functions $f \in \mathsf{AC}^0$ in general, rather provides tamper resilience against *specific* families in AC^0 (ACC^0, etc.) This is because, the bound on the size of F is required to be fixed *before* t-wise independent hash function h (CRS) is chosen. Whereas, AC^0 contains *all* poly-size and constant depth circuit families,

Cheraghchi and Guruswami [42] gave the first characterization of the rate of non-malleable codes. They showed that non-malleable codes with information theoretic security exist for tampering families $\mathcal{F}$ of size $|\mathcal{F}| \leq 2^{2^{\alpha n}}$. Moreover, they also showed that these existential NMC have optimal rate of $1 - \alpha$. The efficiency of the encoding and decoding algorithms in [42] construction, is proportional to $\mathsf{poly}(1/\varepsilon)$ where ε is the error probability, this leads to inefficient instantiations when error is negligible.

Following on the work presented in Chapter 4, Chattopadhyay and Li [36] constructed non-malleable codes using seedless non-malleable extractors. They also construct efficient NMC against t-local tampering functions, with deterministic decoding algorithm. (in contrast, the result in 4 has randomized decoding). We achieve better locality parameters than [36]. They [36], also constructed inefficient non-malleable codes for AC^0 tampering functions. The length of the codeword for [36] construction, is super-polynomial in the message length. Whereas, in chapter 5 we present a construction of computationally secure *efficient* non-malleable code for AC^0 tampering functions in CRS model.

Recently, Faust et.al [70] considered larger tampering classes by considering space bounded tampering adversaries in random oracle model. They defined a new notion of *leaky* continuous non-malleable codes, where the adversary can learn bounded information ($\log(|m|)$ bits) about the underlying message m, and gave construction in random oracle model. In chapter 5 we present information theoretically secure NMC (with standard non-malleability definition in plain model) for streaming space-bounded tampering functions.

Block-wise non-malleable codes (variant of standard NMC) was considered by Chandran et.al in [31]. In this model, the codeword consists of number of blocks and the adversary receives the codeword block-by-block. The tampering function also consists of various function f_is, where each f_i can depend on codeword blocks $c_1, \ldots, c_i$ and modifies c_i to c'_i. It can be observed that standard non-malleability cannot be achieved in this model since, the adversary can simply wait to receive all the blocks of the codeword and then decode the codeword as part of last tampering function. Therefore, [31] define a new notion called non-malleability with replacement which relaxes the non-malleability requirement and considers the attack to be successful only if the tampered codeword is *valid and related* to the original message.

This work is followed by [15, 16, 19], which improves on the results in chapters 4 and 5. In [15], Ball et.al construct efficient, information-theoretic NMC for AC^0 tampering functions, and further improvements are shown in [19] where they construct NMC for tampering functions which can be modeled as sub-exponential size AC^0 circuits. Note that both [15, 19], consider tampering functions with depth $O(\log n/\log\log n)$ which includes AC^0 . Ball et.al in [16], present computational NMC in plain (no-CRS) model, for tampering functions which run in any *fixed* bounded polynomial-time, based on de-randomization assumptions, with inverse polynomial adversarial advantage (instead of negligible). We view these results as complementary to the work presented in this dissertation, since together they highlight how different choices of parameters such as size of the tampering circuit, security model (CRS vs. plain vs. information-theoretic), and desired rate (codeword length), can result in achieving tamper-resilience against adversaries with limited computational resources.

Other works Several other variants and enhanced models were considered. For example, [46], in the context of designing UC secure protocols via tamperable hardware tokens, consider a variant of non-malleable codes which has *deterministic* encoding and decoding. Locally decodable and updatable non-malleable code were introduced by [54]. Faust et.al [71] considered stronger variant which allows continual tampering, and [6] allow for bounded leakage model.

Other works on non-malleable codes include [7, 34, 35, 37, 43, 72, 93, 94, 109].

2.2 Locally Decodable and Updatable Non-Malleable Codes

Locally Decodable Codes Katz and Trevisan [92] introduced *locally decodable* codes. Where,they considered the problem of recovering individual bits of a codeword from accessing a small number of bits from a (possibly) corrupted error correcting codeword. They [92] also showed the impossibility of achieving the above for schemes with linear codeword length. This work was followed by various works, including [40, 67, 119] who achieved constant locality with super-polynomial code length, while on the other hand locally decodable codes with constant rate and sublinear locality have been constructed by [81,84,96]. See [120] for a survey on locally decodable codes.

Locally Updatable and Locally Decodable Codes The notion of *locally updatable and locally decodable codes* was introduced by Chandran et al. in [32] where the constraint of *locality*, i.e. restricting the number of bits accessed, is further applied to updating any codeword obtained from encoding of another message. [32] gave information theoretic construction with amortized update locality of $\mathcal{O}(\log^2 k)$ and read locality of (super-linear) polynomial in k, where k is the length of input message. Another variant called *locally updatable and locally decodable-detectable codes* was also introduced in the same work which ensures that decoding never outputs an incorrect message. [32] gave the construction of such codes in the computational setting with poly-logarithmic locality.

Locally Decodable and Updatable Non-Malleable Codes Dachman-Soled et al. introduced the notion of *locally decodable and updatable non-malleable codes* in [54] and presented a construction in the computational setting. The construction of [54] also achieves leakage resilience in addition to the tamper resilience. [54] then used this notion to construct compilers that transform any RAM machine into a RAM machine secure against leakage and tampering. This application was also studied by Faust et al. [72], who presented a different approach which does not use locally decodable and updatable non-malleable codes. Recently, Chandran et al. [33] gave a construction of locally decodable and updatable non-malleable codes in the information-theoretic setting. However, they addressed only the one-time leakage and tampering case, and to achieve continual leakage and tampering, require a periodic refresh of the entire memory.

Bounds on Non-Malleable Codes. Surprisingly, understanding the limitations and bounds on NMC has received relatively less attention. While there have been a few previous works exploring the lower and upper bounds on NMC and its variants [31, 42, 66], most of the effort has been focused on understanding and/or improving the bounds on the rates of NMC [2, 8, 10, 50, 90, 98]

Perhaps the closest to this work are the results of [42]. Cheragachi and Guruswami [42] studied the "capacity" of non-malleable codes in order to understand the optimal bounds on the efficiency of non-malleable codes. They showed that information theoretically secure efficient NMC exist for tampering families $\mathcal{F}$ of size $|\mathcal{F}|$ if $\log\log|\mathcal{F}| \leq \alpha n$ for $0 \leq \alpha < 1$, moreover these NMC have optimal rate of $1-\alpha$

with error $\varepsilon \in O(1/\mathsf{poly}(n))$.

In chapter 6, we study the bounds on the locality of locally decodable and updatable NMC. We show that for any locally decodable and updatable NMC which allows rewind attacks, the locality parameter of the scheme must be $\omega(1)$, and give an improved version of [54] construction to match the lower bound in computational setting.

In chapter 7, we study the bounds on continuous non-malleable codes (CNMC), and show that 2-split-state CNMC cannot be constructed from any falsifiable assumption without CRS. We also give a construction of 2-split-state CNMC from injective one-way functions in CRS model. Faust et al. [71] showed the impossibility of constructing information-theoretically secure 2-split-state CNMC.

2.3 Continuous Non-Malleable Codes

Continuous Non-Malleable Codes *Continuous Non-Malleable codes (CNMC)* were introduced by Faust et.al. in [71]. They gave a construction of CNMC based on existence of collision resistant hash functions (CRHFs) and non-interactive zero knowledge proof systems (NIZKs) in common reference string (CRS) model. They also showed the impossibility of constructing 2-split state CNMC information theoretically. Subsequent to the original Faust et.al construction, Jafargholi and Wichs [89] presented a general study of CNMCs and its variants with some existential results. Recently, Aggarwal et. al. [5] gave first information theoretic construction in 8-split-state model.

Recently, Ostrovsky et. al. [105] introduced a relaxed notion of CNMC,[1] which is sufficient for many applications. In the work of Ostrovsky et. al. [105], they refer to the original notion as "continuous super-non-malleability" (since it is analogous to "super-non-malleability", a notion that was introduced in the non-continuous setting [73]). They then presented a construction achieving the relaxed definition (which they simply call "continuous non-malleability"), against 2-split-state tampering functions, assuming the existence of injective one-way functions in the plain model (without CRS).

The difference between the two CNMC notions is that in the original CNMC notion, the tampering oracle returns the entire modified codeword C' if $C' = f(C) \neq C$ and $\mathsf{D}(C) \neq \perp$, whereas the relaxation only requires the oracle to return $\mathsf{D}(C')$ but not C' itself. The original notion captures stronger types of tampering attacks; specifically, it provides security against an adversary who learns arbitrary additional information about the modified codeword C' through other side-channels.

Our result and the result of [105] are complementary and together give a full picture of the landscape of assumptions required for CNMC. Our work shows that it is *necessary* to rely on setup assumptions (CRS) in order to achieve the original, stronger security definition of CNMC. Moreover, if one is willing to assume the existence of a CRS, we show that this type of CNMC can be achieved from nearly minimal computational assumptions. In contrast, if one is not willing to assume the existence of a CRS, the work of [105] achieves weaker security guarantees in the

[1]A similar relaxed definition was previously given for a variant of CNMC, known as R-CNMC [69], but in this setting it was shown that it is actually impossible to achieve the stronger notion.

plain model (with no setup assumptions) from the same computational assumptions. We also note that the work of Ostrovsky et. al. [105] explicitly lists the question we address in this work as an interesting open problem. They state:

> "Interesting open questions related to our work are, for instance, whether continuous non-malleability can be achieved, under minimal assumptions, together with additional properties, such as strong non-malleability, **super-non-malleability**, augmented non-malleability, and locality ..."

Other works on non-malleable codes include [50, 69].

Non-Malleable Randomness Encoders (NMRE) NMRE were introduced recently by Kanukurthi et. al. [91] as a building block for constructing efficient (constant-rate) split-state NMC. In this work, we present the stronger variant *Continuous NMRE* which allows continual tampering in split-state model.

Black-Box Separations. Impagliazzo and Rudich ruled out black-box reductions from key agreement to one-way function in their seminal work [86]. Their oracle separation technique was subsequently used to rule out black-box reductions between various primitives such as collision resistant hash functions to one way functions [114], oblivious transfer to public key encryption [79] and many more. The meta-reduction technique (cf. [1, 22, 51, 74–76, 78, 106, 107, 113]) has been useful for ruling out larger classes of reductions—where the construction is arbitrary (non-black-box), but the reduction uses the adversary in a black-box manner. The meta-reduction technique is often used to provide evidence that construction of some cryptographic primitive is impossible under “standard assumptions” (e.g. falsifiable assumptions or non-interactive assumptions).

Chapter 3: Preliminaries and Definitions

In this chapter we will present background material and definitions related to non-malleable codes and its variants. We will also present definitions related to boolean circuits and other cryptographic primitives which will be used in constructions presented in chapters 4, 5 , 6, and 7

3.1 Notation

Firstly, we present some standard notations that will be used in what follows. Let $\mathbb{N}$ be the set of all natural numbers, i.e., $\mathbb{N} = \{1, 2, 3, \ldots\}$. For $n \in \mathbb{N}$, we write $[n] = \{1, \ldots, n\}$. If $x = (x_1, \ldots, x_n) \in \Sigma^n$ (for some set Σ), then $x_{i:j} := (x_i, x_{i+1}, \ldots, x_{j-1}, x_j)$ for $i \leq j$. If Σ is a set, then $\Sigma^\Sigma := \{f : \Sigma \to \Sigma\}$, the set of all functions from Σ to Σ. We say two vectors $x, y \in \Sigma^n$ are ε-*far* if they disagree on at least $\varepsilon \cdot n$ indices, $|\{i : x_i \neq y_i\}| \geq \varepsilon n$. Conversely, we say two vectors $x, y \in \Sigma^n$ are $(1-\varepsilon)$-*close* if they agree on at least $(1-\varepsilon) \cdot n$ indices, $|\{i : x_i = y_i\}| \geq (1-\varepsilon)n$. Alternatively, for $x, y \in \mathrm{GF}(2)^n$ define their distance to be $d(x, y) := \frac{\|x+y\|_0}{n}$. (I.e. x and y are ε-far if $d(x, y) \geq \varepsilon$.)

For a set S, $x \leftarrow S$ denotes, sampling an element x uniformly at random from the set S. For an algorithm A, $y \leftarrow A(x)$ is the output obtained on execution of A on input x. If $A(\cdot,\cdot)$ is a randomized algorithm, then $y \leftarrow A(x,r)$, is the output random variable for input x and randomness r. We also write, $A(x)$ instead of $A(x,r)$ if it is clear from the context for the brevity. A function $\delta(\cdot)$ is called *negligible* if for all sufficiently large n and for every polynomial $p(\cdot)$, it holds that $\delta(n) < 1/p(n)$. We will denote a negligible function by $\mathsf{negl}(\cdot)$.

For a random variable X, we sometimes also denote the corresponding probability distribution by X. An ensemble of probability distributions $\{X_\lambda\}_{\lambda\in\mathbb{N}}$ is a sequence of probability distributions. For two probability ensembles $\{X\}_\lambda$ and $\{Y\}_\lambda$, defined over a domain S with finite support we say that $\{X\}_\lambda$ and $\{Y\}_\lambda$ are *statistically indistinguishable* if there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\frac{1}{2}\sum_{s\in S} \left|\Pr\left[X_\lambda = s\right] - \Pr\left[Y_\lambda = s\right]\right| \leq \mathsf{negl}(\lambda)$$

We denote statistical indistinguishability by $X_\lambda \approx_s Y_\lambda$.

Similarly, we say that two probability ensembles $\{X\}_\lambda$ and $\{Y\}_\lambda$, defined over a domain S with finite support we say that $\{X\}_\lambda$ and $\{Y\}_\lambda$ are *computationally indistinguishable* if for all probabilistic polynomial time distinguishers $\mathcal{D}$, there exists a negligible function $\mathsf{negl}(\cdot)$ such that for all $\lambda \in \mathbb{N}$,

$$\left|\Pr_{x\leftarrow X_\lambda}\left[\mathcal{D}(x) = 1\right] - \Pr_{y\leftarrow Y_\lambda}\left[\mathcal{D}(y) = 1\right]\right| \leq \mathsf{negl}(\lambda)$$

We denote computational indistinguishability by $X_\lambda \approx_c Y_\lambda$.

If S is a set, we denote by U_S the uniform distribution over S. For $\lambda \in \mathbb{N}$, we denote by U_λ the uniform distribution over λ-bit strings.

Remark 3.1.1. *If a distribution $\mathcal{D}$ with support $\mathcal{S}$ of size 2^ℓ is statistically $2^{-\lambda}$-close to the uniform distribution over $\mathcal{S}$, denoted $U_\mathcal{S}$, then for every $x \in \mathcal{S}$, $\left|\Pr_{X\leftarrow\mathcal{D}}[X = x] - 1/2^\ell\right| \leq 1/2^\lambda$. This implies that $1/2^\ell - 1/2^\lambda \leq \Pr_{X\leftarrow\mathcal{D}}[X = x] \leq 1/2^\ell + 1/2^\lambda$.*

Where appropriate, we interpret functions $f : S \to \{\pm 1\}$ as boolean functions (and vice-versa) via the mapping: $0 \leftrightarrow 1$ and $1 \leftrightarrow -1$. The support of vector $\vec{x}$ is the set of indices i such that $x_i \neq 0$. A *bipartite graph* is an undirected graph $G = (V, E)$ in which V can be partitioned into two sets V_1 and V_2 such that $(u, v) \in E$ implies that either $u \in V_1$ and $v \in V_2$ or $v \in V_1$ and $u \in V_2$.

3.2 Non-Malleable Codes

In this section we define the notion of *non-malleable codes* and its variants. In this work, we assume that the decoding algorithm of the non-malleable code may be *randomized* and all of our generic theorems are stated for this case. Nevertheless, only our instantiations in Chapter 4, and Section 5.5 requires a randomized decoding algorithm, while our other instantiations enjoy deterministic decoding. We note that this definition differs from the original one given in [66], in that we allow the decoding to be randomized, while they required deterministic decoding. While this technically weakens our definition (and a code with deterministic decoding would be preferable), we feel that allowing randomized decoding fits the spirit and motivation of non-malleable codes.More importantly, it may allow for a wider classes of functions.

Definition 3.2.1 (Coding Scheme)**.** *Let $\Sigma, \widehat{\Sigma}$ be sets of strings, and $\kappa, \widehat{\kappa} \in \mathbb{N}$ be some parameters. A coding scheme consists of two algorithms* (E, D) *with the following syntax:*

- *The encoding algorithm* (perhaps randomized) *takes input a block of message in Σ and outputs a codeword in $\hat{\Sigma}$.*
- *The decoding algorithm* (perhaps randomized) *takes input a codeword in $\hat{\Sigma}$ and outputs a block of message in Σ.*

We require that for any message $m \in \Sigma$, $\Pr[\mathsf{D}(\mathsf{E}(m)) = m] = 1$, where the probability is taken over the choice of the encoding algorithm. In binary settings, we often set $\Sigma = \{0,1\}^{\kappa}$ and $\widehat{\Sigma} = \{0,1\}^{\widehat{\kappa}}$.

We next provide definitions of non-malleable codes of varying levels of security. The following Definitions 3.2.2, 3.2.3 are the standard definitions of non-malleability and strong non-mallebility, appropriate for the information theroetic setting (without CRS). These definitions are special cases of the corresponding Definitions 3.2.4, 3.2.5, presented in Section 3.2.1; when taking crs to be $\perp$ and $\mathcal{G}$ to be the set of all functions (namely the adversary is not restricted, and there's no CRS).

Definition 3.2.2 (Non-malleability [66])**.** *Let k be the security parameter, $\mathcal{F}$ be some family of functions. For each function $f \in \mathcal{F}$, and $m \in \Sigma$, define the tampering experiment:*

$$\mathbf{Tamper}_m^f \overset{\text{def}}{=} \left\{ \begin{array}{c} c \leftarrow \mathsf{E}(m), \tilde{c} := f(c), \tilde{m} := \mathsf{D}(\tilde{c}). \\ \textit{Output}: \ \tilde{m}. \end{array} \right\},$$

where the randomness of the experiment comes from the encoding algorithm. We say a coding scheme (E, D) *is non-malleable with respect to* $\mathcal{F}$ *if for each* $f \in \mathcal{F}$*, there exists a PPT simulator* Sim *such that for any message* $m \in \Sigma$*, we have*

$$\mathbf{Tamper}_m^f \approx \mathbf{Ideal}_{\mathsf{Sim},m} \overset{\text{def}}{=} \left\{ \begin{array}{c} \tilde{m} \cup \{\mathsf{same}^*\} \leftarrow \mathsf{Sim}^{f(\cdot)}. \\ \textit{Output}: m \textit{ if output of } \mathsf{Sim} \textit{ is } \mathsf{same}^*; \ \textit{otherwise } \tilde{m}. \end{array} \right\}$$

Here the indistinguishability can be either computational or statistical.

Definition 3.2.3 (Strong Non-malleability [66])**.** *Let* k *be the security parameter,* $\mathcal{F}$ *be some family of functions. For each function* $f \in \mathcal{F}$*, and* $m \in \Sigma$*, define the tampering experiment*

$$\mathbf{StrongNM}_m^f \overset{\text{def}}{=} \left\{ \begin{array}{c} c \leftarrow \mathsf{E}(m), \tilde{c} := f(c), \tilde{m} := \mathsf{D}(\tilde{c}) \\ \textit{Output}: \ \mathsf{same}^* \textit{ if } \tilde{c} = c, \ \textit{and } \tilde{m} \textit{ otherwise.} \end{array} \right\}$$

The randomness of this experiment comes from the randomness of the encoding algorithm. We say that a coding scheme (E, D) *is strong non-malleable with respect to the function family* $\mathcal{F}$ *if for any* $m, m' \in \Sigma$ *and for each* $f \in \mathcal{F}$*, we have:*

$$\{\mathbf{StrongNM}_m^f\}_{k \in \mathbb{N}} \approx \{\mathbf{StrongNM}_{m'}^f\}_{k \in \mathbb{N}}$$

where $\approx$ can refer to statistical or computational indistinguishability.

3.2.1 Non-Malleable Code in CRS Model

Here, we present general, game-based definitions that are applicable even for NMC that are in a model with a crs, or that require computational assumptions. The corresponding original definitions of non-malleability (presented in Section 3.2), appropriate for an unconditional setting without a CRS, can be obtained as a special case of the following definitions when setting $\mathsf{crs} = \perp$ and taking $\mathcal{G}$ to include all computable functions.

Definition 3.2.4 (Non-malleability). *Let $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ be a coding scheme. Let $\mathcal{F}$ be some family of functions. For each attacker A, $m \in \Sigma$, define the tampering experiment $\mathsf{Tamper}^{\Pi,\mathcal{F}}_{A,m}(n)$:*

1. Challenger samples $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^n)$ and sends crs to A.
2. Attacker A sends the tampering function $f \in \mathcal{F}$ to the challenger.
3. Challenger computes $c \leftarrow \mathsf{E}(\mathsf{crs}, m)$.
4. Challenger computes the tampered codeword $\tilde{c} = f(c)$ and computes $\tilde{m} = \mathsf{D}(\mathsf{crs}, \tilde{c})$.
5. Experiment outputs $\tilde{m}$.

Figure 3.1: Non-Malleability Experiment $\mathsf{Tamper}^{\Pi,\mathcal{F}}_{A,m}(n)$

We say the coding scheme $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ is non-malleable against tampering class $\mathcal{F}$ and attackers $A \in \mathcal{G}$, if for every $A \in \mathcal{G}$ there exists a PPT simulator

Sim *such that for any message* $m \in \Sigma$ *we have,*

$$\mathsf{Tamper}_{A,m}^{\Pi,\mathcal{F}}(n) \approx \mathbf{Ideal}_{\mathsf{Sim},m}(n)$$

where $\mathbf{Ideal}_{\mathsf{Sim},m}(n)$ *is an experiment defined as follows,*

1. Simulator Sim has oracle access to adversary A and outputs $\tilde{m} \cup \{\mathsf{same}^*\} \leftarrow \mathsf{Sim}^{A(\cdot)}(n)$.
2. Experiment outputs m if Sim outputs same^* and outputs $\tilde{m}$ otherwise.

Figure 3.2: Non-Malleability Experiment $\mathbf{Ideal}_{\mathsf{Sim},m}(n)$

Definition 3.2.5 (Strong Non-malleability)**.** *Let* $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *be a coding scheme. Let* $\mathcal{F}$ *be some family of functions. For each attacker* A*,* $m \in \Sigma$*, define the tampering experiment* $\mathsf{StrongTamper}_{A,m}^{\Pi,\mathcal{F}}(n)$*:*

1. Challenger samples $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^n)$ and sends crs to A.
2. Attacker A sends the tampering function $f \in \mathcal{F}$ to the challenger.
3. Challenger computes $c \leftarrow \mathsf{E}(\mathsf{crs}, m)$.
4. Challenger computes the tampered codeword $\tilde{c} = f(c)$.
5. Compute $\tilde{m} = \mathsf{D}(\mathsf{crs}, \tilde{c})$.
6. Experiment outputs same^* if $\tilde{c} = c$, and $\tilde{m}$ otherwise.

Figure 3.3: Strong Non-Malleability Experiment $\mathsf{StrongTamper}_{A,m}^{\Pi,\mathcal{F}}(n)$

We say the coding scheme $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *is strong non-malleable against* *tampering class $\mathcal{F}$ and attackers $A \in \mathcal{G}$ if we have*

$$\mathsf{StrongTamper}^{\Pi,\mathcal{F}}_{A,m_0}(n) \approx \mathsf{StrongTamper}^{\Pi,\mathcal{F}}_{A,m_1}(n)$$

for any $A \in \mathcal{G}$, $m_0, m_1 \in \Sigma$.

3.2.2 Medium Non-Malleability

We now introduce an intermediate variant of non-malleability, called *Medium Non-malleability*, which informally gives security guarantees "in-between" strong and regular non-malleability. Specifically, the difference is that the experiment is allowed to output same^* only when some predicate g evaluated on $(c, \tilde{c})$ is set to true. Thus, strong non-malleability can be viewed as a special case of medium non-malleability, by setting g to be the identity function. On the other hand, regular non-malleability does not impose restrictions on when the experiment is allowed to output same^*. Note that g cannot be just any predicate in order for the definition to make sense. g must be a predicate such that if g evaluated on $(c, \tilde{c})$ is set to true, then (with overwhelming probability over the random coins of D) $\mathsf{D}(\tilde{c}) = \mathsf{D}(c)$.

Definition 3.2.6 (Medium Non-malleability)**.** *Let $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ be a coding scheme. Let $\mathcal{F}$ be some family of functions.*

Let $g(\cdot,\cdot,\cdot,\cdot)$ be a predicate such that, for each attacker $A \in \mathcal{G}$, $m \in \Sigma$, the output of the following experiment, $\mathsf{Expt}^{\Pi,\mathcal{F}}_{A,m,g}(n)$ is 1 with at most negligible probability:

For g as above, each $m \in \Sigma$, and attacker $A \in \mathcal{G}$, define the tampering exper-

1. Challenger samples $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^n)$ and sends crs to A.
2. Attacker A sends the tampering function $f \in \mathcal{F}$ to the challenger.
3. Challenger computes $c \leftarrow \mathsf{E}(\mathsf{crs}, m)$.
4. Challenger computes the tampered codeword $\tilde{c} = f(c)$.
5. Challenger samples $r \leftarrow U_\ell$.
6. Experiment outputs 1 if $g(\mathsf{crs}, c, \tilde{c}, r) = 1] \wedge \mathsf{D}(\mathsf{crs}, \tilde{c}; r) \neq m)$.

Figure 3.4: The experiment corresponding to the special predicate g.

iment

$\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,m,g}(n)$ *as shown in figure 3.5:*

1. Challenger samples $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^n)$ and sends crs to A.
2. Attacker A sends the tampering function $f \in \mathcal{F}$ to the challenger.
3. Challenger computes $c \leftarrow \mathsf{E}(\mathsf{crs}, m)$.
4. Challenger computes the tampered codeword $\tilde{c} = f(c)$.
5. Challenger samples $r \leftarrow U_\ell$ and computes $\tilde{m} = \mathsf{D}(\mathsf{crs}, \tilde{c}, r)$.
6. Experiment outputs same^* if $g(\mathsf{crs}, c, \tilde{c}, r) = 1$, and $\tilde{m}$ otherwise.

Figure 3.5: Medium Non-Malleability Experiment $\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,m,g}(n)$

We say the coding scheme $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *is medium non-malleable against tampering class* $\mathcal{F}$ *and attackers* $A \in \mathcal{G}$ *if we have*

$$\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,m_0,g}(n) \approx \mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,m_1,g}(n)$$

for any $A \in \mathcal{G}$, $m_0, m_1 \in \Sigma$.

The following Definition 3.2.7 corresponds to a special case of Definition 3.2.6

of medium non-malleability that we introduced, appropriate for the information theroetic setting (without CRS). Again,when taking crs to be $\perp$ and $\mathcal{G}$ to be the set of all functions (namely the adversary is not restricted, and there's no CRS).

Definition 3.2.7 (Medium Non-malleability)**.** *Let k be the security parameter, $\mathcal{F}$ be some family of functions. Let $c \leftarrow \mathsf{E}(m)$ and let $g(c, \tilde{c}, r)$ be a predicate such that, for every c in the support of $\mathsf{E}(m)$ and every $\tilde{c}$,*

$$\Pr[g(c, \tilde{c}, r) = 1] \wedge \mathsf{D}(\tilde{c}; r) \neq m] \leq \mathsf{negl}(n).$$

For g as above, each function $f \in \mathcal{F}$, and $m \in \Sigma$, define the tampering experiment

$$\mathbf{MediumNM}^f_{m,g} \stackrel{\text{def}}{=} \left\{ \begin{array}{c} c \leftarrow \mathsf{E}(m), \tilde{c} := f(c), r \leftarrow U_\ell, \tilde{m} := \mathsf{D}(\tilde{c}; r) \\ \textit{Output}: \ \mathsf{same}^* \ \textit{if}\ g(c, \tilde{c}, r) = 1, \ \textit{and}\ \tilde{m} \ \textit{otherwise.} \end{array} \right\}$$

The randomness of this experiment comes from the randomness of the encoding algorithm and r (the random coins od decoding). We say that a coding scheme (E, D) is medium non-malleable with respect to the function family $\mathcal{F}$ if there exists a g as above and for any $m, m' \in \Sigma$ and for each $f \in \mathcal{F}$, we have:

$$\{\mathbf{MediumNM}^f_{m,g}\}_{k \in \mathbb{N}} \approx \{\mathbf{MediumNM}^f_{m',g}\}_{k \in \mathbb{N}}$$

where $\approx$ can refer to statistical or computational indistinguishability.

It is straightforward to check that Medium Non-Malleability implies non-malleability.

3.2.3 Continuous Non-Malleable Codes (CNMC)

Here, we present the definition of coding scheme, and continuous non-malleable codes.

Definition 3.2.8 (Coding Scheme [66]). *A coding scheme,* $\mathsf{Code} = (\mathsf{E}, \mathsf{D})$*, consists of two functions: a randomized encoding function* $\mathsf{E} : \{0,1\}^\lambda \to \{0,1\}^n$*, and a deterministic decoding function* $\mathsf{D} : \{0,1\}^n \to \{0,1\}^\lambda \cup [\perp]$ *such that, for each* $m \in \{0,1\}^\lambda$*,* $\Pr[\mathsf{D}(\mathsf{E}(m)) = m] = 1$ *(over the randomness of encoding function).*

Next, we present the definition of continuous non malleable codes in CRS model for codes with split-state encoding schemes.

Definition 3.2.9 (Split-State Encoding Scheme in the CRS model [71]). *A split-state encoding scheme in* common reference string *(CRS) model is a tuple of algorithms,* $\mathsf{Code} = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *specified as follows:*

- CRSGen *takes the security parameter as input and outputs the CRS,* $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$*.*
- E *takes a message* $x \in \{0,1\}^\lambda$ *as input along with the CRS* crs*, and outputs a codeword consisting of two parts* (X_0, X_1) *such that* $X_0, X_1 \in \{0,1\}^n$*.*
- D *takes a codeword* $(X_0, X_1) \in \{0,1\}^{2n}$ *as input along with the CRS* crs *and outputs either a message* $x' \in \{0,1\}^\lambda$ *or a special symbol* $\perp$*.*

Before defining the continuous non malleable codes consider the following oracle, $\mathcal{O}_{\mathsf{CNM}}((X_0, X_1), (\mathsf{T}_0, \mathsf{T}_1))$ which is parametrized by the CRS crs and "challenge" codeword (X_0, X_1) and takes functions $\mathsf{T}_0, \mathsf{T}_1 : \{0,1\}^n \to \{0,1\}^n$ as inputs.

$\underline{\mathcal{O}_{\mathsf{CNM}}(\mathsf{crs}, (X_0, X_1), (\mathsf{T}_0, \mathsf{T}_1))\text{:}}$

$(X'_0, X'_1) = (\mathsf{T}_0(X_0), \mathsf{T}_1(X_1))$

If $(X'_0, X'_1) = (X_0, X_1)$ return same

If $\mathsf{D}_{\mathsf{crs}}(X'_0, X'_1) = \perp$, return $\perp$ and "self destruct"

Else return (X'_0, X'_1).

"Self destruct" here means that once $\mathsf{D}_{\mathsf{crs}}(X'_0, X'_1)$ outputs $\perp$, the oracle answers all the future queries with $\perp$.

Definition 3.2.10 (Continuous Non Malleability [71])**.** *Let* $\mathsf{Code} = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *be a split-state encoding scheme in the CRS model. We say that* Code *is* q*-continuously non-malleable code, if for all messages* x, $y \in \{0,1\}^\lambda$ *and all PPT adversary* $\mathcal{A}$ *it holds that*

$$\left\{\mathsf{CTamper}_{\mathcal{A},x}(\lambda)\right\}_{\lambda\in\mathbb{N}} \approx_c \left\{\mathsf{CTamper}_{\mathcal{A},y}(\lambda)\right\}_{\lambda\in\mathbb{N}}$$

where

$$\mathsf{CTamper}_{\mathcal{A},x}(\lambda) \overset{def}{=} \left\{\begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda);\ (X_0, X_1) \leftarrow \mathsf{E}_{\mathsf{crs}}(x); \\ \mathsf{out}_{\mathcal{A}} \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{CNM}}(\mathsf{crs},(X_0,X_1),(\cdot,\cdot))};\ \text{OUTPUT} : \mathsf{out}_{\mathcal{A}} \end{array}\right\}$$

and $\mathcal{A}$ *asks total of* q *queries to* $\mathcal{O}_{\mathsf{CNM}}$*.*

The following is an equivalent formulation

Definition 3.2.11 (Continuous Non Malleability [71], equivalent formulation)**.** *Let* $\mathsf{Code} = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *be a split-state encoding scheme in the CRS model. We say that* Code *is* q*-continuously non-malleable code, if for all messages* m_0, $m_1 \in \{0,1\}^\lambda$*, all PPT adversary* $\mathcal{A}$ *and all PPT distinguishers* D *it holds that*

$$\Pr[D(\mathsf{out}_{\mathcal{A}}^b) = b] \leq 1/2 + \mathsf{negl}(\lambda)$$

where $b \leftarrow \{0,1\}$ *and*

$$\mathsf{out}_{\mathcal{A}}^b \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{CNM}}(\mathsf{crs},(X_0^b,X_1^b),(\cdot,\cdot))} : \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda);\ (X_0^b, X_1^b) \leftarrow \mathsf{E}_{\mathsf{crs}}(m_b)$$

and $\mathcal{A}$ *asks total of* q *queries to* $\mathcal{O}_{\mathsf{CNM}}$*.*

3.2.4 Locally Decodable and Updatable Non-Malleable Codes (LDUNMC)

Definition 3.2.1 (Locally Decodable and Updatable Code)**.** *Let* $\Sigma, \hat{\Sigma}$ *be sets of strings, and* $n, \hat{n}, p, q$ *be some parameters. An* $(n, \hat{n}, p, q)$ *locally decodable and updatable coding scheme consists of three algorithms* $(\mathsf{E}, \mathsf{D}, \mathsf{UP})$ *with the following syntax:*

- *The encoding algorithm* E *(*perhaps randomized*) takes input an* n*-block (in* Σ*) message and outputs an* $\hat{n}$*-block (in* $\hat{\Sigma}$*) codeword.*
- *The (local) decoding algorithm* D *takes input an index in* $[n]$*, reads at most* p *blocks of the codeword, and outputs a block of message in* Σ*. The overall decoding*

algorithm simply outputs $(\mathsf{D}(1), \mathsf{D}(2), \ldots, \mathsf{D}(n))$.

- *The (local) updating algorithm* UP *(*perhaps randomized*) takes inputs an index in* $[n]$ *and a string in* $\Sigma \cup \{\epsilon\}$, *and reads/writes at most* q *blocks of the codeword. Here the string* ϵ *denotes the procedure of refreshing without changing anything.*

Let $\hat{C} \in \hat{\Sigma}^{\hat{n}}$ *be a codeword. For convenience, we denote* $\mathsf{D}^{\hat{C}}, \mathsf{UP}^{\hat{C}}$ *as the processes of reading/writing individual block of the codeword, i.e. the codeword oracle returns or modifies individual block upon a query. Here we view* $\hat{C}$ *as a random access memory where the algorithms can read/write to the memory* $\hat{C}$ *at individual different locations. In binary settings, we often set* $\Sigma = \{0,1\}^{\kappa}$ *and* $\hat{\Sigma} = \{0,1\}^{\hat{\kappa}}$.

Definition 3.2.2 (Correctness)**.** *An* $(n, \hat{n}, p, q)$ *locally decodable and updatable coding scheme (with respect to* $\Sigma, \hat{\Sigma}$*) satisfies the following properties. Let* $\hat{C} = (\hat{c}_1, \hat{c}_2, \ldots, \hat{c}_{\hat{n}}) \leftarrow \mathsf{E}(M)$ *be a codeword output by the encoding algorithm for any message* $M = (m_1, m_2, \ldots, m_n) \in \Sigma^n$. *Then we have:*

- *for any index* $i \in [n]$, $\Pr[\mathsf{D}^{\hat{C}}(i) = m_i] = 1$, *where the probability is over the randomness of the encoding algorithm.*

- *for any update procedure with input* $(j, m') \in [n] \times \Sigma \cup \{\epsilon\}$, *let* $\hat{C}'$ *be the resulting codeword by running* $\mathsf{UP}^{\hat{C}}(j, m')$. *Then we have* $\Pr[\mathsf{D}^{\hat{C}'}(j) = m'] = 1$, *where the probability is over the encoding and update procedures. Moreover, the decodings of the other positions remain unchanged.*[1]

Remark 3.2.1. *The correctness definition can be directly extended to handle any*

[1]Our lower bound rules out even constructions with $1 - \mathsf{negl}(n)$ correctness. We present the correctness definition in terms of *perfect* correctness, as this is the standard notion in the coding literature, including the non-malleable codes literature, see for example [30, 33, 43, 54, 66, 71].

sequence of updates.

Definition 3.2.3 (Continual Tampering and Leakage Experiment)**.** *Let k be the security parameter, $\mathcal{F}, \mathcal{G}$ be some families of functions. Let $(\mathsf{E}, \mathsf{D}, \mathsf{UP})$ be an $(n, \hat{n}, p, q)$-locally decodable and updatable coding scheme with respect to $\Sigma, \hat{\Sigma}$. Let $\mathcal{U}$ be an updater that takes input a message $M \in \Sigma^n$ and outputs an index $i \in [n]$ and $m \in \Sigma$. Then for any blocks of messages $M = (m_1, m_2, \ldots, m_n) \in \Sigma^n$, and any (non-uniform) adversary $\mathcal{A}$, any updater $\mathcal{U}$, define the following continual experiment* $\mathbf{CTamperLeak}_{\mathcal{A},\mathcal{U},M}$*:*

- *The challenger first computes an initial encoding $\hat{C}^{(1)} \leftarrow \mathsf{E}(M)$.*
- *Then the following procedure repeats, at each round j, let $\hat{C}^{(j)}$ be the current codeword and $M^{(j)}$ be the underlying message:*
  - *$\mathcal{A}$ sends either a tampering function $f \in \mathcal{F}$ and/or a leakage function $g \in \mathcal{G}$ to the challenger.*
  - *The challenger replaces the codeword with $f(\hat{C}^{(j)})$, or sends back a leakage $\ell^{(j)} = g(\hat{C}^{(j)})$.*
  - *We define $\vec{m}^{(j)} \stackrel{\text{def}}{=} \Big(\mathsf{D}^{f(\hat{C}^{(j)})}(1), \ldots, \mathsf{D}^{f(\hat{C}^{(j)})}(n)\Big)$.*
  - *Then the updater computes $(i^{(j)}, m) \leftarrow \mathcal{U}(\vec{m}^{(j)})$ for the challenger.*
  - *Then the challenger runs $\mathsf{UP}^{f(\hat{C}^{(j)})}(i^{(j)}, m)$ and sends the index $i^{(j)}$ to $\mathcal{A}$.*
  - *$\mathcal{A}$ may terminate the procedure at any point.*
- *Let t be the total number of rounds above. At the end, the experiment outputs*

$$\left(\ell^{(1)}, \ell^{(2)}, \ldots, \ell^{(t)}, \vec{m}^{(1)}, \ldots, \vec{m}^{(t)}, i^{(1)}, \ldots, i^{(t)}\right).$$

Definition 3.2.4 (Non-malleability and Leakage Resilience against Continual Attacks)**.** *An $(n, \hat{n}, p, q)$-locally decodable and updatable coding scheme with respect to $\Sigma, \hat{\Sigma}$ is continual non-malleable against $\mathcal{F}$ and leakage resilient against $\mathcal{G}$ if for all* PPT*(non-uniform) adversaries $\mathcal{A}$, and* PPT*updaters $\mathcal{U}$, there exists some* PPT*(non-uniform) simulator Sim such that for any $M = (m_1, \ldots, m_n) \in \Sigma^n$,*
$\mathbf{CTamperLeak}_{\mathcal{A},\mathcal{U},M}$ *is (computationally) indistinguishable to the following ideal experiment $\mathbf{Ideal}_{\mathsf{Sim},\mathcal{U},M}$:*

- *The experiment proceeds in rounds. Let $M^{(1)} = M$ be the initial message.*
- *At each round j, the experiment runs the following procedure:*
    - *At the beginning of each round, Sim outputs $(\ell^{(j)}, \mathcal{I}^{(j)}, \vec{w}^{(j)})$, where $\mathcal{I}^{(j)} \subseteq [n]$.*
    - *Define*

$$\vec{m}^{(j)} = \begin{cases} \vec{w}^{(j)} & \text{if } \mathcal{I}^{(j)} = [n] \\ \vec{m}^{(j)}|_{\mathcal{I}^{(j)}} := \perp, \vec{m}^{(j)}|_{\bar{\mathcal{I}}^{(j)}} := M^{(j)}|_{\bar{\mathcal{I}}^{(j)}} & \text{otherwise,} \end{cases}$$

    where $\vec{x}|_{\mathcal{I}}$ denotes the coordinates $\vec{x}[v]$ where $v \in \mathcal{I}$, and the bar denotes the complement of a set.

    - *The updater runs $(i^{(j)}, m) \leftarrow \mathcal{U}(\vec{m}^{(j)})$ and sends the index $i^{(j)}$ to the simulator. Then the experiment updates $M^{(j+1)}$ as follows: set $M^{(j+1)} := M^{(j)}$ for all coordinates except $i^{(j)}$, and set $M^{(j+1)}[i^{(j)}] := m$.*

– *Let t be the total number of rounds above. At the end, the experiment outputs*

$$\left(\ell^{(1)}, \ell^{(2)}, \ldots, \ell^{(t)}, \vec{m}^{(1)}, \ldots, \vec{m}^{(t)}, i^{(1)}, \ldots, i^{(t)}\right).$$

3.3 Boolean Circuits and Related Definitions

We now present the definitions of boolean circuits, circuit complexity classses NC^0 and AC^0 . We also define input/output local functions in this section.

Definition 3.3.1 (Boolean Circuit). *[14] For every $n_1, n_2 \in \mathbb{N}$, a* Boolean Circuit *C with n_1 inputs and n_2 outputs is a directed acyclic graph. It contains n_1 nodes with 0 incoming edges; called* input nodes *and n_2 nodes with 0 outgoing edges, called* output nodes. *All other nodes are called* gates *correspond to one of the logical operations (AND, OR, and NOT).*
The AND and OR gates have fan-in 2, whereas NOT gate has fan-in 1.
The size *of boolean circuit C, is the total number of nodes in the DAG. The* depth *of the circuit is the length of the longest directed path from an input node to the output node.*

Definition 3.3.2 (Circuit Family). *[14] Let $p : \mathbb{N} \to \mathbb{N}$ be a function. A* circuit family *of size $p(n)$ is a sequence of boolean circuits $\{C_n\}_{n\in\mathbb{N}}$, where C_n has n inputs and a single output, such that size of each circuit $C_n \leq p(n)$ for all n.*

Let $f : \{0,1\}^* \to \{0,1\}^*$ be a polynomially bounded function, i.e. $\exists$ some constant $c > 0$, such that for all $x \in \{0,1\}^*$, $f(x) < |x|^c$. We call f *implicitly logspace computable* if the mapping $x, i \to f(x)_i$ can be computed in logarithmic space, where $f(x)_i$ denotes the i-th bit of the output $f(x)$.

Definition 3.3.3 (Logspace Uniform Circuit Families). *[14]*

A circuit family $\{C_n\}$ *is* logspace uniform *if there is an implicitly logspace computable function mapping* 1^n *to the description of the circuit* C_n*.*

We now define the complexity classes NC^0 and AC^0 .

Definition 3.3.4 (NC). *[14]*

A language is in NC^i *if there are constants* $c, i > 0$ *such that it can be decided by logspace uniform circuit family* $\{C_n\}$ *where circuits* C_n *are of size* $O(n^c)$ *and depth* $O(\log^i n)$*.*

The class NC *is* $\cup_{i\geq 0}\mathsf{NC}^i$*.*

NC^0 *corresponds to constant-depth bounded fan-in circuit family.*

Definition 3.3.5 (AC). *[14]*

The class AC^i *is similar to* NC^i *except that the gates in circuit* C_n *can have unbounded fan-in.*

The class AC *is* $\cup_{i\geq 0}\mathsf{AC}^i$*.*

AC^0 *corresponds to constant-depth unbounded fan-in circuit family.*

Next, we present definitions related to local functions.

3.3.1 Local Functions

We next define a class of local functions, where the number of input bits that can affect any output bit (input locality) and the number of output bits that depend on an input bit (output locality) are restricted. Loosely speaking, an input bit x_i affects the output bit y_j if for any boolean circuit computing f, there is a path in the underlying DAG from x_i to y_j. The formal definitions are below, and our notation follows that of [12]

Definition 3.3.6. *We say that a bit x_i* affects *the boolean function f,*

if $\exists \{x_1, x_2, \cdots x_{i-1}, x_{i+1}, \cdots x_n\} \in \{0,1\}^{n-1}$ such that,

$f(x_1, x_2, \cdots x_{i-1}, 0, x_{i+1}, \cdots x_n) \neq f(x_1, x_2, \cdots x_{i-1}, 1, x_{i+1}, \cdots x_n)$.

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

Definition 3.3.7 (Output Locality)**.** *A function $f : \{0,1\}^n \to \{0,1\}^n$ is said to have* output locality *m if every output bit f_i is dependent on at most m input bits.*

Definition 3.3.8 (Input Locality)**.** *A function $f : \{0,1\}^n \to \{0,1\}^n$ is said to have* input locality *ℓ if every input bit f_i is affects at most ℓ output bits.*

Definition 3.3.9 (Local Functions)**.** *[12] A function $f : \{0,1\}^n \to \{0,1\}^n$ is said to be* (m,ℓ)-local, *$f \in \mathrm{Local}_\ell^m$, if it has input locality ℓ and output locality m. We denote the class* Local_n^m *(namely no restriction on the input locality) by* Local^m.

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

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

We also recall some definitions and results related to boolean analysis and present them next.

3.3.2 Background on Boolean Analysis

Definition 3.3.10. *A function* $f : \{0,1\}^n \to \{0,1\}$ *has* correlation c *with a function* $g : \{0,1\}^n \to \{0,1\}$ *if*

$$|\Pr_{x \leftarrow U_n}[f(x) = 1 | g(x) = 1|] - \Pr_{x \leftarrow U_n}[f(x) = 1 | g(x) = 0|]| \leq c.$$

Where, U_n *is the uniform random distribution over* $\{0,1\}^n$*.*

Note that this is equivalent up to absolute value for a more common definition of correlation in the literature when g is taken to be balanced ($\Pr[g(x) = 1] = 1/2$)

Definition 3.3.11. *A function* $f : \{0,1\}^n \to \{0,1\}$ *has correlation* c *with a function* $g : \{0,1\}^n \to \{0,1\}$ *if*

$$\Pr_{x \leftarrow U_n}[f(x) = g(x)] = \frac{1+c}{2}.$$

Where, U_n *is the uniform random distribution over* $\{0,1\}^n$*.*

The correlation c can also be expressed as follows: Let $\Pr[f(x) = g(x)] = \frac{1+c}{2}$.

Then,

$$\begin{aligned}
c &= 2\Pr[f(x) = g(x)]] - 1 \\
&= 2\Pr[f(x) = 1|g(x) = 1]\Pr[g(x) = 1] + 2\Pr[f(x) = 0|g(x) = 0]\Pr[g(x) = 0] - 1 \\
&= \Pr[f(x) = 1|g(x) = 1] + (1 - \Pr[f(x) = 1|g(x) = 0] - 1 \\
&= \Pr[f(x) = 1|g(x) = 1] - \Pr[f(x) = 1|g(x) = 0]
\end{aligned}$$

Theorem 3.3.1 ([83,85]). *Let $f : \{0,1\}^n \to \{0,1\}$ be computed by a depth-d circuit of size S. Then the correlation of f with parity is bounded by*

$$2^{-c_d n/\log^{d-1}(S)},$$

where c_d is a positive constant dependent only on d.

Definition 3.3.12 (Random Restriction [82]). *A* random restriction *ρ parameterized by a small positive real number p is mapping which sets the elements x_i of a vector $\vec{x}$ independently as follows:* $\Pr[x_i = 0] = \frac{1-p}{2}$*,* $\Pr[x_i = 1] = \frac{1-p}{2}$*, and* $\Pr[x_i = \star] = p$*.*

Definition 3.3.13 (Boolean Function Restriction). *For a vector $v \in \{0,1,*\}^n$ and a Boolean function $f : \{0,1\}^n \to \{0,1\}^n$ the restriction of f to v, $\tilde{f}|_v$ is defined as $\tilde{f}|_v(x) = f(z)$ where,*

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

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

Lemma 3.3.1 ([82, 118]). *Let $f : \{0,1\}^\ell \to \{0,1\}$ be a function computable by a depth-d AC^0 circuit of size s. Let ρ be a random restriction with $\Pr[\star] = q < 1/9^d$. The probability over ρ that f_ρ cannot be written as a decision tree of depth t is $\leq s(9q^{1/d}t)^t$.*

We next recall some standard definitions of public-key encryption (PKE), pseudorandom generator (PRG), and non-interactive zero knowledge proof systems with simulation soundness.

3.4 Public Key Encryption Scheme and PRG

In this section, we present the definitions of well known cryptographic primitives such as public key encryption scheme and pseudorandom generator which are used as building blocks for particular instantiations. A public key encryption scheme $\mathcal{E}$ consists of three algorithms: $(\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$.

- $\mathsf{Gen}(1^n) \to (\textsc{pk}, \textsc{sk})$. The key generation algorithm takes in the security parameter and outputs a public key PK and a secret key SK.
- $\mathsf{Encrypt}(\textsc{pk}, m) \to c$. The encryption algorithm takes in a public key PK and a message m. It outputs a ciphertext c.

- $\mathsf{Decrypt}(\textsc{sk}, c) \to m$. The decryption algorithm takes in a ciphertext c and a secret key $\textsc{sk}$. It outputs a message m.

Correctness. The PKE scheme satisfies correctness if $\mathsf{Decrypt}(\textsc{sk}, c) = m$ with all but negligible probability whenever $\textsc{pk}, \textsc{sk}$ is produced by Gen and c is produced by $\mathsf{Encrypt}(\textsc{pk}, m)$.

Security. We define IND-CPA security for PKE schemes in terms of the following game between a challenger and an attacker. We let n denote the security parameter.

Setup Phase. The game begins with a setup phase. The challenger calls $\mathsf{Gen}(1^n)$ to create the initial secret key $\textsc{sk}$ and public key $\textsc{pk}$.

Challenge Phase. The attacker receives $\textsc{pk}$ from the challenger. The attacker chooses two messages m_0, m_1 which it gives to the challenger. The challenger chooses a random bit $b \in \{0, 1\}$, encrypts m_b, and gives the resulting ciphertext to the attacker. The attacker then outputs a guess b' for b. The attacker wins the game if $b = b'$. We define the advantage of the attacker in this game as $\left|\frac{1}{2} - \Pr[b' = b]\right|$.

Definition 3.4.1 (IND-CPA security)**.** *We say a Public Key Encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is IND-CPA secure if any probabilistic polynomial time attacker only has a negligible advantage (negligible in n) in the above game.*

Definition 3.4.2 (α-correctness [63])**.** *For any function $\alpha : \mathbb{N} \to [0, 1]$, a public-key encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ is* α-correct *if*

$\Pr[\mathsf{Decrypt}(\textsc{sk}, (\mathsf{Encrypt}(\textsc{pk}, m)) \neq m] \leq 1 - \alpha(n)$, *where the probability is taken over the random coins of* Gen *used to generate* $(\textsc{pk}, \textsc{sk}) \leftarrow \mathsf{Gen}(1^n)$, *for uniform random message* $m \in \{0,1\}^n$, *and for all possible random coins of* $\mathsf{Encrypt}$.

Definition 3.4.3 (Almost-all-keys Perfect Decryption [63]). *A public-key encryption scheme* $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is* almost-all-keys perfectly correct *if with all but negligible probability over the random coins of* Gen *used to generate* $(\textsc{pk}, \textsc{sk}) \leftarrow \mathsf{Gen}(1^n)$, *for uniform random message* $m \in \{0,1\}^n$, *and for all possible random coins of* $\mathsf{Encrypt}$, *it holds that* $\Pr[\mathsf{Decrypt}(\textsc{sk}, (\mathsf{Encrypt}(\textsc{pk}, m)) \neq m] = 0$.

3.4.1 Pseudorandom Generators of Space-Bounded Computation

Definition 3.4.4 (Pseudorandom Generator). *A* pseudorandom generator *is an efficient, deterministic map* $\mathsf{prg} : \{0,1\}^n \rightarrow \{0,1\}^{\ell(n)}$, *where* $\ell(n) > n$ *such that for all PPT distinguishers* D; $|\Pr[D(G(x)) = 1] - \Pr[D(y) = 1]| \leq \mathsf{negl}(n)$, *when* $x \in \{0,1\}^n$ *and* $y \in \{0,1\}^{\ell(n)}$ *are chosen uniform randomly.*

Definition 3.4.5. *[103] A generator* $\mathsf{prg} : \{0,1\}^m \rightarrow (\{0,1\}^n)^k$ *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*

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

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

Theorem 3.4.1. *[103] 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* $n, 2^k$*).*

3.5 Non-Interactive Zero Knowledge

Another important cryptographic primitive used in the constructions of non-malleable codes in Chapter 5 is non-interactive zero knowledge proof systems.

Definition 3.5.1 (Non-Interactive Zero Knowledge [111])**.** $\Pi = (\ell, \mathsf{P}, \mathsf{V}, \mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2))$ *is an* efficient adaptive single-theorem non-interactive zero knowledge proof system *for language* $L \in \mathsf{NP}$ *with witness relation* W*, if* ℓ *is a polynomial and the following are true:*

- *Completeness: For all* $x \in L$*, and all* w *such that* $W(x, w) = 1$*, for all strings* crs *of length* $\ell(|x|)$*, we have* $\mathsf{V}(x, \mathsf{P}(x, w, \mathsf{crs}), \mathsf{crs})) = 1$

- *Soundness: For all adversaries* A*, if* $\mathsf{crs} \in \{0,1\}^{\ell(k)}$ *is chosen randomly, then* $\Pr[\mathsf{V}(x, \pi, \mathsf{crs}) = 1] \leq \mathsf{negl}(k)$*. Where,* $(x, \pi) \leftarrow A(\mathsf{crs})$ *and* $x \notin L$*.*

- *Single-Theorem Zero Knowledge: For all non-uniform polynomial-time adversaries* $A = (A_1, A_2)$ *we have that* $|\Pr[\mathsf{Expt}_A(k) = 1] - \Pr[\mathsf{Expt}_A^{\mathsf{Sim}}(k) = 1]| \leq \mathsf{negl}(k)$ *for following experiments* $\mathsf{Expt}_A(k)$ *and* $\mathsf{Expt}_A^{\mathsf{Sim}}(k)$

```
$\mathsf{Expt}_A(k)$:                          $\mathsf{Expt}_A^{\mathsf{Sim}}(k)$:

$\mathsf{crs} \leftarrow \{0,1\}^{\ell(k)}$               $(\mathsf{crs}, \kappa) \leftarrow \mathsf{Sim}_1(1^k)$
$(x, w, \tau) \leftarrow A_1(\mathsf{crs})$               $(x, w, \tau) \leftarrow A_1(\mathsf{crs})$
$\pi \leftarrow \mathsf{P}(x, w, \mathsf{crs})$               $\pi \leftarrow \mathsf{Sim}_2(x, \kappa)$
$\mathsf{return} A_2(\pi, \tau)$                  $\mathsf{return} A_2(\pi, \tau)$
```

Definition 3.5.2 (Weak Simulation Soundness [111]). *Let* $\Pi = (\ell, \mathsf{P}, \mathsf{V}, \mathsf{Sim} = (\mathsf{Sim}_1, \mathsf{Sim}_2))$ *be an efficient adaptive single-theorem non-interactive zero knowledge proof system for language L. We say that Π is* simulation-sound *if for all non-uniform probabilistic polynomial-time adversaries $A = (A_1, A_2)$,* $\Pr[\mathsf{Expt}_{A,\Pi}^{\mathsf{Sim}}(k) = 1] \leq \mathsf{negl}(k)$*, where* $\mathsf{Expt}_{A,\Pi}^{\mathsf{Sim}}(k)$ *is the following experiment:*

```
$\mathsf{Expt}_{A,\Pi}^{\mathsf{Sim}}(k)$:

$(\mathsf{crs}, \kappa) \leftarrow \mathsf{Sim}_1(1^k)$
$(x, \tau) \leftarrow A_1(\mathsf{crs})$
$\pi \leftarrow \mathsf{Sim}_2(x, \mathsf{crs}, \kappa)$
$(x^*, \pi^*) \leftarrow A_2(x, \pi, \mathsf{crs}, \tau)$
Output 1 iff $(\pi^* \neq \pi)$ and $(x^* \notin L)$ and $(\mathsf{V}(x^*, \pi^*, \mathsf{crs}) = 1)$
```

We say Π is one-time weak simulation sound *if the above holds for any probabilistic polynomial time A only allowed a single query to* Sim.

Sahai [111] constructed one-time simulation sound NIZK proof system from any given efficient non-interactive single-theorem adaptive zero knowledge proof system, and strong one-time signature schemes (which was built from one-way functions in the same work).

Definition 3.5.3 (Same-String NIZK [55])**.** *A NIZK argument system is called same-string NIZK if it satisfies the following property for all k:*

- ***(Same-String Zero Knowledge):*** *For all non-uniform probabilistic polynomial-time adversaries A, we have that*

$$|\Pr[X = 1] - \Pr[Y = 1]| \leq \mathsf{negl}(k)$$

where X and Y are as defined in (and all probabilities are taken over) the experiment $\mathsf{Expt}(k)$ *below:*

```
Expt(k):
    1. $(\mathsf{crs}, \tau) \leftarrow \mathsf{Sim}_1(1^k)$
    2. $X \leftarrow A^{\mathsf{P}(\cdot,\cdot,\mathsf{crs})}(\mathsf{crs})$
    3. $Y \leftarrow A^{\mathsf{Sim}'(\cdot,\cdot,\mathsf{crs},\tau)}(\mathsf{crs})$
```

where $\mathsf{Sim}'(x, w, \mathsf{crs}, \tau) \overset{\text{def}}{=} \mathsf{Sim}_2(x, \mathsf{crs}, \tau)$

- ***(Same-String Zero Knowledge,cont.):*** *The distribution on* crs *produced by* $\mathsf{Sim}_1(1^k)$ *is the uniform distribution over* $\{0, 1\}^{\ell(k)}$.

Definition 3.5.4 (Non-Interactive Simulatable Proof System)**.** *A tuple of probabilistic polynomial time algorithms* $\Pi^{\mathsf{NI}} = (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ *is a* non-interactive simulatable proof system *for language* $L \in NP$ *with witness relation* W *if* $(\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ *have the following syntax:*

- $\mathsf{CRSGen}^{\mathsf{NI}}$ *is a randomized algorithm that outputs* $(\mathsf{crs}^{\mathsf{NI}}, \tau_{\mathsf{sim}})$.
- *On input* crs, $x \in L$ *and witness* w *such that* $W(x,w) = 1$, $\mathsf{P}^{\mathsf{NI}}(\mathsf{crs}, x, w)$ *outputs proof* π.
- *On input* crs, x, π, $\mathsf{V}^{\mathsf{NI}}(\mathsf{crs}, x, \pi)$ *outputs either* 0 *or* 1.
- *On input* crs, τ_{sim} *and* $x \in L$, $\mathsf{Sim}^{\mathsf{NI}}(\mathsf{crs}, \tau_{\mathsf{sim}}, x)$ *outputs simulated proof* π'.

Completeness: *We require the following completeness property: For all* $x \in L$, *and all* w *such that* $W(x,w) = 1$, *for all strings* $\mathsf{crs}^{\mathsf{NI}}$ *of length* $\mathrm{poly}(|x|)$, *and for all adversaries* $\mathcal{A}$ *we have*

$$\Pr\left[\begin{array}{c}(\mathsf{crs}^{\mathsf{NI}}, \tau_{\mathsf{Sim}}) \leftarrow \mathsf{CRSGen}^{\mathsf{NI}}(1^n); (x,w) \leftarrow \mathcal{A}(\mathsf{crs}^{\mathsf{NI}});\\ \\ \pi \leftarrow \mathsf{P}^{\mathsf{NI}}(\mathsf{crs}^{\mathsf{NI}}, x, w) : \mathsf{V}^{\mathsf{NI}}(\mathsf{crs}^{\mathsf{NI}}, x, \pi) = 1\end{array}\right] \geq 1 - \mathsf{negl}(n)$$

Soundness: *We say that* Π^{NI} *enjoys soundness against adversaries* $\mathcal{A} \in \mathcal{G}$ *if: For all* $x \notin L$, *and all adversaries* $\mathcal{A} \in \mathcal{G}$:

$$\Pr\left[\begin{array}{c}(\mathsf{crs}^{\mathsf{NI}}, \tau_{\mathsf{Sim}}) \leftarrow \mathsf{CRSGen}^{\mathsf{NI}}(1^n);\\ \\ (x,\pi) \leftarrow \mathcal{A}(\mathsf{crs}^{\mathsf{NI}}) : \mathsf{V}^{\mathsf{NI}}(\mathsf{crs}^{\mathsf{NI}}, x, \pi) = 0\end{array}\right] \geq 1 - \mathsf{negl}(n)$$

The security properties that we require of Π^{NI} *will depend on our particular non-malleable code construction as well as the particular class,* $\mathcal{F}$, *of tampering* *functions that we consider. The exact properties needed are those that will arise from Theorems 5.3.1 and 5.4.1. In subsequent sections, we will show how to construct non-interactive simulatable proof systems satisfying these properties.*

Chapter 4: Non-Malleable Codes for Bounded Depth, Bounded Fan-in Circuits

4.1 Introduction

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

We prove the following.

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

This construction can be instantiated for any $d_o = o(n/\log n)$, and the resulting code has rate $\Theta(1/d_o)$. In general, since the the output length is $n = \Theta(kd_o)$ 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* $\Theta(1/d_o)$ *for any* $d_o = \omega(1)$*.*
- NC^0_c *for any constant* c*, with constant rate.*

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

While our result for NC^0 correspond to constant depth circuits, the first corollary above implies as a special case that the code is also non-malleable against any $\delta \log n$ depth NC circuit, for any constant $0 \leq \delta < 1$. 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.

Intermediate Results for (Input-and-Output) Local Functions. To prove our results, we use the concept of *non-malleable reduction*, introduced by Aggarwal et al. [3]. Informally, a class of functions $\mathcal{F}$ reduces to the class $\mathcal{G}$, if there is an encoding and decoding algorithms satisfying the following: applying the encoding to the input, then applying any $f \in \mathcal{F}$, and then applying the decoding, can be simulated by directly applying some function $g \in \mathcal{G}$ to the input. [3] prove that in this case a non-malleable code for $\mathcal{G}$ can be used to construct one for $\mathcal{F}$, and further prove a composition theorem, providing an elegant and powerful way to construct non-malleable codes.

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

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

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

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

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

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

Relation of Our Class to Previous Work. As mentioned above, almost all previous results presenting explicit and efficient non-malleable codes, do so for a split state tampering class (with two or more states). These classes are a special case of ours, as we explained, which is not surprising given that we use results for split state functions as a starting point to prove our result. As for the exceptions that go beyond split state, we note that the class of functions that permute the bits or apply bitwise manipulations, introduced by [9], is also a special case of our class, as it is a subset of Local^1 (in fact, even a subset of Local_1^1). The restricted linear tampering class considered by [29], on the other hand, seems incomparable to our class of output-local functions.

Thus, in terms of the tampering class captured, our results simultaneously encompass (and significantly extend) almost all previously known explicit, efficient constructions of non-malleable codes (we are aware of only one exception). This is not the case in terms of the *rate*, where several previous works focus on optimizing the rate for smaller classes of functions (e.g., [43] achieve rate $1 - o(1)$ non-malleable codes for bit-wise tampering functions, and [2] do so for split-state functions under computational assumptions), while we only achieve a constant rate for these classes.

We also mention that the original work of Dziembowski et al. [65] already considered the question of constructing non-malleable codes against the class $\mathrm{Local}^{\delta \cdot n}$, where n is the length of the codeword and $\delta < 1$ is a constant. We emphasize, however, that in [65] (and an improvement in [42]), they showed a construction of non-malleable codes against $\mathsf{Local}^{\delta \cdot n}$ in a *non-standard, random oracle model* where the encoding and decoding functions make queries to a random oracle, but the adversarial tampering function *does not* query the random oracle. Our work shows that it is possible to construct non-malleable codes for $\mathsf{Local}^{\delta \cdot n}$ for $\delta = o(1/\log n)$ in the standard model, with *no* random oracle.

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

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

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

4.1.1 Technical Overview

We give a high level technical overview of our constructions. We use as an underlying tool a so called "reconstructable probabilistic encoding scheme", a code that can correct a constant fraction of errors (denoted c^{err}), and enjoys some additional secret-sharing like properties: given a (smaller) constant fraction c^{sec} of the codeword gives no information on the encoded message, and can in fact be completed to a (correctly distributed) encoding of any message. This or similar tools were used in previous works either implicitly or explicitly, e.g., the construction based on Reed Solomon codes and Shamir secret sharing with Berlekamp-Welch correction, as used already in [23] 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., [44, 45, 49, 57].

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 d_o, then at most $n_L d_o$ bits from s^{R} could possibly influence the output of f^L. Thus, we will use E^R that is an RPE with $n_L d_o \leq c^{\mathsf{sec}} n_R$. This means that we can just fix the relevant $n_L d_o$ 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 ($n_R \geq \frac{n_L d_o}{c^{\mathsf{sec}}}$).

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 d_i, then at most $n_L d_i$ 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 $n_L d_i$ 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 [44], and a forthcoming version by the same authors [45], 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 [44, 45], while the real decoding algorithm uses the *first* subset large enough, it can be simulated by using *any* other large enough subset.

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

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

Removing Input Locality.

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

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

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

The solution is to use a PRG with a short seed. The seed of the PRG is now the additional string that is appended to the left side and the output of the PRG yields an indicator string which specifies the "relevant" locations for decoding. Note that now since the PRG seed of length r is short, we can, using the leakage resilient properties of the underlying local code, leak *all* $r \cdot d_o \leq c^{\mathsf{sec}} \cdot n_L \leq c^{\mathsf{sec}} \cdot n_R$ 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 3.4.5), an *unconditional* PRG is sufficient for our construction. Thus, our construction does not rely on any computational assumption.

Improving the parameters.

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

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

4.2 Preliminaries

In this section we will provide some more definitions and background required for our constructions and results presented in chapter 4.

4.2.1 Non-Malleable Codes: Alternate Definitions

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

Definition 4.2.1 (Non-Malleable Code). *[3] Let $\mathcal{F}$ denote a family of tampering functions. Let $\mathsf{E} : B \to A$, $\mathsf{D} : A \to B$ be a coding scheme. For all $f \in \mathcal{F}$ and all $x \in B$ define:*

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

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

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

and Tamper_x^f is at most ε.

The above of definition has its origins in [66]. Dziembowski, Pietrzak, and Wichs required the simulator to be efficient. Aggarwal et al. demonstrated that the above relaxation is, in fact, equivalent for deterministic decoding. Allowing decoding to be randomized does not affect their proof. For this reason, we will not concern ourselves with the efficiency of a simulator (or, equivalently, sampling relevant distributions) for the remainder of this chapter.

Aggarwal et al. provide a simpler alternative to the above simulation-based definition, which they prove equivalent. [3] Their definition is based on the notion of non-malleable reduction, which we will use in this chapter.

Definition 4.2.2 (Non-Malleable Reduction). *[3] Let $\mathcal{F} \subset A^A$ and $\mathcal{G} \subset B^B$ be some classes of functions. We say $\mathcal{F}$* reduces to *$\mathcal{G}$, $(\mathcal{F} \Rightarrow \mathcal{G}, \varepsilon)$, if there exists an efficient (randomized) encoding function $\mathsf{E} : B \to A$, and an efficient (randomized) decoding function $\mathsf{D} : A \to B$, such that*

(a) $\forall x \in B, \Pr[\mathsf{D}(\mathsf{E}(x)) = x] = 1$ *(over the randomness of* E, D*).*

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

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

Definition 4.2.3 (Non-Malleable Code). *[3] Let NM_k denote the set of* trivial manipulation functions *on k-bit strings, consisting of the identity function $id(x) = x$ and all constant functions $f_c(x) = c$, where $c \in \{0, 1\}^k$.*

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

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

Theorem 4.2.1 (Composition). *[3] If $(\mathcal{F} \Rightarrow \mathcal{G}, \varepsilon_1)$ and $(\mathcal{G} \Rightarrow \mathcal{H}, \varepsilon_2)$, then $(\mathcal{F} \Rightarrow \mathcal{H}, \varepsilon_1 + \varepsilon_2)$.*

We note that the proof given in [3] goes through unchanged with randomized decoding.

4.2.2 Tampering Families

Definition 4.2.4 (Split-State Model). *[66] The* split-state model, SS_k, *denotes the set of all functions:*

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

Theorem 4.2.2 (Efficient Split-State Non-malleable Codes). *[97] For any $k \in \mathbb{N}$, there exists an efficient, explicit 2-state non-malleable code with block length $2k$ and rate $\Omega(1/\log k)$ with negligible error.*

4.2.3 Reconstructable Probabilistic Encoding Scheme

Reconstructable Probabilistic Encoding (RPE) schemes were defined by Choi et al. (in an in-submission journal version of [44], as well as in [45]), extending a definition given by Decatur, Goldreich and Ron [57]. Informally, this is an error correcting code, which has an additional secrecy property and reconstruction property. The secrecy property allows a portion of the output to be revealed without leaking any information about the encoded message. The reconstruction property allows, given a message and a partial codeword for it, to reconstruct a complete consistent codeword. Thus, this is a combination of error correcting code and secret sharing, similar to what has been used in the literature already starting with Ben-Or, Goldwasser, and Wigderson [23].

Definition 4.2.5 (Binary Reconstructable Probabilistic Encoding)**.** *[44, 45] We say a triple* $(\mathsf{E}, \mathsf{D}, \mathsf{Rec})$ *is a* binary reconstructable probabilistic encoding scheme *with parameters* $(k, n, c^{\mathsf{err}}, c^{\mathsf{sec}})$*, where* $k, n \in \mathbb{N}$*,* $0 < c^{\mathsf{err}}, c^{\mathsf{sec}} < 1$*, if it satisfies the following properties:*

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

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

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

Choi et al. show that a construction of Decatur, Goldreich, and Ron [57] meets the above requirements.

Lemma 4.2.1. *[44, 45] For any $k \in \mathbb{N}$, there exists constants $0 < c^{\mathsf{rate}}, c^{\mathsf{err}}, c^{\mathsf{sec}} < 1$ such that there is a binary RPE scheme with parameters $(k, c^{\mathsf{rate}}k, c^{\mathsf{err}}, c^{\mathsf{sec}})$.*

Remark 4.2.1. *To achieve longer encoding lengths ck, with the same c^{err} and c^{sec} parameters, one can simply pad the message to an appropriate length.*

Specifically, Decatur, Goldreich and Ron [57] construct a probabilistic encoding scheme that possesses the first two properties listed above. Moreover, since the construction they present, instantiates E with a linear error-correcting code, we have that property (3) holds. (Any linear error-correcting code has efficient reconstruction.)

These are the parameters we use here, but we believe it may be possible to achieve a better rate if we use parameters based on the recent result of Coretti et al. [48] (see also [43]).

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

Theorem 4.3.1. (E, D) *is a* $(\mathrm{Local}^{d_o(k)}_{d_i(k)} \Rightarrow \mathrm{SS}_k, \mathsf{negl}(k))$*-non-malleable reduction given the following parameters for* $\mathrm{Local}^{d_o(k)}_{d_i(k)}$ *(where $c^{\mathsf{rate}}, c^{\mathsf{err}}, c^{\mathsf{sec}}$ are taken from lemma 4.2.1):*

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

Putting together Theorem 4.3.1 with Theorems 4.2.1 and 4.2.2, we obtain the following.

Corollary 4.3.1. $(\mathsf{E} \circ \mathsf{E}_{\mathrm{SS}}, \mathsf{D}_{\mathrm{SS}} \circ \mathsf{D})$ *is a* $(\mathrm{Local}_\ell^\ell, k, \mathsf{negl}(k))$*-non-malleable code with rate* $\Theta(1/\ell)$*, where* $\ell = \tilde{O}(\sqrt{n})$*.*

Remark 4.3.1. *The reduction presented below is, in fact, a* $(\mathrm{XLocal}_\ell^\ell \Rightarrow \mathrm{SS}_k, \mathsf{negl}(k))$*-non-malleable reduction, where* $\ell = \tilde{O}(\sqrt{n})$ *and* $\mathrm{XLocal}_\ell^\ell$ *is the following class of functions* $f : \{0,1\}^{n_L+n_R} \to \{0,1\}^{n_L+n_R}$*:*

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

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

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

We construct an encoding scheme (E, D) summarized in Figure 4.1 and parametrized below. We then show that the pair (E, D) is an $(\mathrm{Local}_{d_i(k)}^{d_o(k)}, \mathrm{SS}_k, \mathsf{negl}(k))$-non-malleable reduction. This immediately implies that given a non-malleable encoding scheme $(\mathsf{E}^{\mathsf{ss}}, \mathsf{D}^{\mathsf{ss}})$ for class SS_k (where SS is the class of split-state functions), the encoding scheme scheme $\Pi = (\mathsf{E}^{\mathsf{bd}}, \mathsf{D}^{\mathsf{bd}})$, where $\mathsf{E}^{\mathsf{bd}}(m) := \mathsf{E}(\mathsf{E}^{\mathsf{ss}}(m))$ and $\mathsf{D}^{\mathsf{bd}}(\vec{s}) := \mathsf{D}^{\mathsf{ss}}(\mathsf{D}(\vec{s}))$ yields a non-malleable code against $\mathrm{Local}_{d_i(k)}^{d_o(k)}$.

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

- $(\mathsf{E}_L, \mathsf{D}_L)$ parametrized by $(k, n_L, c_L^{\mathsf{err}}, c_L^{\mathsf{sec}}) := (k, c^{\mathsf{rate}}k, c^{\mathsf{err}}, c^{\mathsf{sec}})$ where $c^{\mathsf{err}}, c^{\mathsf{sec}}$, c^{rate} are taken from lemma 4.2.1.

- $n_{\mathsf{check}} := \log^2(k)$.

- $d_{\mathsf{sec}} := \sqrt{\frac{cn_L}{n_{\mathsf{check}}}} = \Theta(\frac{\sqrt{k}}{\log(k)})$.

- $(\mathsf{E}_R, \mathsf{D}_R)$ parametrized by $(k, n_R, c_R^{\mathsf{err}}, c_R^{\mathsf{sec}}) := (k, \frac{d_o c^{\mathsf{rate}} k}{c^{\mathsf{sec}}}, c^{\mathsf{err}}, c^{\mathsf{sec}})$.

- $n := d_o c^{\mathsf{rate}} k + c^{\mathsf{rate}} k = O(\frac{k^2}{\log^2(k))}$.

Note that this setting of parameters is taken with our forthcoming reduction in mind. (See Corollary 4.3.2 and Theorem 4.4.1.) 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}.

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

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

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

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

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

Figure 4.1: The $(\mathrm{Local}_{d_i(k)}^{d_o(k)}, \mathrm{SS}, \mathsf{negl}(k))$-non-malleable reduction (E, D)

- 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 $\ell \in R_{\mathsf{check}}$.

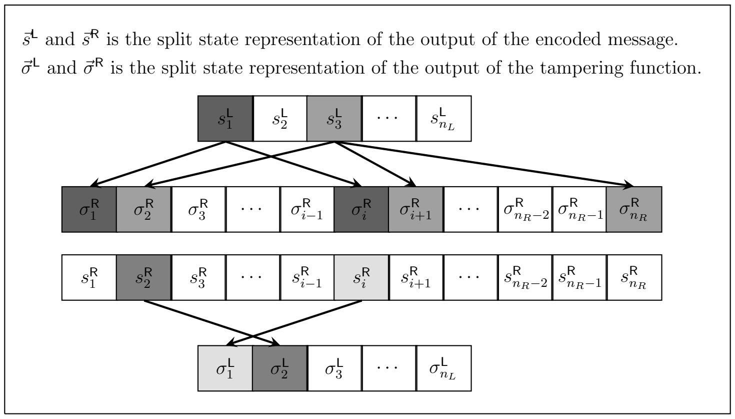
$\vec{s}^{\mathsf{L}}$ and $\vec{s}^{\mathsf{R}}$ is the split state representation of the output of the encoded message.
$\vec{\sigma}^{\mathsf{L}}$ and $\vec{\sigma}^{\mathsf{R}}$ is the split state representation of the output of the tampering function.
s^{L}_1 s^{L}_2 s^{L}_3 $\cdots$ $s^{\mathsf{L}}_{n_L}$
σ^{R}_1 σ^{R}_2 σ^{R}_3 $\cdots$ $\sigma^{\mathsf{R}}_{i-1}$ σ^{R}_i $\sigma^{\mathsf{R}}_{i+1}$ $\cdots$ $\sigma^{\mathsf{R}}_{n_R-2}$ $\sigma^{\mathsf{R}}_{n_R-1}$ $\sigma^{\mathsf{R}}_{n_R}$
s^{R}_1 s^{R}_2 s^{R}_3 $\cdots$ s^{R}_{i-1} s^{R}_i s^{R}_{i+1} $\cdots$ $s^{\mathsf{R}}_{n_R-2}$ $s^{\mathsf{R}}_{n_R-1}$ $s^{\mathsf{R}}_{n_R}$
σ^{L}_1 σ^{L}_2 σ^{L}_3 $\cdots$ $\sigma^{\mathsf{L}}_{n_L}$

Figure 4.2: The adversary chooses tampering function $f = (f^{\mathsf{L}}, f^{\mathsf{R}}) \in \mathrm{Local}^{d_o(k)}_{d_i(k)}$ 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 4.2. We observe the following immediate facts about their sizes:

Observation 1. *For $f \in \mathrm{Local}^{d_o}_{d_i}$, 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 $|\mathcal{S}_{\mathsf{L}\to\mathsf{R}}| \le d_i \cdot n_L \le n_R - t$.)

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 $\mathcal{S}_{\mathsf{check}} \leq d_o \cdot n_{\mathsf{check}}$.)

3. *By choice of parameters $n_L, n_R, c_R^{\mathsf{sec}}$, we have that $|\mathcal{S}_{\mathsf{R}\to\mathsf{L}}| \leq d_o \cdot n_L \leq n_R \cdot c_R^{\mathsf{sec}}$.*

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

- Choose a random subset $R_{\mathsf{check}} \subseteq [n_R]$ of size n_{check}.
- Choose vectors $I^{\mathsf{L}} \in \{0,1\}^{n_L} \times \{*\}^{n_L}$, $I^{\mathsf{R}} \in \{*\}^{n_L} \times \{0,1\}^{n_R}$ uniformly at random.
- Let J^* be the subset of $[n_R]$ as described in Observation 1.
- The split-state tampering function $g := (g_L, g_R) \in \text{SS}_k$ 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 $\vec{s}^{\mathsf{L}} := \mathsf{Rec}(\mathcal{S}_{\mathsf{check}}, I^{\mathsf{L}}, \mathsf{L})$. Let $(\sigma_1^{\mathsf{L}}, \ldots, \sigma_{n_L}^{\mathsf{L}}) := f^{\mathsf{L}}|_{I^{\mathsf{R}}}(\vec{s}^{\mathsf{L}})$.
     Compute $((w_1^{\mathsf{L}}, \ldots, w_{n_L}^{\mathsf{L}}), \widetilde{\mathsf{L}}) \leftarrow \mathsf{D}_L(\sigma_1^{\mathsf{L}}, \ldots, \sigma_{n_L}^{\mathsf{L}})$. If the decoding fails, set $\widetilde{\mathsf{L}} := \perp$.
  2. (`output`) Output $\widetilde{\mathsf{L}}$.

  $g_R(\mathsf{R})$:

  1. (`apply tampering and decoding-check on right`) Let $\vec{s}^{\mathsf{R}} = (s_1^{\mathsf{R}}, \ldots, s_{n_R}^{\mathsf{R}}) := \mathsf{Rec}(\mathcal{S}_{\mathsf{R}\to\mathsf{L}}, I^{\mathsf{R}}, \mathsf{R})$. Let $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{n_R}^{\mathsf{R}}) := f^{\mathsf{R}}|_{I^{\mathsf{L}}}(\vec{s}^{\mathsf{R}})$.

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 $((w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}}), \widetilde{\mathsf{R}}) \leftarrow \mathsf{D}_R(\sigma_1'^{\mathsf{R}}, \ldots, \sigma_t'^{\mathsf{R}})$. If the decoding fails or $(w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}})$ is not $c_R^{\mathsf{err}}/4$-close to $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{n_R}^{\mathsf{R}})$, then set $\widetilde{\mathsf{R}} := \perp$.

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

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

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

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

By the definition of a non-malleable reduction (Definition 4.2.2), in order to complete the proof of Theorem 4.3.1, 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 $f \in \text{Local}_{d_i}^{d_o}$,

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

where G_f is the distribution defined above.

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

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

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

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

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

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

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

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

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

Claim 4.3.1.

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

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

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

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

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

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

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

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

$$\Pr[\mathsf{D}|_{\vec{\sigma}^{\mathsf{L}}}(\vec{\sigma}^{\mathsf{R}}) = \mathsf{D}'|_{\vec{\sigma}^{\mathsf{L}}}(\vec{\sigma}^{\mathsf{R}})] \geq 1 - \mathsf{negl}(n), \tag{4.3.1}$$

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

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

To analyze (4.3.1), we define the following set of instances (which intuitively corresponds to the set of instances on which both $\mathsf{D}|_{\vec{\sigma}^{\mathsf{L}}}$ and $\mathsf{D}'|_{\vec{\sigma}^{\mathsf{L}}}$ 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 $\vec{\sigma} = (\vec{\sigma}^{\mathsf{L}}, \vec{\sigma}^{\mathsf{R}})$, $\Delta(\{\mathsf{D}(\vec{\sigma})\}, \{\mathsf{D}'(\vec{\sigma})\}) = \Delta(\{\mathsf{D}^{\mathsf{R}}|_{\vec{\sigma}^{\mathsf{L}}}(\vec{\sigma}^{\mathsf{R}})\}, \{\mathsf{D}'^{\mathsf{R}}|_{\vec{\sigma}^{\mathsf{L}}}(\vec{\sigma}^{\mathsf{R}})\}) \leq \rho(n_R, \delta, n_{\mathsf{check}})$. 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: $I^{\mathsf{L}} \leftarrow \mathsf{E}_L(\mathsf{L})$, $I^{\mathsf{R}} \leftarrow \mathsf{E}_R(\mathsf{R})$.

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

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

  $g_L(\mathsf{L})$:

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

  $g_R(\mathsf{R})$:

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

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

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

Claim 4.3.2.

$$H_1 \equiv H_2.$$

The claim can be verified by inspection.

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

Claim 4.3.3.

$$H_2 \equiv H_3.$$

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.

4.3.1 Extending to Leaky Local

The construction from Section 4.3 is actually secure against a slightly larger class of tampering functions beyond $\mathrm{Local}_{d_i}^{d_o}$ functions, which we call LL, or "Leaky Local." Notice that the parameters given above (as in observation 1) in fact yield:

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

2. $|\mathcal{S}^{+}_{\mathsf{R}\to\mathsf{L}}| \le d_o \cdot n_L \le n_R \cdot \frac{2c^{\mathsf{sec}}_R}{3}$.

It is not too hard to see that we can leak 1/3 of the security threshold, on both the left and right, to a tampering adversary. Given this leakage, the adversary can then select a tampering function from the subset of Local^{d_o} where all but a fraction of the first n_L bits have input locality d_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 4.3.1. *Let* $\text{LL} \subseteq \{\{0,1\}^{q_L} \times \{0,1\}^{q_R} \to \{0,1\}^{q_L} \times \{0,1\}^{q_R}\}$*,* 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* $\vec{s}^{\mathsf{L}} \in \{0,1\}^{q_L}$*,* h_1 *outputs a subset of* $c^{\mathsf{err}}_L/3$ *of its input bits.*
- *On input* $\vec{s}^{\mathsf{R}} \in \{0,1\}^{q_R}$*,* h_2 *outputs a subset of* $c^{\mathsf{err}}_R/3$ *of its input bits.*
- *On input* $h_1(\vec{s}^{\mathsf{L}}), h_2(\vec{s}^{\mathsf{L}}) \in \{0,1\}^{c^{\mathsf{err}}_L/3} \times \{0,1\}^{c^{\mathsf{err}}_R/3}$*,* f *outputs a circuit* $C : \{0,1\}^{q_L} \times \{0,1\}^{q_R} \to \{0,1\}^{q_L} \times \{0,1\}^{q_R}$*, where* C *has output-locality* d_o*. Of the first* q_L *input bits, all but at most* $c^{\mathsf{err}}_L/3$*-fraction have input-locality at most* d_i*.*

The following corollary can be easily verified.

Corollary 4.3.2. $(\mathsf{E} \circ \mathsf{E}_{\mathrm{SS}}, \mathsf{D}_{\mathrm{SS}} \circ \mathsf{D})$ *is an* $(\text{LL}, \text{SS}_k, \mathsf{negl}(k))$*-non-malleable reduction.*

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

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

Theorem 4.4.1. $(\mathsf{E}', \mathsf{D}')$ *is a* $(\text{Local}^{d_o{}'} \Rightarrow \text{LL}, \mathsf{negl}(n))$*-non-malleable reduction given the following parameters for* $\text{Local}^{d_o{}'}$*:*

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

We construct an encoding scheme $(\mathsf{E}', \mathsf{D}')$ summarized in Figure 4.3 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 $(\text{Local}^{d_o{}'}, \text{LL}, \mathsf{negl}(n))$-non-malleable reduction. Combined with our previous construction, this immediately implies that given a non-malleable encoding scheme $(\mathsf{E}^{\mathsf{ss}}, \mathsf{D}^{\mathsf{ss}})$ for SS_k, the encoding scheme scheme $\widehat{\Pi} = (\widehat{\mathsf{E}^{\mathsf{bd}}}, \widehat{\mathsf{D}^{\mathsf{bd}}})$, where $\widehat{\mathsf{E}^{\mathsf{bd}}}(m) := \mathsf{E}'(\mathsf{E}(\mathsf{E}^{\mathsf{ss}}(m)))$ and $\widehat{\mathsf{D}^{\mathsf{bd}}}(\vec{s}) := \mathsf{D}^{\mathsf{ss}}(\mathsf{D}(\mathsf{D}'(\vec{s})))$ yields the following corollary, a non-malleable code against $\text{Local}^{d_o{}'}$.

Corollary 4.4.1. $(\mathsf{E}', \mathsf{D}')$ *yields, with previous results, a* $(\text{Local}^{\tilde{O}(\sqrt{n})}, k, \mathsf{negl}(k))$*-non-malleable reduction with sublinear rate, where* $n = \Theta(\frac{k^2}{\log^2(k)})$*.*

Remark 4.4.1. *As before, the encoding scheme presented below is independent on the left and right. Therefore, our reduction holds for not just for* $\text{Local}^{d_o'}$ *but additionally any split-state function, independent on each side, trivially.*

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

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

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

- Choose $\zeta \leftarrow \{0,1\}^r$ uniformly at random. Compute $y := \mathsf{prg}(\zeta)$, where $y = y_1, \ldots, y_\tau$. For $i \in [\tau]$, compute Compute $\rho_i := \phi(y_i)$.
- If ρ has less than n_L number of ones, then set h_1, h_2, f all to the constant function 0.
- Otherwise, choose vector $I^{\mathsf{L}} \in \{0,1\}^{\tau + n_R}$ such that $\forall\, i$ such that $1 \leq i \leq \tau$ if $\rho_i = 1$ then $I^{\mathsf{L}}_i = *$ and otherwise, I^{L}_i 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 $r \cdot d_o' \leq c_L^{\mathsf{sec}}/3 \cdot n_L$).
- 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 $r \cdot d_o' \leq c_R^{\mathsf{sec}}/3 \cdot n_R$).

Let prg be a pseudorandom generator for space bounded computations (see Definition 3.4.5), 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 $\mathsf{D}' : \{0,1\}^N \to \{0,1\}^n$.
$\mathsf{E}'(x^{\mathsf{L}} := x_1^{\mathsf{L}}, \ldots, x_{n_L}^{\mathsf{L}}, x^{\mathsf{R}})$:

1. Choose $\zeta \leftarrow \{0,1\}^r$ uniformly at random. Choose $\zeta \leftarrow \{0,1\}^r$ uniformly at random. Compute $\rho := \mathrm{G}(\zeta)$.
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 4.3: The $(\mathrm{Local}^{d_o{}'}, \mathrm{LL}, \mathsf{negl}(n))$-non-malleable reduction $(\mathsf{E}', \mathsf{D}')$

- 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: $((\mu^{\mathsf{L},\zeta}|_{I^{\mathsf{L}}})_{\widetilde{\rho}^*}, \mu^{\mathsf{R},\zeta}|_{I^{\mathsf{L}}})$.
  - If C is in LL, then f outputs C. Otherwise, f outputs the constant function 0.

By the definition of a non-malleable reduction (Definition 4.2.2), in order to complete the proof of Theorem 4.4.1, 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 $\mu \in \mathrm{Local}^{d_o{}'}$,

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

where G_μ is the distribution defined above.

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

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

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

Clearly, by multiplicative Chernoff bounds, if ρ is chosen uniformly at random, then the probability that ρ contains less than n_L ones is negligible. We now show that the probability that $C \notin$ LL is negligible. If $C \notin$ LL, it means that more than $c_L^{\mathsf{sec}}/3$ number of positions i in X^{L} are such that (1) X_i^{L} has "high input locality" (i.e. input locality greater than $12/c_L^{\mathsf{sec}} \cdot d_o{}' = d_i$) (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 $y = y_1, \dots, y_\tau$ (where either $y := \mathsf{prg}(\zeta)$ or y is chosen uniformly at random), $T(y)$ does the following:

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

T can clearly be implemented by a read-once, Finite State Machine (FSM) with $2^{O(\log^2(\tau))}$ number of states. However, note that by Theorem 3.4.1, 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.

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

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

Theorem 4.5.1. $(\mathsf{E}', \mathsf{D}')$ *is a* $(\mathrm{Local}^{d_o} \Rightarrow SS, \mathsf{negl}(n))$*-non-malleable reduction given the following parameters for* Local^{d_o}*:*

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

Putting together Theorem 4.5.1 with Theorems 4.2.1 and 4.2.2, we obtain the following.

Corollary 4.5.1. $(\mathsf{E} \circ \mathsf{E}_{\mathrm{SS}}, \mathsf{D}_{\mathrm{SS}} \circ \mathsf{D})$ *is a* $(\mathrm{Local}^{d_o}, k, \mathsf{negl}(k))$*-non-malleable code with rate* $\Theta(1/d_o)$*, where* $d_o = o(n/\log n)$.

Remark 4.5.1. *Note that* $n = \Theta(d_o k)$*. Thus, for resilience against* $d_o = n^{1-\varepsilon}$ *our codes is of polynomial length* $n = k^{1/\varepsilon}$.

We construct an encoding scheme (E, D) summarized in Figure 4.1 and parametrized below. We then show that the pair (E, D) is a $(\mathrm{Local}^{d_o}, \mathrm{SS}_k, \mathsf{negl}(k))$-non-malleable reduction.

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

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

- $c^{\mathsf{dec}} := 1 - \frac{t}{n_R}$.
- $\delta := \frac{c^{\mathsf{sec}}}{9}$.
- $(\mathsf{E}_L, \mathsf{D}_L)$ parametrized by $(k, n_L, c_L^{\mathsf{err}}, c_L^{\mathsf{sec}}) := (k, c^{\mathsf{rate}}k, c^{\mathsf{err}}, c^{\mathsf{sec}})$ where $c^{\mathsf{err}}, c^{\mathsf{sec}}$, c^{rate} are taken from lemma 4.2.1.
- $n_{\mathsf{check}} := k$.
- $n_R := \lceil \frac{2d_o c^{\mathsf{rate}} k}{\delta c^{\mathsf{dec}} c^{\mathsf{sec}}} \rceil$
- $(\mathsf{E}_R, \mathsf{D}_R)$ parametrized by $(k, n_R, c_R^{\mathsf{err}}, c_R^{\mathsf{sec}}) := (k, n_R, c^{\mathsf{err}}, c^{\mathsf{sec}})$.
- $r := k$
- $\tau := \lceil \frac{30 d_o c^{\mathsf{rate}} k}{(c^{\mathsf{sec}})^2 c^{\mathsf{dec}}} + \frac{1}{\delta} \rceil$
- $n := r + \tau + n_R = O(d_o k)$.

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 d_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

Let $(\mathsf{E}_L, \mathsf{D}_L, \mathsf{Rec}_L)$ be a binary reconstructable probabilistic encoding scheme with parameters $(k, n_L, c_L^{\mathsf{err}}, c_L^{\mathsf{sec}})$ and let $(\mathsf{E}_R, \mathsf{D}_R, \mathsf{Rec}_R)$ be a binary reconstructable probabilistic encoding scheme with parameters $(k, n_R, c_R^{\mathsf{err}}, c_R^{\mathsf{sec}})$. Also, let n_{check} be a parameter. Let prg be a pseudorandom generator for space bounded computations (see Definition 3.4.5), 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 $\rho := 1_L^n 0^{\tau - n_L}$. Otherwise, flip all but the first n_L 1's in ρ to 0.
4. Output ρ.

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

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

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

1. ($\texttt{Recover}$ $\widetilde{\rho}$) 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. ($\texttt{Recover}$ 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 as: For $j \in [\min(\widetilde{\mathsf{num}}, n_L)]$, $\sigma_j^{\mathsf{L}} := \widetilde{X}_{\mathsf{pos}_j}^{\mathsf{L}}$.
4. ($\texttt{plain decoding on left}$) Compute $((w_1^{\mathsf{L}}, \ldots, w_{n_L}^{\mathsf{L}}), \mathsf{L}) \leftarrow \mathsf{D}_L(\sigma_1^{\mathsf{L}}, \ldots, \sigma_{n_L}^{\mathsf{L}})$. If the decoding fails, set $\mathsf{L} := \perp$.
5. ($\texttt{decoding-check on right}$) 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 $((w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}}), \mathsf{R}) \leftarrow \mathsf{D}_R(\sigma_1'^{\mathsf{R}}, \ldots, \sigma_t'^{\mathsf{R}})$. If the decoding fails or $(w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}})$ is not $c_R^{\mathsf{err}}/4$-close to $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{t_R}^{\mathsf{R}})$, set $\mathsf{R} := \perp$.
6. ($\texttt{codeword-check on right}$) 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. ($\texttt{output}$) Output $x := (\mathsf{L}, \mathsf{R})$.

Figure 4.4: THE $(\mathrm{Local}^{d_o}, \mathrm{SS}, \mathsf{negl}(k))$-NON-MALLEABLE REDUCTION (E, D)

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}.

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

- Let $S_{\text{in}}(d_i)$ denote the set of i such that $\vec{s}^{\mathsf{L}}_i$ has input locality greater than d_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 4.5.1. *The event Good_f occurs if for tampering function $f \in \mathrm{Local}^{d_o}$ all of the following hold:*

1. *ρ contains at least n_L ones.*

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

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

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

Claim 4.5.1. *Suppose ρ is chosen truly at random (ones occuring with bias $p = 3n_L/2\tau$). Then for every $f \in \mathrm{Local}^{d_o}$, $\Pr[\mathsf{Good}_f] \geq 1 - \mathsf{negl}(n)$.*

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

1. Recall that $|\rho| = \tau$, $n_L = c^{\mathsf{rate}}k$, $n = O(d_o k)$ 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, $|\mathcal{S}^+_{\mathsf{check}}| \leq d_o n_{\mathsf{check}}$, $|S^+_{\mathrm{in}}(1/\delta \cdot d_o)| \leq \delta n \leq 2\delta\tau$ and $|\mathcal{S}^+_{\mathsf{L}\to\zeta}| \leq d_o r$. Therefore, we can upper bound $|\mathcal{S}^+_{\mathsf{check}} \cup S^+_{\mathrm{in}}(1/\delta \cdot d_o) \cup \mathcal{S}^+_{\mathsf{L}\to\zeta}| \leq 3\delta\tau$.

   We would now like to show that with high probability, $|\mathcal{S}_{\mathsf{check}} \cup S_{\mathrm{in}}(1/\delta \cdot d_o) \cup \mathcal{S}_{\mathsf{L}\to\zeta}| \leq c^{\mathsf{sec}} n_L$. 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 $\mathcal{S}^+_{\mathsf{check}} \cup S^+_{\mathrm{in}}(1/\delta \cdot d_o) \cup \mathcal{S}^+_{\mathsf{L}\to\zeta}$ corresponds to an $\vec{s}_i^{\mathsf{L}}$ input variable (selected by ρ) with independent probability p. We therefore observe that the size of $|\mathcal{S}_{\mathsf{check}} \cup S_{\text{in}}(1/\delta \cdot d_o) \cup \mathcal{S}_{\mathsf{L}\to\zeta}|$ 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 $\Pr[|\mathcal{S}_{\mathsf{check}} \cup S_{\text{in}}(1/\delta \cdot d_o) \cup \mathcal{S}_{\mathsf{L}\to\zeta}| \geq 9n_L \cdot \delta = c^{\mathsf{sec}} \cdot n_L] \leq e^{-\frac{c^{\mathsf{sec}}\cdot n_L}{6}}$, which is negligible.

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

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

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

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

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

$\square$

Now, for every $f \in \text{Local}^{d_o}$, 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}. $(\text{id}|_{I^{\text{mask}}}(I^{\mathsf{L}}).)$
- 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 $\widetilde{\rho} := \text{G}(\widetilde{\zeta})$, 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 $|\mathcal{S}^{J^*}_{\mathsf{L}\to\mathsf{R}} \setminus S_{\text{in}}(d_o/\delta)| = 0$. Note that such J^* must exist since Good_f occurs.
- The split-state tampering function $g := (g_L, g_R) \in \text{SS}_k$ 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 $\vec{s}^{\mathsf{L}} := \mathsf{Rec}(\mathcal{S}_{\mathsf{check}} \cup \mathcal{S}_{\mathsf{L}\to\zeta} \cup S_{\text{in}}(d_o/\delta), I^{\mathsf{L}}, \mathsf{L})$.

Let $(\sigma_1^{\mathsf{L}}, \ldots, \sigma_{n_L}^{\mathsf{L}}) := (f_X|_{\zeta, I^{\text{mask}}, I^{\mathsf{R}}}(\vec{s}^{\mathsf{L}}))_{\widetilde{U}}$. 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 $((w_1^{\mathsf{L}}, \ldots, w_{n_L}^{\mathsf{L}}), \widetilde{\mathsf{L}}) \leftarrow \mathsf{D}_L(\sigma_1^{\mathsf{L}}, \ldots, \sigma_{n_L}^{\mathsf{L}})$. If the decoding fails, set $\widetilde{\mathsf{L}} := \perp$.

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

$g_R(\mathsf{R})$:

1. (`apply tampering and decoding-check on right`) Let

   $\vec{s}^{\mathsf{R}} = (s_1^{\mathsf{R}}, \ldots, s_{n_R}^{\mathsf{R}}) := \mathsf{Rec}(\mathcal{S}_{\mathsf{R}\to\mathsf{L}}, I^{\mathsf{R}}, \mathsf{R})$.

   Let $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{n_R}^{\mathsf{R}}) := f^{\mathsf{R}}|_{\zeta, I^X}(\vec{s}^{\mathsf{R}})$. 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 $((w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}}), \widetilde{\mathsf{R}}) \leftarrow \mathsf{D}_R(\sigma_1'^{\mathsf{R}}, \ldots, \sigma_t'^{\mathsf{R}})$. If the decoding fails or $(w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}})$ is not $c_R^{\mathsf{err}}/4$-close to $(\sigma_1^{\mathsf{R}}, \ldots, \sigma_{n_R}^{\mathsf{R}})$, then set $\widetilde{\mathsf{R}} := \perp$.

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

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

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

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

Remark 4.5.2. *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 4.2.2), in order to complete the proof of Theorem 4.3.1, we must show that (E, D) have the following properties:

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

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

   where G_f is the distribution defined above.

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

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

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

Firstly, we will argue that with high probability, the pseudorandomness of the

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

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

So for our final hybrid, we will use a technique from [44] to show that we can decode from a "clean" portion of $\vec{\sigma}^{\mathsf{R}}$, J^*. This is possible because, by Claim 4.5.2 (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 4.5.2. *In any given tampering experiment, with tampering function $f \in\in \mathrm{Local}^{d_o}$, the event Good_f holds with all but negligible probability, even when ρ generated according to Figure 4.4.*

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

$S_{\text{in}}(d_o/\delta)$. ($\mathcal{S}_{\mathsf{L}\to\zeta}, \mathcal{S}_{\mathsf{check}}$ 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 $d_o n_R + \delta n + d_o r \leq c^{\mathsf{sec}}\tau/3$ number of positions in X^{L} in $\mathcal{S}_{\mathsf{check}} \cup S_{\text{in}}(d_o/\delta) \cup \mathcal{S}_{\mathsf{L}\to\zeta}$. 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) $X_i^{\mathsf{L}} \in \mathcal{S}_{\mathsf{check}} \cup S_{\text{in}}(d_o/\delta) \cup \mathcal{S}_{\mathsf{L}\to\zeta}$ (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 $S_{\text{in}}(d_o/\delta) \cup \mathcal{S}_{\mathsf{L}\to\zeta} \cup \mathcal{S}_{\mathsf{check}}$.

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

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

T can clearly be implemented by a read-once, Finite State Machine (FSM) with $2^{O(\log^2(\tau))}$ number of states. However, note that by Theorem 3.4.1, 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. $\square$

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

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 4.5.3.

$$H_0 \equiv H_1.$$

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 $S_{\text{in}}(d_o/\delta)$.
2. Choose $I^{\mathsf{L}} \in \{0,1\}^{n_L}, I^{\mathsf{R}} \in \{0,1\}^{n_R}$ uniformly at random.
3. Compute $\vec{s}^{\mathsf{L}} = (s_1^{\mathsf{L}}, \ldots, s_{n_L}^{\mathsf{L}}) \leftarrow \mathsf{Rec}(\mathcal{S}_{\mathsf{check}} \cup \mathcal{S}_{\mathsf{L}\to\zeta} \cup S_{\text{in}}(d_o/\delta), I^{\mathsf{L}}, \mathsf{L})$ and $\vec{s}^{\mathsf{R}} = (s_1^{\mathsf{R}}, \ldots, s_{n_R}^{\mathsf{R}}) \leftarrow \mathsf{Rec}(\mathcal{S}_{\mathsf{R}\to\mathsf{L}}, I^{\mathsf{R}}, \mathsf{R})$.
4. Choose $\zeta \leftarrow \{0,1\}^r$ uniformly at random. Compute $\rho := \mathrm{G}(\zeta)$.
5. For $j \in [n_L]$, let pos_j denote the j-th position i such that $\rho_i = 1$.
6. Let $X^{\mathsf{L}} \in \{0,1\}^{\tau}$ be defined in the following way: For $j \in [n_L]$, $X^{\mathsf{L}}_{\mathsf{pos}_j} := s^{\mathsf{L}}_j$. In all other locations, X^{L}_i is set uniformly at random.
7. Output the encoding $(\zeta, X^{\mathsf{L}}, \vec{s}^{\mathsf{R}})$.

Claim 4.5.4.

$$H_1 \equiv H_2.$$

By Claim 4.5.2, 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 $j \in [\min(\widetilde{\mathsf{num}}, n_L)]$, $\sigma^{\mathsf{L}}_j := \widetilde{X}^{\mathsf{L}}_{\mathsf{pos}_j}$.

4. (`plain decoding on left`) Compute $((w^{\mathsf{L}}_1, \ldots, w^{\mathsf{L}}_{n_L}), \mathsf{L}) \leftarrow \mathsf{D}_L(\sigma^{\mathsf{L}}_1, \ldots, \sigma^{\mathsf{L}}_{n_L})$. If the decoding fails, set $\mathsf{L} :=\perp$.

5. (`decoding-check on right`) Let $t := \lceil n_R(1 - c^{\mathsf{err}}_R/4) \rceil$ 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 $\ell = 1, \ldots, t$. Set $\sigma'^{\mathsf{R}}_\ell := 0$ for $\ell = t + 1, \ldots, n_R$. Compute $((w^{\mathsf{R}}_1, \ldots, w^{\mathsf{R}}_{n_R}), \mathsf{R}) \leftarrow \mathsf{D}_R(\sigma'^{\mathsf{R}}_1, \ldots, \sigma'^{\mathsf{R}}_t)$. If the decoding fails or $(w^{\mathsf{R}}_1, \ldots, w^{\mathsf{R}}_{n_R})$ is not $c^{\mathsf{err}}_R/4$-close to $(\sigma^{\mathsf{R}}_1, \ldots, \sigma^{\mathsf{R}}_{t_R})$, set $\mathsf{R} :=\perp$.

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

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 $|\mathcal{S}_{\mathsf{L}\to\mathsf{R}}^{J^*} \cap S_{\text{in}}(\bar{d}_o/\delta)| = 0$. Compute $((w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}}), \mathsf{R}) \leftarrow \mathsf{D}_R(\sigma_1'^{\mathsf{R}}, \ldots, \sigma_{t_R}'^{\mathsf{R}})$. If the decoding fails or $(w_1^{\mathsf{R}}, \ldots, w_{n_R}^{\mathsf{R}})$ is not $c_R^{\mathsf{err}}/4$-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 4.5.5.

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

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 4.3.1.

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

Claim 4.5.6.

$$H_3 \equiv H_4.$$

This claim follows by inspection.

Chapter 5: Non-Malleable Codes from Average-Case Hardness: AC^0, Decision Trees, and Streaming Space-Bounded Tampering

5.1 Introduction

In this chapter, we present general frameworks for constructing non-malleable codes for encoding one and multi-bits against various tampering classes $\mathcal{F}$ for which average case hardness results are known. Our frameworks (one for single-bit and one for multi-bit) include both a generic construction, which requires that certain underlying primitives are instantiated in a suitable way, as well as a proof "template." Our frameworks are inspired by the well-known double-encryption paradigm for constructing CCA2-secure public key encryption schemes [99, 102, 111]. And although we rely on techniques that are typically used in the cryptographic setting, we instantiate our framework for particular tampering classes $\mathcal{F}$ in both the computational setting and in the information theoretic one. For the computational setting, our results rely on computational assumptions, and require a common-reference string (CRS), which the adversary can see before selecting the tampering function (as typical in other NMC works using CRS or random oracles). For the information theoretic setting, our results do not require CRS nor any computational assumption (as the primitives in our framework can be instantiated information theoretically). Our general theorem statements provide sufficient conditions for achieving NMC against a class $\mathcal{F}$. Somewhat informally, the main such condition, especially for the one-bit framework, is that there are sufficiently strong average-case hardness results known for the class $\mathcal{F}$.   In particular, we obtain the following results, where all the constructions are efficient and, for the multi-bit NMC, the achieved rate is $1/\operatorname{poly}(m)$ where m is the length of the message being encoded.

- **Constructions for AC^0 tampering:** We obtain computational NMC in the CRS model against AC^0 tampering. Our constructions require public key encryption schemes with decryption in AC^0, which can be constructed e.g. from exponential hardness of learning parity with noise [26], as well as non-interactive zero knowledge (NIZK), which can be constructed in the CRS model from enhanced trapdoor permutations.

  Previous results by Chattopadhyay and Li [36] achieve NMC for AC^0 with information theoretic security (with no CRS), but are inefficient, with super-polynomial rate.

- **Constructions for bounded-depth decision trees:** We obtain computational NMC in the CRS model against tampering with bounded-depth decision trees. Our construction requires the same computational assumptions as the AC^0 construction above. The depth of the decision tree we can handle is m^{ϵ}, where m is the number of bits being encoded, and ϵ is any constant.

No results for this class were previously known.

- **Constructions for streaming, space-bounded tampering:** We obtain *unconditional* non-malleable codes against streaming, space-bounded tampering, where the tampering function is represented by a read-once, bounded-width branching program. Our construction does not require CRS or computational assumptions.

  No NMC results for this standard complexity theoretic class were previously known. However, this tampering class can be viewed as a subset (or the intersection) of the space bounded class considered by Faust et al. [70] (who don't limit the adversary to be streaming), and the block-wise tampering class considered by Chandran et al. [31] (who don't bound the adversary's space, but don't give security in the event that decoding fails). In both cases there cannot be NMC with the standard notion of security, and so those previous works must relax the security requirement (and [70] also relies on a random oracle). In contrast, we achieve standard (in fact, even stronger) notion of NMC, without random oracle (nor CRS, nor any computational assumption) for our class.

- **Additional Constructions:** We also briefly note two additional applications of our paradigm as proof of concept. Both complexity classes can be represented circuits of size $O(n^c)$ for some fixed c, a class which [73] provide non-malleable codes for in the CRS model, *without* computational assumptions. We include these results here, merely to show the applicability of our framework to general correlation bounds; for example strong correlation bounds against $\mathsf{ACC}^0[p]$ or TC^0 are likely immediately lead to non-malleable codes against the same classes using our framework.

1. Under the same assumptions invoked in the constructions against AC^0 and bounded-depth decision trees we obtain computational NMC in the CRS model against tampering with small threshold circuits: threshold circuits with depth d and $n^{1+\epsilon}$ wires.
2. Assuming any public key encryption scheme and zk-SNARKs, we obtain computational NMC in the CRS model against tampering by Turing Machines running in time $O(n^k)$, where k is a constant. However, we should note that these codes have weak tampering guarantees: tampering experiments with respect to different messages are only polynomially close to one another.

5.1.1 Technical Overview

We begin by describing our computational NMC construction (in the CRS model) for one-bit messages secure against tampering in AC^0, which will give the starting point intuition for our results. We then show how the AC^0 construction can be modified to derive a general template for constructing NMC for one-bit messages secure against a wider range of tampering classes $\mathcal{F}$, and discuss various classes $\mathcal{F}$for which the template can be instantiated. We then discuss how the template can be extended to achieve NMC for multi-bit messages secure against a wide range of tampering classes $\mathcal{F}$. Finally, we discuss some particular instantiations of our multi-bit template, including our constructions of computational NMC (in the CRS model) against tampering in AC^0 and against bounded-depth decision trees, as well as our *unconditional* NMC (with no CRS) against streaming tampering adversaries with bounded memory.

The starting point: Computational NMC against AC^0 for one-bit messages. The idea is to use a very similar paradigm to the Naor and Yung paradigm for CCA1 encryption [102] (later extended to achieve CCA2 [99, 111]), using double encryption with simulation-sound NIZK. The main observation is that using the tableaua method, we can convert *any* NIZK proof system with polynomial verification into a NIZK proof system with a verifier in AC^0.

We also need a PKE scheme with perfect correctness and decryption in AC^0(this can be constructed using the transformation of Dwork et al. [63] on top of the scheme of Bogdanov and Lee [26]).

We now sketch (a slightly simplified version of) the NM encoding scheme:

The CRS will contain a public key PK for an encryption scheme

$\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ as above, and a CRS for a NIZK. For $b \in \{0,1\}$, Let $\mathcal{D}_b$ denote the distribution over $x_1, \ldots, x_n \in \{0,1\}^n$ such that $x_1, \ldots, x_n$ are uniform random, conditioned on the parity of the bits being equal to b.

To encode a bit b:

1. Randomly choose bits $x_1, \ldots, x_n$ from $\mathcal{D}_b$

2. Compute $c_1 \leftarrow \mathsf{Encrypt}_{\text{PK}}(x_1), \ldots, c_n \leftarrow \mathsf{Encrypt}_{\text{PK}}(x_n)$ and $c \leftarrow \mathsf{Encrypt}_{\text{PK}}(b)$.

3. Compute n NIZK proofs $\pi_1, \ldots, \pi_n$ that $c_1, \ldots, c_n$ are encryptions of bits $x_1, \ldots, x_n$.

4. Compute a NIZK proof π that there exists a bit b' such that the plaintexts underlying $c_1, \ldots, c_n$ are in the support of $\mathcal{D}_{b'}$ and b' is the plaintext underlying c.

5. Compute tableaus $T_1, \ldots, T_n$ of the computation of the NIZK verifier on $\pi_1, \ldots, \pi_n$.

6. Compute a tableau T of the computation of the NIZK verifier on proof π.

7. Output $(c_1, \ldots, c_n, c, T, (x_1, T_1), \ldots, (x_n, T_n))$.

To decode $(c_1, \ldots, c_n, c, T, (x_1, T_1), \ldots, (x_n, T_n))$**:**

1. Check the tableaus $T_1, \ldots, T_n, T$.

2. If they all accept, output the parity of $x_1, \ldots, x_n$.

In the proof we will switch from an honest encoding of b to a simulated encoding and from an honest decoding algorithm to a simulated decoding algorithm. At each point we will show that the decodings of tampered encodings stay the same. Moreover, if, in the final hybrid, decodings of tampered encodings depend on b, we will use this fact to build a circuit in AC^0, whose output is correlated with the parity of its input, reaching a contradiction. In more detail, in the first hybrid we switch to simulated proofs. Then we switch $c_1, \ldots, c_n, c$, in the "challenge" encoding to encryptions of garbage $c'_1, \ldots, c'_n, c'$, and next we switch to an alternative decoding algorithm *in* AC^0 , which requires the trapdoor SK (corresponding to the public key PK which is contained in the CRS).

Alternative Decoding Algorithm:

To decode $(c_1, \ldots, c_n, c, T, (x_1, T_1), \ldots, (x_n, T_n))$**:**

1. check the tableaus $T_1, \ldots, T_n, T$
2. If it accepts, output the decryption of c using trapdoor SK.

In the final hybrid, the simulator will not know the parity of $x_1, \ldots, x_n$ in the challenge encoding and will have received precomputed $T_1^0, T_1^1, \ldots, T_n^0, T_n^1, T$ as non-uniform advice, where T is a simulated proof of the statement "the plaintexts underlying $c_1', \ldots, c_n'$ and the plaintext underlying c' have the same parity" and for $i \in [n], \beta \in \{0,1\}, T_i^\beta$ is a simulated proof of the statement "c_i' is an encryption of the bit β".

We will argue by contradiction that if the decoding of the tampered encoding is correlated with the parity of $x_1, \ldots, x_n$ then we can create a circuit whose output is correlated with the parity of its input in AC^0 . Specifically, the AC^0 circuit will have the crs, SK, precomputed $c_1', \ldots, c_n', c', T, T_1^0, T_1^1, \ldots, T_n^0, T_n^1$ and adversarial tampering function f hardwired in it. It will take $x_1, \ldots, x_n$ as input. It will compute the simulated encoding in AC^0 by selecting the correct tableaus: $T_1^{x_1}, \ldots, T_n^{x_n}$ according to the corresponding input bit. It will then apply the adversarial tampering function (in AC^0), perform the simulated decoding (in AC^0) and output a guess for the parity of $x_1, ..x_n$ based on the result of the decoding. Clearly, if the decoding in the final hybrid is correlated with parity, then we have constructed a distribution over AC^0 circuits such that w.h.p. over choice of circuit from the distribution, the output of the circuit is correlated with the parity of its input. This contradicts known results on the hardness of computing parity in AC^0 .

A general template for one-bit NMC. The above argument can be used to derive a template for the construction/security proof of NMC against more general classes $\mathcal{F}$. The idea is to derive a high-level sequence of hybrid distributions and corresponding *minimal* requirements for proving the indistinguishability of consecutive hybrids. We can now instantiate the tampering class $\mathcal{F}$, "hard distributions" $(\mathcal{D}_0, \mathcal{D}_1)$, encryption scheme and NIZK proof in any way that satisfies these minimal requirements. Note that each hybrid distribution is a distribution over the output of the tampering experiment. Therefore, public key encryption and NIZK against arbitrary PPT adversaries may be too strong of a requirement. Indeed, it is by analyzing the exact security requirements needed to go from one hybrid to the other that (looking ahead) we are able to remove the CRS and all computational assumptions from our construction of NMC against streaming adversaries with bounded memory. In addition, we can also use our template to obtain constructions (in the CRS model and under computational assumptions) against other tampering classes $\mathcal{F}$.

Extending the template to multi-bit NMC. The construction for AC^0 given above and the general template do not immediately extend to multi-bit messages. In particular, encoding m bits by applying the parity-based construction bit-by-bit fails, even if we use the final proof T to "wrap together" the encodings of multiple individual bits. The problem is that the proof strategy is to entirely decode the tampered codeword and decide, based on the results, whether to output 0 or 1 as the guess for the parity of some $x_1, \ldots, x_n$. But if we encode many bits, $b_1, \ldots, b_m$, then the adversary could maul in such a way that the tampered codeword decodes to $b'_1, \ldots, b'_m$ where each of b'_i is *individually* independent of the parity of the corresponding $x^i_1, \ldots, x^i_n$, but taken as a whole, the entire output may be correlated. As a simple example, the attacker might maul the codeword so that it decodes to $b'_1, \ldots, b'_m$ that are uniform subject to satisfying $b'_1 \oplus \cdots \oplus b'_m = b_1 \oplus \cdots \oplus b_m$. Clearly, there is a correlation here between the input and output, but we cannot detect this correlation in AC^0, since detecting the correlation itself seems to require computing parity!

In the case of parity (and the class AC^0), the above issue can be solved by setting m sufficiently small (but still polynomial) compared to n. We discuss more details about the special case of parity below. However, we would first like to explain how the general template must be modified for the multi-bit case, given the above counterexample. Specifically, note that the difficulty above comes into play only in the final hybrid. Thus, we only need to modify the final hybrid slightly and require that for any Boolean function F over m variables, it must be the case that the composition of F with the simulated decoding algorithm is in a computational class that still cannot distinguish between draws $x_1, \ldots, x_n$ from $\mathcal{D}_0$ or $\mathcal{D}_1$. While the above seems like a strong requirement, we show that by setting m much smaller than n, we can still obtain meaningful results for classes such as AC^0 and bounded-depth decision trees.

Multi-bit NMC against AC^0. If we want to encode m bits, for each of the underlying encodings $i \in [m]$, we will use $n :\approx m^3$ bits: $\vec{x}^i = x_1^i, \ldots, x_n^i$. To see why this works, we set up a Hybrid argument, where in each step we will fix all the underlying encodings except for a single one: $\vec{x} = x_1, \ldots, x_n$, which we will switch from having parity 0 to having parity 1. Therefore, we can view C—the function computing the output of the tampering experiment in this hybrid—to be a function of variables $\vec{x} = x_1, \ldots, x_n$ only (everything else is constant and "hardwired"). For $i \in [m]$, let C_i denote the i-th output bit of C. We use $\mathsf{PAR}(\vec{x})$ to denote the parity of $\vec{x}$.

Now, for any Boolean function F over m variables, consider $F(C_1(\vec{x}), C_2(\vec{x}), \ldots, C_m(\vec{x}))$, where we are simply taking an arbitrary Boolean function F of the decodings of the individual bits. Our goal is to show that $F(C_1(\vec{x}), C_2(\vec{x}), \ldots, C_m(\vec{x}))$ is not correlated with parity of $\vec{x}$. Consider the Fourier representation of $F(y_1, \ldots, y_m)$. This is a linear combination of parities of the input variables $y_1, \ldots, y_m$, denoted $\chi_S(y_1, \ldots, y_m)$, for all subsets $S \in \{0,1\}^m$. (See here [56]).

On the other hand, $F(C_1(\vec{x}), C_2(\vec{x}), \ldots, C_m(\vec{x}))$ is a Boolean function over $n \approx m^3$ variables (i.e. a linear combination over parities of the input variables $x_1, \ldots, x_n$, denoted $\chi_{S'}(x_1, \ldots, x_n)$, for all subsets $S' \in \{0,1\}^n$). A representation of $F(C_1(\vec{x}), C_2(\vec{x}), \ldots, C_m(\vec{x}))$ can be obtained by taking each term $\hat{F}(S)\chi_S(y_1, \ldots, y_m)$ in the Fourier representation of F and composing with $C_1, \ldots, C_m$ to obtain the term $\hat{F}(S)\chi_S(C_1(\vec{x}), C_2(\vec{x}), \ldots, C_m(\vec{x}))$. Since, by well-known properties of the Fourier transform, $|\hat{F}(S)| \leq 1$, we can get an upper bound on the correlation of

$F(C_1(\vec{x}), C_2(\vec{x}), \ldots, C_m(\vec{x}))$ and $\mathsf{PAR}(\vec{x})$, by summing the correlations of each function $\chi_S(C_1(\vec{x}), C_2(\vec{x}), \ldots,$

$C_m(\vec{x}))$ and $\mathsf{PAR}(\vec{x})$. Recall that the correlation of a Boolean function g with $\mathsf{PAR}(\vec{x})$ is by definition, exactly the Fourier coefficient of g corresponding to parity function $\chi_{[n]}$. Thus, to prove that the correlation of $\chi_S(C_1(\vec{x}), C_2(\vec{x}), \ldots, C_m(\vec{x}))$ and $\mathsf{PAR}(\vec{x})$ is low, we use the fact that $\chi_S(C_1(\vec{x}), C_2(\vec{x}), \ldots, C_m(\vec{x}))$ can be computed by a (relatively) low depth circuit. To see this, note that each C_i is in AC^0 and so has low depth, moreover, since S has size at most m, we only need to compute parity over m variables, which can be done in relatively low depth when $m \ll n$. We now combine the above with Fourier concentration bounds for low-depth circuits [117]. Ultimately, we prove that for each S, the correlation of $\chi_S(C_1(\vec{x}), C_2(\vec{x}), \ldots, C_m(\vec{x}))$ and $\mathsf{PAR}(\vec{x})$, is less than $1/2^{m(1+\delta)}$, where δ is a constant between 0 and 1. This means that we can afford to sum over all 2^m terms in the Fourier representation of F and still obtain negligible correlation.

Multi-bit NMC against bounded-depth decision trees. Our result above extends to bounded-depth decision trees by noting that (1) If we apply a random restriction (with appropriate parameters) to input $x_1, \ldots, x_n$ then, w.h.p. the AC^0 circuit used to compute the output of the tampering experiment collapses to a bounded-depth decision tree of depth $m^\varepsilon - 1$; (2) on the other hand, again choosing parameters of the random restriction appropriately, $\mathsf{PAR}(x_1, \ldots, x_n)$ collapses to parity over at least $m^{1+\varepsilon}$ variables; (3) any Boolean function over m variables can be computed by a decision tree of depth m; (4) the composition of a depth-$m^\varepsilon - 1$ decision tree and depth-m decision tree yields a decision tree of depth at most $(m^{\varepsilon} - 1)(m) < m^{1+\varepsilon}$. Finally, we obtain our result by noting that decision trees of depth less than $m^{1+\varepsilon}$ are uncorrelated with parity over $m^{1+\varepsilon}$ variables.

Unconditional NMC (with no CRS) against bounded, streaming tampering. Recently, Raz [110] proved that learning parity is hard for bounded, streaming adversaries. In particular, this gives rise to hard distributions $\mathcal{D}_b, b \in \{0, 1\}$ such that no bounded, streaming adversary can distinguish between the two. $\mathcal{D}_b$ corresponds to choosing a random parity χ_S, outputting random examples $(\vec{x}, \chi_S(\vec{x}))$ and then outputting $\vec{x}^*$ such that $\chi_S(\vec{x}^*)$ is equal to b. The above also yields an unconditional, "parity-based" encryption scheme against bounded, streaming adversaries. Note, however, that in order to decrypt (without knowledge of the secret key), we require space beyond the allowed bound of the adversary. Given the above, we use $\mathcal{D}_b, b \in \{0, 1\}$ as the hard distributions in our construction and use the parity-based encryption scheme as the "public key encryption scheme" in our construction. Thus, we get rid of the public key in the CRS (and the computational assumptions associated with the public key encryption scheme).

To see why this works, note that in the hybrid where we require semantic security of the encryption scheme, the decryption algorithm is not needed for decoding (at this point the honest decoding algorithm is still used). So essentially we can set the parameters for the encryption scheme such that the output of the Tampering experiment in that hybrid (which outputs the decoded value based on whether $x_1, .., x_n$ is in the support of $\mathcal{D}_0$ or $\mathcal{D}_1$) can be computed in a complexity class that is too weak to run the decryption algorithm. On the other hand, we must also consider the later hybrid where we show that the output of the Tampering experiment can be computed in a complexity class that is too weak to distinguish $\mathcal{D}_0$ from $\mathcal{D}_1$. In this hybrid, we do use the alternate decoding procedure. But now it seems that we need decryption to be contained in a complexity class that is too weak to decide whether $x_1, \ldots, x_n$ is in the support of $\mathcal{D}_0$ or $\mathcal{D}_1$, while previously we required exactly the opposite! The key insight is that since we are in the streaming model and since (1) the simulated ciphertexts $(c'_1, \ldots, c'_n, c')$ in this hybrid contain no information about $x_1, \ldots, x_n$ and (2) the simulated ciphertexts precede $x_1, \ldots, x_n$, the output of the tampering function in blocks containing ciphertexts *does not depend on* $x_1, \ldots, x_n$ *at all.* So the decryption of the tampered ciphertexts can be given as non-uniform advice, instead of being computed on the fly, and we avoid contradiction.

In order to get rid of the CRS and computational assumption for the NIZK, we carefully leverage some additional properties of the NMC setting and the streaming model. First, we consider cut-and-choose based NIZK's (based on MPC-in-the-head), where the Verifier is randomized and randomly checks certain locations or "slots" in the proof to ensure soundness. Specifically, given a Circuit-SAT circuit C and witness w, the prover will secret share $w := w_1 \oplus \cdots \oplus w_\ell$ and run an MPC protocol among ℓ parties (for constant ℓ), where Party i has input w_i and the parties are computing the output of $C(w_1 \oplus \cdots \oplus w_\ell)$. The prover will then "encrypt" each view of each party in the MPC protocol, using the parity-based encryption scheme described above and output this as the proof. This is then repeated λ times (where λ is security parameter). The Verifier will then randomly select two parties from each of the λ sets, decrypt the views and check that the views correspond to the output of 1 and are consistent internally and with each other.

We next note that in our setting, the NIZK simulator can actually know the randomness used by the Verifier. This is because the simulated codeword and the decoding are done by the same party in the NMC security experiment. Therefore, the level of “zero-knowledge” needed from the simulation of the NIZK is in-between honest verifier and malicious. This is because the adversary can still use the tampering function to “leak” information from the unchecked slots of the proof to the checked slots, while a completely honest verifier would learn absolutely nothing about the unchecked slots. In order to switch from a real proof to a simulated proof, we fill in unchecked slots one-by-one with parity-based encryptions of garbage. We must rely on the fact that a bounded, streaming adversary cannot distinguish real encryptions from garbage encryptions in order to argue security. Specifically, since we are in the bounded streaming model, we can argue that the adversary can only “leak” a small amount of information from the unchecked slots to the checked slots. This means that the entire output of the experiment can be simulated by a bounded, streaming adversary, which in turn means that the output of the experiment must be indistinguishable when real, unchecked encodings are replaced with encodings of garbage. Arguing simulation soundness, requires a similar argument, but more slots are added to the proof and slots in an honest proof are only filled if the corresponding position in the bit-string corresponding to the statement to be proven is set to 1. We encode the statement in such a way that if the statement changes, the adversary must switch an unfilled slot to a filled slot. Intuitively, since the bounded streaming attacker can only carry over a small amount of information from previous slots, this will be as difficult as constructing a new proof from scratch.

5.2 Definitions

In this section we will provide some more definitions and background required for our constructions and results presented in chapter 5.

5.2.1 Incompressible Functions

Definition 5.2.1. *A function $\psi : \{0,1\}^n \to \{0,1\}$ is* ε-incompressible *by function $f : \{0,1\}^n \to \{0,1\}^\ell$ if for every function $h : \{0,1\}^\ell \to \{0,1\}$, for uniform random $x \in \{0,1\}^n$, $\Pr\left[h(f(x)) = \psi(x)\right] \leq \frac{1}{2} + \varepsilon$.*

We say, ψ is (ℓ, ε)-incompressible *by class $\mathcal{F}$, if for every $f : \{0,1\}^n \to \{0,1\}^\ell \in \mathcal{F}$, ψ is ε-incompressible by f.*

Following theorem was proved by [62]

Theorem 5.2.1 (Parity is incompressible [62])**.** *Let $0 < \delta < 1$ be a constant. Parity is $(n^\delta, 2^{-\Omega(n^{\frac{1-\delta}{d}})})$ - incompressible by circuits of depth $d \geq 2$ of size $2^{O(n^{\frac{1-\delta}{d}})}$.*

5.2.2 Proof Systems for Circuit SAT

We now consider proof of knowledge systems for Circuit SAT, where the prover and/or verifier have limited computational resources.

Definition 5.2.2 (Proof of Knowledge Systems for Circuit SAT with Computationally Bounded Prover/Verifier)**.** *For a circuit C, let $\mathcal{L}(C)$ denote the* *set of strings x such that there exists a witness w such that $C(x,w) = 1$. For a class $\mathcal{C}$, let $\mathcal{L}(\mathcal{C})$ denote the set $\{\mathcal{L}(C) \mid C \in \mathcal{C}\}$. $\Pi = (\mathsf{P}, \mathsf{V})$ is a Circuit SAT proof system for the class $\mathcal{L}(\mathcal{C})$ with prover complexity $\mathcal{D}$ and verifier complexity $\mathcal{E}$ if the following are true:*

- *For all $C \in \mathcal{C}$ and all valid inputs (x,w) such that $C(x,w) = 1$, $\mathsf{P}(C,\cdot,\cdot)$ can be computed in complexity class $\mathcal{D}$.*
- *For all $C \in \mathcal{C}$, $\mathsf{V}(C,\cdot,\cdot)$ can be computed in complexity class $\mathcal{E}$.*
- *Completeness: For all $C \in \mathcal{C}$ and all (x,w) such that $C(x,w) = 1$, we have $\mathsf{V}(C,x,\mathsf{P}(C,x,w)) = 1$*
- *Extractability: For all (C,x,π), if $\Pr_r[\mathsf{V}(C,x,\pi;r) = 1]$ is non-negligible, then given (C,x,π) it is possible to efficiently extract w such that $C(x,w) = 1$.*

We construct Circuit SAT proof systems for the class $\mathcal{L}(\mathsf{P/poly})$ with verifier complexity AC^0 in this section. We also construct Circuit SAT proof systems for the class. $\mathcal{L}(\mathsf{P/poly})$ with streaming verifier

5.2.2.1 Circuit SAT proof system for the class $\mathcal{L}(\mathcal{C})$ with prover complexity $\mathcal{D}$ and verifier complexity AC^0 .

- $\mathsf{P}(C,x,w)$ the prover simply outputs a tableau T of the computation $C(x,w) = 1$.
- $\mathsf{V}(C,x,T)$ the verifier computes an AND of all the local checks.

Completeness clearly holds. To show extractability, note that the inputs to the tableau T correspond to x, w. Thus if tableau T accepts then the extractor can simply output those inputs corresponding to w.

Given the above, we have the following theorem:

Theorem 5.2.2. *Assuming the existence of same-string, weak one-time simulation sound NIZK with deterministic verifier, there exists same-string, weak one-time simulation sound NIZK with verifier in* AC^0 *.*

5.2.2.2 Circuit SAT proof system for the class $\mathcal{L}(\mathcal{C})$ with prover complexity $\mathcal{D}$ and streaming verifier.

- $\mathsf{P}(C, x, w)$ the prover computes a tableau T of the computation $C(x, w) = 1$. Let d denote the depth of the tableau T. For each level $i \in [d]$, the i-th level, T_i, consisting of ℓ gates is the following ordered tuple:

$$[(G_i^j, in_i^{j,a}, in_i^{j,b}, out_i^j)]_{j \in \ell},$$

  where G_i^j denotes the j-th gate at that level, $(in_i^{j,a}, in_i^{j,b})$ denote the j-th pair of input wires at that level and out_i^j denotes the j-th output wire at that level. For simplicity of notation, we assume that the input wires to the first level, T_1 consist only of x, and that wires corresponding to the input w will occur as outputs of level T_1. The P outputs $(T_1, \ldots, T_d)$.

- $\mathsf{V}(C, x, T_1, \ldots, T_d)$ the verifier chooses k at random and computes $h_0 = h_k(x)$, where h is a universal hash function. For each level $i \in [d]$, the verifier then does the following:

- Parse $T_i = [(G_i^j, in_i^{j,a}, in_i^{j,b}, out_i^j)]_{j \in \ell}$.
- For $j \in [\ell]$, (1) Check consistency of the gate's computation, (2) Add $(in_i^{j,a}, in_i^{j,b})$ to the streaming computation of the hash $h_k([(in_i^{j,a}, in_i^{j,b})]_{j \in \ell})$, (3) Add out_i^j to the streaming computation of the hash $h_k([out_i^j]_{j \in \ell})$.
- Check that $h_k([(in_i^{j,a}, in_i^{j,b})]_{j \in \ell}) = h_{i-1}$.
- Set $h_i := h_k([out_i^j]_{j \in \ell})$.
- If any of the above checks fail, abort and output 0.

If all checks succeed, the verifier outputs 1.

Completeness clearly holds. To show extractability, note that the only way the inputs/outputs of level T_1 do not correspond to x, w such that $C(x, w) = 1$ and yet all checks pass is if the proof ouputted by the prover consists of consecutive levels T_i, T_{i+1} such that $h_k([(in_{i+1}^{j,a}, in_{i+1}^{j,b})]_{j \in \ell}) = h_k([out_i^j]_{j \in 2\ell})$ but $[(in_{i+1}^{j,a}, in_{i+1}^{j,b})]_{j \in \ell} \neq [out_i^j]_{j \in 2\ell}$. The probability over choice of k that this occurs for a single pair of consecutive levels is $1/2^{2\ell}$, since h is universal. So the probability it occurs for any pair of consecutive levels is at most $d/2^{2\ell}$, which is negligible.

Theorem 5.2.3 ([110]). *For any $c < \frac{1}{20}$, there exists $\alpha > 0$, such that the the following holds: Let $x \overset{u}{\leftarrow} \{0,1\}^n$. Let $m \leq 2^{\alpha n}$. Let A be an algorithm that is given as input a stream of samples, $(a_1, b_1), \ldots, (a_m, b_m)$, where each a_t is uniformly* *distributed over $\{0,1\}^n$ and for every t, $b_t = a_t \cdot x$. Assume A uses at most cn^2 memory bits and outputs a string $\tilde{x} \in \{0,1\}^n$. Then,* $\Pr[\tilde{x} = x] \leq O(2^{-\alpha n})$.

Lemma 5.2.1 (Inner Product is a strong extractor [108])**.** *Let X, Y be random variables over $\{0,1\}^n$ such that $H_\infty(X) \geq k_X$ and $H_\infty(Y) \geq k_Y$. Let $u \leq k_X$*

$$d(\langle X, Y\rangle | X : U) \leq 2(2^{u-k_X} + 2^{(n+1-u-k_Y)/2}),$$

where $d(X|Y) : \sum_y \Pr[Y = y]\Delta(X|Y = y; U)$ for U the uniform distribution (independent of X).

5.2.3 Computational Model for Streaming Adversaries

In this section we discuss the computational model used for analysis of the streaming adversaries. This model is similar to the one used in [110].

We first discuss streaming adversaries in general, and then discuss the specific case of streaming adversaries for learning parity and streaming tampering functions..

General Streaming Adversaries. The input is represented as a stream $S_1, \ldots, S_\ell$, where for $i \in [\ell]$, each $S_i \in \{0,1\}^B$, where B is the block length. We model the adversary by a *branching program*. A branching program of length ℓ and width w, is a directed acyclic graph with the vertices arranged in $\ell + 1$ layers such that no layer contains more than w vertices. Intuitively, each layer represents a time step of computation whereas, each vertex in the graph corresponds to the potential memory state learned by the adversary. The first layer (layer 0) contains a single vertex, called the *start vertex*, which represents the input. A vertex is called *leaf* if it has out-degree 0, and represents the output (the learned value of x) of the program. Every non-leaf vertex in the program has exactly 2^{n+1} outgoing edges, labeled by elements $S \in \{0,1\}^B$, with exactly one edge labeled by each such S, and all the edges from layer $j-1$ going to vertices in layer j. Intuitively, these edges represent the computation on reading S_i as streaming input. The stream $S_1, \ldots, S_\ell$, therefore, define a computation-path in the branching program.

We discuss the streaming branching program adversaries, and streaming adversaries for learning parity next.

Definition 5.2.3 (Streaming Branching Program Adversaries)**.** *A branching program of length m and width w is a directed acyclic graph with vertices arranged in $m+1$ layers containing at most w vertices each. In the first layer, that we call layer 0, there is only one vertex, called the start vertex. A vertex of out-degree 0 is called a leaf. All the vertices in the layer m are leaves. Every non-leaf vertex in the program has exactly 2^{n+1} outgoing edges, labeled by elements $S \in \{0,1\}^B$, with exactly one edge labeled by each such S, and all the edges from layer $j-1$ going to vertices in layer j.*

Computation Path: *The stream $S_1, \ldots, S_\ell \in \{0,1\}^B$ that are given as input, define a computation-path in the branching program, by starting form the start vertex and following at step i the edge labeled by S_i, until reaching a leaf.*

Streaming Adversaries for Learning Parity. Recall, that in the Parity Learning setting, the adversary aims to learn a uniform random string $x \in \{0,1\}^n$, from a stream of samples, $(a_1, b_1), (a_2, b_2), \dots, (a_m, b_m)$, where each a_i is uniformly distributed over $\{0, 1\}^n$ and for every i, $b_i = a_i \cdot x$.

Definition 5.2.4 (Streaming Branching Program for Parity Learning). *[110] A branching program of length m and width w, for parity learning is a directed acyclic graph with vertices arranged in $m + 1$ layers containing at most w vertices each. In the first layer, that we call layer 0, there is only one vertex, called the start vertex. A vertex of out-degree 0 is called a leaf. All the vertices in the layer m are leaves. Every non-leaf vertex in the program has exactly 2^{n+1} outgoing edges, labeled by elements $(a, b) \in \{0, 1\}^n \times \{0, 1\}$, with exactly one edge labeled by each such (a, b), and all the edges from layer $j - 1$ going to vertices in layer j.*

Computation Path: *The samples $(a_1, b_1), (a_2, b_2), \dots, (a_m, b_m) \in \{0, 1\}^n \times \{0, 1\}$ that are given as input, define a computation-path in the branching program, by starting form the start vertex and following at step i the edge labeled by (a_i, b_i), until reaching a leaf.*

Streaming Tampering Functions. The input is represented as a stream $S_1, \dots, S_\ell$, where for $i \in [\ell]$, each $S_i \in \{0, 1\}^B$, where B is the block length. We model the adversary by a *branching program*, which *reads in a block of length B* and *writes out a block of length B* in each step. A branching program of length ℓ and width w, is a directed acyclic graph with the vertices arranged in $\ell + 1$ layers such that no layer contains more than w vertices. Intuitively, each layer represents a time step of computation whereas, each vertex in the graph corresponds to the potential memory state learned by the adversary. The first layer (layer 0) contains a single vertex, called the *start vertex*, which represents the input. A vertex is called *leaf* if it has out-degree 0, and represents the output (the learned value of x) of the program. Every non-leaf vertex in the program has exactly 2^{n+1} outgoing edges, labeled by pairs of elements $S_{\mathsf{in}}, S_{\mathsf{out}} \in \{0,1\}^B$, with exactly one edge labeled by each such S_{in}, and all the edges from layer $j-1$ going to vertices in layer j. Intuitively, these edges represent the computation on reading S_i as streaming input, as well as the output in that time step. The stream $S_1, \ldots, S_\ell$, therefore, define a computation-path in the branching program.

Definition 5.2.5 (Streaming Tampering Functions)**.** *A branching program of length m and width w is a directed acyclic graph with vertices arranged in $m+1$ layers containing at most w vertices each. In the first layer, that we call layer 0, there is only one vertex, called the start vertex. A vertex of out-degree 0 is called a leaf. All the vertices in the layer m are leaves. Every non-leaf vertex in the program has exactly 2^{n+1} outgoing edges, labeled by pairs of elements $S_{\mathsf{in}}, S_{\mathsf{out}} \in \{0,1\}^B$, with exactly one edge labeled by each such S_{in}, and all the edges from layer $j-1$ going to vertices in layer j.*

Computation Path: *The stream $S_1, \ldots, S_\ell \in \{0,1\}^B$ that are given as input, define a computation-path in the branching program, by starting form the start vertex and following at step i the edge labeled by S_i, until reaching a leaf.*

In this work we consider the *Polynomial-time uniform* family of branching programs which can be informally defined as follows:

A family of branching programs of size s (number of nodes in the branching program), denoted by $\mathsf{BP} = \{\mathsf{BP}_s : s \in \mathbb{N}\}$ is *Polynomial-time uniform* if there exists a deterministic Turing machine M, such that

- M runs in polynomial time (i.e. $\mathrm{poly}(s)$), and
- For all $s \in \mathbb{N}$, M outputs the description (nodes and corresponding labels) of BP_s on input 1^s

5.3 Generic Construction for One-Bit Messages

In this section we present the generic construction for encoding a single bit messages.

Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a public key encryption scheme with perfect correctness (see Definition 3.4.3). Let $\Pi^{\mathsf{NI}} = (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ be a non-interactive simulatable proof system with soundness against adversaries $\mathcal{A} \in \mathcal{G}$ (see Definition 3.5.4). Note that in the CRS model, we implicitly assume that all algorithms take the CRS as input, and for simplicity of notation, sometimes do not list the CRS as an explicit input.
$\mathsf{CRSGen}(1^n)$:

1. Choose $(\textsc{pk}, \textsc{sk}) \leftarrow \mathsf{Gen}(1^n)$.
2. Choose $[(\mathsf{crs}_i^{\mathsf{NI}}, \tau_{\mathsf{sim}}^i)]_{i\in\{0,\ldots n\}} \leftarrow \mathsf{CRSGen}^{\mathsf{NI}}(1^n)$. Let $\overrightarrow{\mathsf{crs}}^{\mathsf{NI}} := [\mathsf{crs}_i^{\mathsf{NI}}]_{i\in\{0,\ldots n\}}$ and let $\overrightarrow{\tau}_{\mathsf{sim}} := [\tau_{\mathsf{sim}}^i]_{i\in\{0,\ldots n\}}$ and output $\mathsf{crs} := (\textsc{pk}, \overrightarrow{\mathsf{crs}}^{\mathsf{NI}})$.

Languages. We define the following languages:

- $\mathcal{L}_i^\beta$: For $i \in [n]$, $\beta \in \{0,1\}$, $s := (\hat{\vec{k}}, \vec{c}, c) \in \mathcal{L}_i^\beta$ iff the i-th ciphertext $c_i := k_i \oplus \beta$ (where $\vec{c} = c_1, \ldots, c_n$) and the i-th encryption $\hat{k}_i$ (where $\hat{\vec{k}} = \hat{k}_1, \ldots, \hat{k}_{n+1}$) is an encryption of k_i under $\textsc{pk}$ (where $\textsc{pk}$ is hardwired into the language).
- $\mathcal{L}$: $s := (\hat{\vec{k}}, \vec{c}, c) \in \mathcal{L}$ iff $(x_1, \ldots, x_n)$ is in the support of D_b where:
  1. For $i \in [n]$, $x_i := c_i \oplus k_i$, and $b := c \oplus k_{n+1}$
  2. $\hat{\vec{k}}$ is an encryption of $k_1, \ldots, k_{n+1}$ under $\textsc{pk}$ (where $\textsc{pk}$ is hardwired into the language).

$\mathsf{E}(\mathsf{crs}, b)$:

1. Sample $\vec{x} \leftarrow D_b$, where $\vec{x} = x_1, \ldots, x_n$.
2. Choose an $n+1$-bit key $\vec{k} = k_1, \ldots, k_n, k$ uniformly at random. For $i \in [n]$, compute $\hat{k}_i \leftarrow \mathsf{Encrypt}(\textsc{pk}, k_i)$ and compute $\hat{k}_{n+1} \leftarrow \mathsf{Encrypt}(\textsc{pk}, k)$. Let $\hat{\vec{k}} := \hat{k}_1, \ldots, \hat{k}_{n+1}$.
3. Compute $c_1 := k_1 \oplus x_1, \ldots, c_n := k_n \oplus x_n$. Let $\vec{c} := c_1, \ldots, c_n$. Also, compute $c := b \oplus k$.
4. For $i \in [n]$, compute a non-interactive, simulatable proof T_i proving $s := (\hat{\vec{k}}, \vec{c}, c) \in \mathcal{L}_i^{x_i}$ relative to $\mathsf{crs}_i^{\mathsf{NI}}$.
5. Compute a non-interactive, simulatable proof T proving $s := (\hat{\vec{k}}, \vec{c}, c) \in \mathcal{L}$ relative to $\mathsf{crs}_0^{\mathsf{NI}}$.
6. Output $\mathsf{CW} := (\hat{\vec{k}}, c_1, \ldots, c_n, c, T, x_1, T_1, .., x_n, T_n)$.

$\mathsf{D}(\mathsf{crs}, \mathsf{CW})$:

1. Parse $\mathsf{CW} := (\hat{\vec{k}}, c_1, \ldots, c_n, c, T, x_1, T_1, .., x_n, T_n)$
2. Check that V^{NI} outputs 1 on all proofs $T_1, .., T_n, T$, relative to the corresponding CRS. If yes, output b such that $x_1 \ldots x_n$ is in the support of D_b. If not, output 0.

Figure 5.1: Non-malleable code $(\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$, secure against $\mathcal{F}$ tampering.

$\mathsf{E}_1(\mathsf{crs}, \vec{\tau}_{\mathsf{sim}}, r, b)$:

1. Sample $\vec{x} \leftarrow D_b$, where $\vec{x} = x_1, \ldots, x_n$.
2. Choose an $n+1$-bit key $\vec{k} = k_1, \ldots, k_n, k$ uniformly at random. For $i \in [n]$, compute $\hat{k}_i \leftarrow \mathsf{Encrypt}(\textsc{pk}, k_i)$ and compute $\hat{k}_{n+1} \leftarrow \mathsf{Encrypt}(\textsc{pk}, k)$. Let $\hat{\vec{k}} := \hat{k}_1, \ldots, \hat{k}_{n+1}$.
3. Compute $c_1 := k_1 \oplus x_1, \ldots, c_n := k_n \oplus x_n$. Let $\vec{c} := c_1, \ldots, c_n$.
4. Compute $c := b \oplus k$.
5. For $i \in [n]$, use τ^i_{sim} and r to simulate a non-interactive proof T'_i proving $(\hat{\vec{k}}, \vec{c}, c) \in \mathcal{L}^{x_i}_i$, relative to $\mathsf{crs}^{\mathsf{NI}}_i$.
6. Use τ^0_{sim} and r to simulate a non-interactive proof T' proving $(\hat{\vec{k}}, \vec{c}, c) \in \mathcal{L}$, relative to $\mathsf{crs}^{\mathsf{NI}}_0$.
7. Output $\mathsf{CW} := (\hat{\vec{k}}, c_1, \ldots, c_n, c, T', x_1, T'_1, .., x_n, T'_n)$.

Figure 5.2: Encoding algorithm with simulated proofs.

$\mathsf{E}_2(\mathsf{crs}, \vec{\tau}_{\mathsf{sim}}, r, b)$:

1. Sample $\vec{x} \leftarrow D_b$, where $\vec{x} = x_1, \ldots, x_n$.
2. Choose $c'_1, \ldots, c'_n$ uniformly at random. Let $\vec{c'} := c'_1, \ldots, c'_n$.
3. Choose c' uniformly at random.
4. Set $\vec{k'} = c'_1, \ldots, c'_n, c'$. For $i \in [n]$, compute $\hat{k}'_i \leftarrow \mathsf{Encrypt}(\textsc{pk}, k'_i)$ and compute $\hat{k}'_{n+1} \leftarrow \mathsf{Encrypt}(\textsc{pk}, k')$.
   Let $\hat{\vec{k'}} := \hat{k}'_1, \ldots, \hat{k}'_{n+1}$.
5. For $i \in [n]$, use τ^i_{sim} and r to simulate a non-interactive proof T'_i proving $(\hat{\vec{k'}}, \vec{c'}, c) \in \mathcal{L}^{x_i}_i$, relative to $\mathsf{crs}^{\mathsf{NI}}_i$.
6. Use τ^0_{sim} and r to simulate a non-interactive proof T' proving $(\hat{\vec{k'}}, \vec{c'}, c) \in \mathcal{L}$, relative to $\mathsf{crs}^{\mathsf{NI}}_0$.
7. Output $\mathsf{CW} := (\hat{\vec{k'}}, c'_1, \ldots, c'_n, c', T', x_1, T'_1, .., x_n, T'_n)$.

Figure 5.3: Encoding algorithm with simulated proofs and encryptions.

```
$\mathsf{Ext}(\mathsf{crs}, \textsc{sk}, \mathsf{CW})$:

  1. Parse $\mathsf{CW} := (\hat{\vec{k}}, c_1, \ldots, c_n, c, T, x_1, T_1, .., x_n, T_n)$,

  2. Output $\mathsf{Decrypt}(\textsc{sk}, \hat{k}_{n+1})$.
```

Figure 5.4: Extracting procedure Ext.

```
$\mathsf{D}'(\mathsf{crs}, k, \mathsf{CW})$:

  1. Parse $\mathsf{CW} := (\hat{\vec{k}}, c_1, \ldots, c_n, c, T, x_1, T_1, .., x_n, T_n)$,

  2. Check that $\mathsf{V}^{\mathsf{NI}}$ outputs 1 on all proofs $T_1, .., T_n, T$, relative to the corresponding
     CRS,

  3. If not, output 0. Otherwise, output $b := k \oplus c$.
```

Figure 5.5: Alternate decoding procedure D', given additional extracted key k as input.

```
$g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r)$:

  1. Parse $\mathsf{CW} = (\hat{\vec{k}}, \vec{c}, c, T, x_1, T_1, .., x_n, T_n)$, $\mathsf{CW}^* = (\hat{\vec{k}}^*, \vec{c}^*, c^*, T^*, x_1^*, T_1^*, .., x_n^*, T_n^*)$.

  2. If (1) $\mathsf{V}^{\mathsf{NI}}$ outputs 1 on all proofs $T^*, T_1^*, .., T_n^*$, relative to the corresponding
     CRS; and (2) $(\hat{\vec{k}}, \vec{c}, c) = (\hat{\vec{k}}^*, \vec{c}^*, c^*)$, then output 1. Otherwise output 0.
```

Figure 5.6: The predicate $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r)$.

Let $\Psi(p, c, x, y, r, z)$ be defined as a function that takes as input a predicate p, and variables c, x, y, r, z. If $p(c, x, y, r) = 1$, then Ψ outputs 0. Otherwise, Ψ outputs z.

Theorem 5.3.1. *Let* (E, D), E_1, E_2, Ext, D' *and* g *be as defined in Figures 5.1, 5.2, 5.3, 5.4, 5.5 and 5.6. Let* $\mathcal{F}$ *be a computational class. If, for every adversary* $\mathcal{A} \in \mathcal{G}$ *outputting tampering functions* $f \in \mathcal{F}$*, all of the following hold:*

Simulation of proofs.

1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_0, f(\mathsf{CW}_0), r_0) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1) = 1]$,

2. $\Psi(g, \mathsf{crs}, \mathsf{CW}_0, f(\mathsf{CW}_0), r_0, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_0); r_0)) \approx$

   $\Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1))$,

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_0, r_1 *are sampled uniformly at random,*

$\mathsf{CW}_0 \leftarrow \mathsf{E}(\mathsf{crs}, 0)$ *and* $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, 0)$.

Simulation of Encryptions.

1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1]$,

2. $\Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1)) \approx$

   $\Psi(g, \mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2))$, *where* $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$*,* $f \leftarrow \mathcal{A}(\mathsf{crs})$*,* r_1, r_2 *are sampled uniformly at random,*

$\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, 0)$ *and* $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, 0)$*.*

Simulation Soundness.

$$\Pr\left[\begin{array}{c}\mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2) \neq \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \\ \wedge g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 0\end{array}\right] \leq \mathsf{negl}(n),$$

where $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$*,* $f \leftarrow \mathcal{A}(\mathsf{crs})$*,* r_2 *is sampled uniformly at random and*

$\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r, 0)$*.*

Hardness of D_b relative to Alternate Decoding.

1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_3, f(\mathsf{CW}_3), r_3) = 1]$,

2. $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \approx \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$,

where $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$*,* $f \leftarrow \mathcal{A}(\mathsf{crs})$*,* r_2, r_3 *are sampled uniformly at random,*

$\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, 0)$ *and* $\mathsf{CW}_3 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_3, 1)$*.*

Then the construction presented in Figure 5.1 is a non-malleable code for class $\mathcal{F}$ *against adversaries* $\mathcal{A} \in \mathcal{G}$*.*

5.3.1 Proof of Theorem 5.3.1

In this subsection we prove Theorem 5.3.1.

We take g to be the predicate that is used in the $\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,m,g}(n)$ tampering experiment. We must argue that for every $m \in \{0,1\}$ and every attacker $A \in \mathcal{G}$ the output of the experiment $\mathsf{Expt}^{\Pi,\mathcal{F}}_{A,m,g}(n)$ is 1 with at most negligible probability.

Assume towards contradiction that for some $A \in \mathcal{G}$ the output of the experiment is 1 with non-negligible probability. Then this means that the probability in the last line of experiment $\mathsf{Expt}^{\Pi,\mathcal{F}}_{A,m,g}(n)$ that $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r) = 1 \wedge \mathsf{D}(\mathsf{crs}, \mathsf{CW}^*; r) \neq m$ is non-negligible. Parse $\mathsf{CW} = (\hat{\vec{k}}, \vec{c}, c, T, x_1, T_1, .., x_n, T_n)$, and

$\mathsf{CW}^* = (\hat{\vec{k}}^*, \vec{c}^*, c^*, T^*, x_1^*, T_1^*, .., x_n^*, T_n^*)$.

Recall that $\mathsf{D}(\mathsf{crs}, \mathsf{CW}; r) = m$. Thus, if the above event occurs, it means that $\mathsf{D}(\mathsf{crs}, \mathsf{CW}; r) \neq \mathsf{D}(\mathsf{crs}, \mathsf{CW}^*; r)$. But since $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r) = 1$, it means that V^{NI} outputs 1 on all proofs $T^*, [T_i^*]_{i \in [n]}$ and $(\hat{\vec{k}}, \vec{c}, c) = (\hat{\vec{k}}^*, \vec{c}^*, c^*)$.
This, in turn, means that there must be some bit x_i, x_i^* that CW and CW^* differ on. But note that by assumption $c_i = c_i^*$. Due to the fact that CW is well-formed and perfect correctness of the encryption scheme, it must mean that $c_i^* \notin \mathcal{L}_i^{x_i^*}$. But recall that by assumption, proof T_i^* verifies correctly. This means that soundness is broken by $A \in \mathcal{G}$. This contradicts the security of the proof system Π^{NI}.

Next, recall that we wish to show that for any adversary $A \in \mathcal{G}$ outputting tampering function

$\{\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,0,g}\}_{k \in \mathbb{N}} \approx \{\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,1,g}\}_{k \in \mathbb{N}}$

To do so we consider the following hybrid argument:

Hybrid 0: The real game, $\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,0,g}$, relative to g, where the real encoding $\mathsf{CW}_0 \leftarrow \mathsf{E}(\mathsf{crs}, 0)$ and the real decoding oracle D are used.

Hybrid 1: Replace the encoding from the previous game with $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, 0)$ where r_1 is chosen uniformly at random and g, D use random coins r_1.

Hybrid 2: Replace the encoding from the previous game with $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, 0)$, where r_2 is chosen uniformly at random and g, D use random coins r_2.

Hybrid 3: Replace the decoding from the previous game, with $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$. where r_2 is chosen uniformly at random and g, E_2 use random coins r_2.

Hybrid 4: Same as Hybrid 3, but replace the encoding with $\mathsf{CW}_3 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_3, 1)$, where r_3 is chosen uniformly at random and g, D' use random coins r_3.

Now, we prove our hybrids are indistinguishable.

Claim 5.3.1. *Hybrid 0 is computationally indistinguishable from Hybrid 1.*

Proof. The claim follows immediately from the **Simulation of proofs** property in Theorem 5.3.1. □

Claim 5.3.2. *Hybrid 1 is computationally indistinguishable from Hybrid 2.*

Proof. The claim follows immediately from the **Simulation of Encryptions** property in Theorem 5.3.1. □

Claim 5.3.3. *Hybrid 2 is computationally indistinguishable from Hybrid 3.*

Proof. This claim follows from the fact that (1) if $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r) = 1$, then the experiment outputs same^* in both Hybrid 2 and Hybrid 3; and (2) the probability that $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r) = 0$ and the output of the experiment is different in Hybrid 2 and Hybrid 3 is at most negligible, due to the **Simulation Soundness** property in Theorem 5.3.1. □

Claim 5.3.4. *Hybrid 3 is computationally indistinguishable from Hybrid 4.*

Proof. This follows from the fact that (1) for $\gamma \in \{2, 3\}$ if $g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1$ then

$\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_\gamma)), f(\mathsf{CW}_\gamma); r_\gamma)$ always outputs 0 and so

$$\begin{aligned}
&\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_\gamma)), f(\mathsf{CW}_\gamma); r_\gamma) \\
&\qquad \equiv \Psi(g, \mathsf{crs}, \mathsf{CW}_\gamma, f(\mathsf{CW}_\gamma), r_\gamma, \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_\gamma)), f(\mathsf{CW}_\gamma); r_\gamma));
\end{aligned}$$

and (2) the **Hardness of D_b relative to Alternate Decoding** property in Theorem 5.3.1. □

5.3.2 One-Bit NMC for AC^0 and beyond

In this section, we show that our generic construction yields efficient NMC for AC^0 in the CRS model, when each of the underlying primitives is appropriately instantiated.

Theorem 5.3.2. $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *(presented in Figure 5.1) is a one-bit, computational, non-malleable code in the CRS model, secure against tampering by* AC^0 *circuits, if the underlying components are instantiated in the following way:*

- $\mathcal{E} := (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is a public key encryption scheme with perfect correctness and decryption in* AC^0 *.*
- $\Pi^{\mathsf{NI}} := (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ *is a same-string, weak one-time simulation-sound NIZK with verifier in* AC^0 *.*
- *For* $b \in \{0, 1\}$*,* D_b *is the distribution that samples bits* $x_1 \ldots x_n$ *uniformly at random, conditioned on* $x_1 \oplus \cdots \oplus x_n = b$*.*

Note that given Theorem 5.2.2, proof systems Π^{NI} as above exist, under the assumption that same-string, weak one-time simulation-sound NIZK with (arbitrary polynomial-time) deterministic verifier exists. Such NIZK can be constructed in the CRS model from enhanced trapdoor permutations [111]. Public key encryption with perfect correctness and decryption in AC^0 can be constructed by applying the low-decryption-error transformation of Dwork et al. [63] to the (reduced decryption error) encryption scheme of Bogdanov and Lee [26]. We now provide an instantiation of the public key encryption scheme.

Public key encryption in AC^0 . We now present the result presented by Bogdanov and Lee in [26] which showed that the encryption scheme given by Applebaum et al. in [13] can be implemented by circuit with constant depth and size polynomial in the security parameter.

PKE Scheme based on Bipartite Graphs [13]

- $\mathsf{Gen}(1^n)$: The key generation algorithm takes security parameter n as input and outputs a random bipartite graph $G = ((U_1, U_2), E)$ as the public key $\textsc{pk}$, where $|U_1| = n$ and $|U_2| = r = n^{0.9}$ generated in the following way. First choose the random subsets $S_1 \subseteq U_1$ and $S_2 \subseteq U_2$ of sizes s and $s/3$ respectively for $s = O(\log n)$. Each vertex in S_1 is connected to d (possibly repeated) random vertices in S_2 and each vertex outside S_1 is connected to d random vertices in U_2. The secret key $\textsc{sk}$ is an odd size subset of S_1 such that each vertex in S_2 has an even number of neighbors in $\textsc{sk}$.

- $\mathsf{Encrypt}(\textsc{pk}, b)$: To encrypt bit $b \in \{0,1\}$, choose a random subset ${S'}_2 \subset U_2$ and output $\vec{c} = \vec{y} + \vec{e} + b \cdot \vec{1}$, where each coordinate of $\vec{y} \in \{0,1\}^n$ is the degree of corresponding vertex in S_1 restricted to ${S'}_2 \bmod 2$, $\vec{e} \in \{0,1\}^n$ is a vector with each coordinate $(e_i : i \in [n])$ sampled from distribution $\hat{\eta}$ with $\Pr[e_i = 0] = \eta$ independently, and $\vec{1} \in \{0,1\}^n$ is the vector of all 1s.

- $\mathsf{Decrypt}(\textsc{sk}, \vec{c})$: Output $b = \sum_{i \in \textsc{sk}} c_i \bmod 2$.

Refer [13] for the security of the scheme presented above. We next present the AC^0 implementation of the PKE presented above as shown in [26].

AC^0 Implementation of [13] PKE Scheme based on Bipartite Graphs [26]

- Gen:

  1. Sample $y_1, y_2, \ldots, y_s$ from $[n]$ and $w_1, w_2, \ldots, w_{s/3}$ from $[r]$ to represent the subsets $S_1 \subseteq U_1$ and $S_2 \subseteq U_2$ respectively.
  2. Sample $v_{i,1}, v_{i,2}, \ldots, v_{i,d}$ from $[r]$ for all $i \in [n]$. These represent the random neighbors of each vertex in $U_1 \setminus S_1$.
  3. Sample $\hat{v}_{i,1}, \hat{v}_{i,2}, \ldots \hat{v}_{i,d}$ from $[s/3]$ for all $i \in [s]$. These become the random neighbors of the vertices in S_1 after being mapped to the w_i's by the index function $\iota : [s/3] \to [r]$ such that $\iota(i) = w_i$. This is written as:

$$\iota(i) = \bigvee_{j=1}^{s/3} [(i = j) \wedge w_j]$$

.

The key generation circuit outputs $v_{i,1}, v_{i,2}, \ldots, v_{i,d}$ if the vertex i is not in S_1 and outputs $\iota(\hat{v}_{i,1}), \iota(\hat{v}_{i,2}), \ldots \iota(\hat{v}_{i,d})$ otherwise. Now we can output the j^{th} random neighbor of each vertex $i \in U_1$ as

$$\left[\delta_i \wedge \bigvee_{k=1}^{s} [(i = k) \wedge \iota(\hat{v}_{k,j})]\right] \vee (\bar{\delta}_i \wedge v_{i,j}),$$

where $\delta_i := \bigvee_{k=1}^{s} (i = y_k)$ indicates whether i belongs to S_1.

To come up with secret key SK, we enumerate all the possible subsets of S_1 (this is still efficient since $s = O(\log n)$) and output the first one that satisfies the linear dependency. Given an odd size subset of S_1 indicated by the support of the vector $\vec{a} \in \{0,1\}^s$, note that the formula

$$f_{\vec{a}} = \bigvee_{j=1}^{s/3} \bigoplus_{i:a_i=1} \bigoplus_{k=1}^{d} (\hat{v}_{i,k} = j)$$

outputs 0 only if every vertex in S_2 has an even number of neighbors in support of $\vec{a}$ and outputs 1 otherwise. (Since the XOR involves only $O(d \log n)$ inputs it can be calculated with a circuit of depth 2 and size $n^{O(d)}$.) We can therefore, enumerate all the possible $\vec{a} \in \{0,1\}^s$ with odd hamming weight and output the first $\vec{a}$ such that $f_{\vec{a}} = 0$. The secret key is represented by a vector $\vec{z}$ containing s entries in $[n]$, where each non-zero entry corresponds to a vertex in SK. More precisely, we output i^{th} entry as

$$z_i = \iota \left(\bigvee_{\vec{a} \in \{0,1\}^s :\, wt(\vec{a}) \text{ is odd}} \left[\bar{f_{\vec{a}}} \wedge \left(\bigwedge_{\vec{a'} < \vec{a}} f_{\vec{a'}} \right) \wedge (a_i \wedge i) \right] \right).$$

- Encrypt: Given a public key represented by the neighbors $v_{i,1}, v_{i,2}, \ldots, v_{i,d}$ of each vertex $i \in U_1$. To encrypt bit $b \in \{0,1\}$, choose a random vector $\vec{x} \in \{0,1\}^r$ whose support forms the subset S'_2 of U_2, a noise vector $\vec{e} \in \{0,1\}^n$ by choosing each of its entries independently from $\hat{\eta}$. The i^{th} bit of ciphertext can then be written as

$$\bigvee_{\substack{k_i \neq k_j; \\ 1 \leq i \leq j \leq d; \\ k_i \in [r]; \\ a_1, \ldots, a_d : a_1 + \cdots + a_d = 1 \bmod 2}} \left[\bigwedge_{j=1}^{d} (v_{i,j} = k_j) \wedge (x_{k_1} = a_1) \wedge \cdots \wedge (x_{k_d} = a_d) \right] \oplus e_i \oplus b$$

- $\mathsf{Decrypt}$: Given the ciphertext $\vec{c}$ and secret key SK represented by the vector $\vec{z} \in \{0,1\}^{s \times \log n}$, output

$$\bigoplus_{i=1}^{s} \bigvee_{k=1}^{n} [(z_i = k) \wedge c_k].$$

Reducing the decryption error The [13] encryption scheme suffers from significant encryption error (and thus decryption error) however, this can be minimized arbitrarily by encrypting the message multiple times independently. The decryption algorithm can then take approximate majority to compute the encrypted bit. Approximate majority can be computed with constant depth circuits [11] (depth 3) and thus the overall decryption algorithm is still in AC^0 .

We now use the following transformation given by [63] to obtain almost-all keys perfect decryption for the above encryption scheme.

Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be any public-key encryption scheme. Also let $\ell(n) > n$ be the number of bits used by $\mathsf{Encrypt}$ to encrypt n-bit messages. Let prg be a pseudorandom generator that expands n bits to $\ell(n)$ bits. Then the modified encryption scheme $\mathcal{E}' = (\mathsf{Gen}', \mathsf{Encrypt}', \mathsf{Decrypt}')$ is obtained as follows: On input 1^n, Gen' outputs $((\textsc{pk}, \bar{r}), \textsc{sk})$ where $(\textsc{pk}, \textsc{sk}) \leftarrow \mathsf{Gen}(1^n)$ and $\bar{r} \in \{0,1\}^{\ell(n)}$ is chosen uniform randomly. To encrypt message m, $\mathsf{Encrypt}'$ samples a random n-bit string r and outputs $\mathsf{Encrypt}(\textsc{pk}, m)$ using $\mathsf{prg}(r) \oplus \bar{r}$ as randomness for $\mathsf{Encrypt}$. $\mathsf{Decrypt}'$ is same as $\mathsf{Decrypt}$, note that this preserves the computational complexity of decryption.

Theorem 5.3.3. *[63] Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be any $(1 - 2^{-4n})$ correct public key encryption scheme with $\mathsf{Decrypt}$ being deterministic. Then $\mathcal{E}' =$*

$(\mathsf{Gen}', \mathsf{Encrypt}', \mathsf{Decrypt})$ *is an almost-all-key perfectly correct public encryption scheme. Furthermore, if $\mathcal{E}$ is IND-CPA secure then so is $\mathcal{E}'$.*

Note that the above transformation takes a public key encryption scheme $\mathcal{E}$ with sufficiently low decryption error and transforms it into a public key encryption scheme that enjoys perfect correctness, and furthermore, note that the decryption algorithm $\mathsf{Decrypt}$ remains unchanged. Therefore, if we start with the (reduced decryption error) version of the AC^0 Bogdanov and Lee public key encryption scheme, we obtain a perfectly correct public key encryption scheme with decryption in AC^0 , as desired.

Proof of theorem 5.3.2. To prove the theorem, we need to show that for every PPT adversary $\mathcal{A}$ outputting tampering functions $f \in \mathcal{F}$, the necessary properties from Theorem 5.3.1 hold. We next go through these one by one.

- **Simulation of proofs.**

    1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_0, f(\mathsf{CW}_0), r_0) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1) = 1]$,

    2. $\Psi(g, \mathsf{crs}, \mathsf{CW}_0, f(\mathsf{CW}_0), r_0, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_0); r_0)) \approx$

       $\Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1))$,

  where $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_0, r_1 are sampled uniformly at random,

  $\mathsf{CW}_0 \leftarrow \mathsf{E}(\mathsf{crs}, 0)$ and $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, 0)$.

This follows immediately from the zero-knowledge property of

$\Pi^{\mathsf{NI}} = (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$.

- **Simulation of Encryptions.**

  1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1]$,

  2. $\Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1)) \approx$

     $\Psi(g, \mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2))$,

  where $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_1, r_2 are sampled uniformly at random,

  $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, 0)$ and $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, 0)$.

  This follows immediately from the fact that $\vec{c}, c$ and $\vec{c}', c'$ are identically distributed when generated by E_1 versus E_2 and from the semantic security of the public key encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$.

- **Simulation Soundness.**

$$\Pr\begin{bmatrix} \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2) \neq \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \\ \wedge g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 0 \end{bmatrix} \leq \mathsf{negl}(n),$$

  where $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_2 is sampled uniformly at random and

$\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r, 0)$.

Note that $g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 0$ only if either of the following is true: (1) V^{NI} did not output 1 on all tampered proofs $T^*, T_1^*, \ldots, T_n^*$ in $f(\mathsf{CW}_2)$; or (2) the first 3 elements of CW_2 and $f(\mathsf{CW}_2)$ are not identical (i.e., $(\hat{\vec{k}}, \vec{c}, c) \neq (\hat{\vec{k^*}}, \vec{c^*}, c^*)$). Now in case (1), both $\mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2)$, and $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ output 0. This is contradiction to the claim that $\mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2) \neq \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$. In case (2), the extractor $\mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2))$ outputs $k_{n+1}^* := \mathsf{Decrypt}(\mathrm{SK}, \hat{k^*}_{n+1})$ and $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ outputs $b^* = c^* \oplus k_{n+1}^*$. Now, if $\mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2) \neq \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ but V^{NI} outputs 1 on all tampered proofs $T^*, T_1^*, \ldots, T_n^*$ in $f(\mathsf{CW}_2)$ then one-time simulation soundness of $\Pi^{\mathsf{NI}} = (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ does not hold.

- **Hardness of D_b relative to Alternate Decoding.**

  1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_3, f(\mathsf{CW}_3), r_3) = 1]$,

  2. $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \approx$

     $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$,

  where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_2, r_3 are sampled uniformly at random,

  $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, 0)$ and $\mathsf{CW}_3 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_3, 1)$.

Let $\vec{X}$ denote a random variable where $\vec{X}$ is sampled from D_0 with probability 1/2 and $\vec{X}$ is sampled from D_1 with probability 1/2 and let random variable CW denote the output of E_2 when $\vec{X}$ replaces $\vec{x}$.

To show (1), assume $\Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1]$ and $\Pr[g(\mathsf{crs}, \mathsf{CW}_3, f(\mathsf{CW}_3), r_3) = 1]$ differ by a non-negligible amount. This implies that takes as input $\vec{X}$, hardwires all other random variables, and outputs 1 in the case that $g(\mathsf{crs}, \mathsf{CW}, f(\mathsf{CW}), r) = 1$ and 0 otherwise, implying that it has non-negligible correlation to the parity of its input $\vec{X}$. We will show that the above can be computed by an AC^0 circuit with input $\vec{X}$, thus contradicting Theorem 3.3.1, which says that an AC^0 circuit has at most negligible correlation with parity of its input $\vec{X}$, denoted $\mathcal{P}(\vec{X})$. Details follow.

We construct the distribution of circuits $\mathcal{C}^1_{\mathcal{F}}$. A draw $C \sim \mathcal{C}^1_{\mathcal{F}}$ is done as follows:

1. Sample $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$.
2. Sample tampering function $\mathcal{A}(\mathsf{crs}) \rightarrow f$.
3. Sample $\vec{c}', c'$ uniformly at random.
4. Set $\vec{k}' = c'_1, \ldots, c'_n, c$. For $i \in [n]$, compute $\hat{k}'_i \leftarrow \mathsf{Encrypt}(\textsc{pk}, k'_i)$ and compute $\hat{k}'_{n+1} \leftarrow \mathsf{Encrypt}(\textsc{pk}, k')$.
5. Sample r uniformly at random.
6. Sample simulated proofs $[T_i^{'\beta}]_{\beta \in \{0,1\}, i \in [n]}$ and T' (as described in Figure 5.3).
7. Output the following circuit C that has the following structure:

- **hardwired variables:** crs, $\textsc{sk}$, f, $\hat{\vec{k}}'$, $\vec{c}'$, c', r, $[T_i'^{\beta}]_{\beta\in\{0,1\}, i\in[n]}$.
- **input:** $\vec{X}$.
- **computes and outputs:**

$$g(\mathsf{crs}, \mathsf{CW}, f(\mathsf{CW}), r).$$

Note that given all the hardwired variables, computing CW is in AC^0 since all it does is, for $i \in [n]$, select the correct simulated proof $T_i'^{x_i}$ based on the corresponding input bit x_i. Additionally, f in AC^0 and g in AC^0 , since bit-wise comparison is in AC^0 and V^{SAT} is in AC^0 . Thus, the entire circuit is in AC^0 .

To show (2), assume $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ and $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$ have non-negligible statistical distance. This implies that a circuit that takes as input $\vec{X}$, hardwires all other random variables, and outputs $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW})), f(\mathsf{CW}); r_2)$ has non-negligible correlation to the parity of $\vec{X}$. We will show that $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW})), f(\mathsf{CW}); r_2)$ can be computed by an AC^0 circuit with input $\vec{X}$, thus contradicting Theorem 3.3.1, which says that an AC^0 circuit has at most negligible correlation with the parity of its input $\vec{X}$, denoted $\mathcal{P}(\vec{X})$. Details follow.

We construct the distribution of circuits $\mathcal{C}^2_{\mathcal{F}}$. A draw $C \sim \mathcal{C}^2_{\mathcal{F}}$ is done as follows:

1. Sample $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$.

2. Sample tampering function $\mathcal{A}(\mathsf{crs}) \to f$.

3. Sample $\vec{c}', c'$ uniformly at random.

4. Set $\vec{k}' = c'_1, \ldots, c'_n, c$. For $i \in [n]$, compute $\hat{k}'_i \leftarrow \mathsf{Encrypt}(\textsc{pk}, k'_i)$ and compute $\hat{k}'_{n+1} \leftarrow \mathsf{Encrypt}(\textsc{pk}, k')$.

5. Sample r uniformly at random.

6. Sample simulated proofs $[T_i'^{\beta}]_{\beta\in\{0,1\}, i\in[n]}$ and T' (as described in Figure 5.3).

7. Output the following circuit C that has the following structure:

   – **hardwired variables:** crs, $\textsc{sk}$, f, $\hat{\vec{k}}'$, $\vec{c}'$, c', r, $[T_i'^{\beta}]_{\beta\in\{0,1\}, i\in[n]}$.

   – **input:** $\vec{X}$.

   – **computes and outputs:**

$$\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW})), f(\mathsf{CW}); r_2).$$

Note that $\mathsf{Ext} \in \mathsf{AC}^0$ since decryption for $\mathcal{E} := (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ in AC^0 . Moreover, as above, given all the hardwired variables, computing CW is in AC^0 since all it does is, for $i \in [n]$, select the correct simulated proof $T_i'^{x_i}$ based on the corresponding input bit x_i. Additionally, f in AC^0 and D' is in AC^0 , since xor of two bits is in AC^0 and V^{SAT} is in AC^0 . Thus, the entire circuit is in AC^0 .

□

We present the analysis for more tampering classes next.

5.3.3 Tampering classes beyond AC^0 .

Let $\mathcal{F} \subsetneq P$ be a tampering class. Relative to this class $\mathcal{F}$, define the circuit classes $\mathcal{C}^1_{\mathcal{F}}$ and $\mathcal{C}^2_{\mathcal{F}}$ as in the proof above.

Theorem 5.3.4. *Let $\{\mathcal{D}_0, \mathcal{D}_1\}$ be (probabilistic polynomial time) samplable distributions with disjoint support. If the following hold:*

- *There exists a ppt distinguishing algorithm D such that for $b \in \{0,1\}$,*

$$\Pr_{\vec{x} \sim \mathcal{D}_b}[D(\vec{x}) = b] = 1.$$

- *For all $C \in \mathcal{C}^1_{\mathcal{F}} \cup \mathcal{C}^2_{\mathcal{F}}$*

$$\left| \Pr_{\vec{x} \sim \mathcal{D}_0}[C(\vec{x}) = 1] - \Pr_{\vec{x} \sim \mathcal{D}_1}[C(\vec{x}) = 1] \right| \leq \mathsf{negl}(n).$$

Then, under the same assumptions as Theorem 5.3.2, $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ is a computational non-malleable code against tampering by $\mathcal{F}$ that encodes a single bit.

We informally argue that Theorem 8 yields non-malleable codes against new classes: small threshold circuits and time-bounded probabilistic RAM machines. As noted earlier, non-malleable codes (in the CRS model without computational assumptions) from [73] are resilient against these classes. We provide theorems simply to demonstrate the applicability of our framework to a broad class of correlation bounds.

Theorem 5.3.5 ([41]). *For all d there exists $\epsilon_d > 0$ such that the following holds. There exists a probabilistic polynomial time computable f (the Generalized Andreev Function) such that for any depth-d threshold circuit with $n^{1+\epsilon_d}$ wires, C, f has correlation at most $2^{-n^{\Omega(1)}}$ with C.*

Corollary 5.3.1. *Let f be as in Theorem 5.3.5. Fix x_0, x_1 such that $f(x_b) = b$. Let D_b define a variable, X, which is defined by rejection sampling the uniform distribution over $\{0,1\}^n$ conditioned on $f(X) = b$; if after $O(n)$ tries the rejection sampling has not succeeded, output x_b.*

Then, assuming PKE in AC^0 and same-string weak one-time simulation-simulation sound NIZK, there exists a constant d_0 such that for $d > d_0$, $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ is a computational non-malleable code against depth-d threshold circuits with $n^{1+\epsilon'_d}$ wires, where ϵ'_d is any positive constant less than ϵ_d from Theorem 5.3.5.

The corollary follows from the fact that given the appropriate choice of security parameters for the encryption scheme and NIZK, any $C \in \mathcal{C}^1_{\mathcal{F}} \cup \mathcal{C}^2_{\mathcal{F}}$ has a representation as depth-d threshold circuit with $1 + \epsilon_d$ wires, so long as d is large enough, and the fact that rejection sampling fails with very low probability as f is balanced.

Using a generalization of a Theorem from [20] (combined with a result on prime finding from [104]):

Theorem 5.3.6 ([20]). *Let k be an integer (constant). Assuming one of the following:*

1. *a randomized variant of the Strong Exponential Time Hypothesis (BPSETH): $\forall \epsilon > 0, \exists q$ such that no randomized algorithm running in time $O(2^{1-\epsilon)n})$ is correct with probability $> 2/3$ on* every *instance of qSAT.*

2. *the randomized k-Orthogonal Vector Conjecture (BPkOVC): the k-Orthogonal Vector problem requires time $\Omega(n^{k-o(1)})$ for randomized algorithms that are correct with probability $> 2/3$ on* every *instance.*

   k-Othogonal Vector is a generalization of the well-studied Orthogonal Vector problem that asks given k sets of vectors $U_1, \ldots, U_k \subset \{0,1\}^{\log^2 n}$ each of size n, does there exist
   $u^{(1)} \in U_1, \ldots, u^{(k)} \in U_k$ such that $\sum_{i\in[log^2 n]} u_i^{(1)} \cdots u_i^{(k)} = 0$?

Then, there exists a function $\mathcal{FOV}_k$ such that any randomized time $t = \Omega(2^{\log^\epsilon n})$ (for $\epsilon > 0$) algorithm whose output (on a random instance x) is correct (the algorithm outputs $\mathcal{FOV}_k(x)$with probability at least δ must obey the following bound:

$$\frac{t}{\delta^2} = \Omega(n^{k-o(1)})$$

Combining the above with simulation sound zk-SNARKS, for example from [80], to reduce the proof size and verification time we get the following corollary.

Corollary 5.3.2. *Let $k \in \mathbb{N}$. Let $t(n) = \Omega(2^{\log^\epsilon(n)})$ and $\delta(n) > 0$ such that $T/\delta^2 = \Omega(n^{k-o(1)})$. Let L_k be as in Theorem 5.3.5. Fix x_0, x_1 such that $L_k(x_b) = b$. Let D_b define a variable, X, which is defined by rejection sampling the uniform distribution over $\{0,1\}^n$ conditioned on $L_k(X) = b$; if after $O(n)$ tries the rejection sampling* *has not succeeded, output* x_b.

Then, assuming BPSETH or BPkOVC, PKE, and simulation sound zk-SNARK, $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *is a computational* $(\mathit{BPTIME}(t(n)), \delta(n) + \mathsf{negl}(n))$*-non-malleable code.*

In other words, for all $A \in \mathit{BPP}$*,*

$$\mathsf{Tamper}_{A,0}^{\Pi,\mathit{BPTIME}(t(n))} \approx_{\delta(n)+\mathsf{negl}(n)} \mathsf{Tamper}_{A,1}^{\Pi,\mathit{BPTIME}(t(n))}$$

Note, however, that the tampering experiments are only inverse-polynomially indistinguishable (not negligible). Stronger bounds on the probability of correctness (δ) in Theorem 5.3.6 will yield stronger bounds on the tampering experiments.

5.4 Construction for Multi-Bit Messages

The construction for encoding multi-bit messages is similar to that for encoding a single bit, presented in section 5.3. The construction repeats the procedure for encoding single bit m times, for encoding m-bit messages and binds it with a proof T.

Let $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ be a public key encryption scheme with perfect correctness (see Definition 3.4.3). Let $\Pi^{\mathsf{NI}} = (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ be a non-interactive simulatable proof system with soundness against adversaries $\mathcal{A} \in \mathcal{G}$ (see Definition 3.5.4). Note that in the CRS model, we implicitly assume that all algorithms take the CRS as input, and for simplicity of notation, sometimes do not list the CRS as an explicit input.

$\mathsf{CRSGen}(1^n)$: Choose $(\text{PK}, \text{SK}) \leftarrow \mathsf{Gen}(1^n)$. Choose $[\mathsf{crs}^{\mathsf{NI}}_{i,j}, \tau^{i,j}_{\mathsf{sim}}]_{(i,j)=(0,0), i\in[m], j\in[n]} \leftarrow \mathsf{CRSGen}^{\mathsf{NI}}(1^n)$. Let $\overrightarrow{\mathsf{crs}}^{\mathsf{NI}} := [\mathsf{crs}^{\mathsf{NI}}_{i,j}]_{(i,j)=(0,0), i\in[m], j\in[n]}$ and let $\overrightarrow{\tau}_{\mathsf{sim}} := [\tau^{i,j}_{\mathsf{sim}}]_{(i,j)=(0,0), i\in[m], j\in[n]}$, and output $\mathsf{crs} := (\text{PK}, \overrightarrow{\mathsf{crs}}^{\mathsf{NI}})$.

<u>**Languages.**</u> We define the following languages:

- $\mathcal{L}^{\beta}_{i,j}$: For $i \in [m], j \in [n]$, $\beta \in \{0,1\}$, $s := ([\vec{\hat{k}}^i]_{i\in[m]}, \vec{c}, \bar{c}) \in \mathcal{L}^{\beta}_{i,j}$ iff the (i,j)-th ciphertext $c^i_j := k^i_j \oplus \beta$ (where $\vec{c} = [c^i_j]_{i\in[m],j\in[n]}$) and the (i,j)-th encryption $\hat{k}^i_j$ (where $\vec{\hat{k}}^i = \hat{k}^i_1, \ldots, \hat{k}^i_{n+1}$) is an encryption of k^i_j under PK (where PK is hardwired into the language).

- $\mathcal{L}$: $s := ([\vec{\hat{k}}^i]_{i\in[m]}, \vec{c}, \bar{c}) \in \mathcal{L}$ iff For each $i \in [m]$, $(x^i_1, \ldots, x^i_n)$ is in the support of D_{b^i} where:
  1. For $i \in [m], j \in [n]$, $x^i_j := c^i_j \oplus k^i_j$, and $b^i := c^i \oplus k^i_{n+1}$ (where $\bar{c} := c^1, \ldots, c^m$)
  2. $\vec{\hat{k}}^i$ is an encryption of $k^i_1, \ldots, k^i_{n+1}$ under PK (which is hardwired into the language).

$\mathsf{E}(\mathsf{crs}, \vec{b} := b^1, \ldots, b^m)$:

1. Sample $\vec{\vec{x}} := \vec{x}^1, \ldots, \vec{x}^m \leftarrow D_{\vec{b}}$, where for $i \in [m]$, $\vec{x}^i = x^i_1, \ldots, x^i_n$.
2. Choose an $m \cdot (n+1)$-bit key $\vec{\vec{k}} := [\vec{k}^i]_{i\in[m]} = [k^i_1, \ldots, k^i_n, k^i]_{i\in[m]}$ uniformly at random. For $i \in [m], j \in [n+1]$, compute $\hat{k}^i_j \leftarrow \mathsf{Encrypt}(\text{PK}, k^i_j)$. For $i \in [m]$, let $\vec{\hat{k}}^i := \hat{k}^i_1, \ldots, \hat{k}^i_{n+1}$. For $i \in [m], j \in [n]$, compute $c^i_j := k^i_j \oplus x^i_j$. Let $\vec{c} := [c^i_j]_{i\in[m],j\in[n]}$.
3. For $i \in [m]$, compute $c^i := k^i \oplus b^i$. Let $\bar{c} := [c^i]_{i\in[m]}$. For $i \in [m], j \in [n]$, compute a NI, simulatable proof T^i_j proving $([\vec{\hat{k}}^i]_{i\in[m]}, \vec{c}, \bar{c}) \in \mathcal{L}^{x^i_j}_{i,j}$ relative to $\mathsf{crs}^{\mathsf{NI}}_{i,j}$.
4. Compute a NI, simulatable proof T proving $([\vec{\hat{k}}^i]_{i\in[m]}, \vec{c}, \bar{c}) \in \mathcal{L}$ relative to $\mathsf{crs}^{\mathsf{NI}}_{0,0}$.
5. Output $\mathsf{CW} := ([\vec{\hat{k}}^i]_{i\in[m]}, \vec{c}, \bar{c}, T, [(x^i_j, T^i_j)]_{i\in[m],j\in[n]})$.

$\mathsf{D}(\mathsf{crs}, \mathsf{CW})$:

1. Parse $\mathsf{CW} := ([\vec{\hat{k}}^i]_{i\in[m]}, \vec{c}, \bar{c}, T, [(x^i_j, T^i_j)]_{i\in[m],j\in[n]})$
2. Check that V^{NI} outputs 1 on all proofs $[T^i_j]_{i\in[m],j\in[n]}, T$, relative to corresponding CRS. If yes, output $[b^i]_{i\in[m]}$ such that $x^i_1 \ldots x^i_n$ is in the support of D_{b^i}. Else, output $\vec{0}$.

Figure 5.7: Non-malleable code $(\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$, secure against $\mathcal{F}$ tampering.

$\mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r, \vec{b} := b^1, \ldots, b^m)$:

1. Sample $\vec{\vec{x}} := \vec{x}^1, \ldots, \vec{x}^m \leftarrow D_b$, where for $i \in [m]$, $\vec{x}^i = x^i_1, \ldots, x^i_n$.
2. Choose an $m \cdot (n+1)$-bit key $\vec{\vec{k}} := [\vec{k}^i]_{i \in [m]} = [k^i_1, \ldots, k^i_n, k^i]_{i \in [m]}$ uniformly at random. For $i \in [m], j \in [n+1]$, compute $\hat{k}^i_j \leftarrow \mathsf{Encrypt}(\textsc{pk}, k^i_j)$. For $i \in [m]$, let $\hat{\vec{k}}^i := \hat{k}^i_1, \ldots, \hat{k}^i_{n+1}$.
3. For $i \in [m], j \in [n]$, compute $c^i_j := k^i_j \oplus x^i_j$. Let $\vec{\vec{c}} := [c^i_j]_{i \in [m], j \in [n]}$.
4. For $i \in [m]$, compute $c^i := k^i \oplus b^i$. Let $\bar{c} := [c^i]_{i \in [m]}$.
5. For $i \in [m], j \in [n]$, simulate, using $\tau^{i,j}_{\mathsf{sim}}$ and r, a non-interactive proof T'^i_j proving $s := ([\hat{\vec{k}}^i]_{i \in [m]}, \vec{\vec{c}}, \bar{c}) \in \mathcal{L}^{x^i_j}_{i,j}$, relative to $\mathsf{crs}^{\mathsf{NI}}_{i,j}$.
6. Simulate, using $\tau^{0,0}_{\mathsf{sim}}$ and r, a non-interactive proof T' proving $s := ([\hat{\vec{k}}^i]_{i \in [m]}, \vec{\vec{c}}, \bar{c}) \in \mathcal{L}$,relative to $\mathsf{crs}^{\mathsf{NI}}_{0,0}$.
7. Output $\mathsf{CW} := ([\hat{\vec{k}}^i]_{i \in [m]}, \vec{\vec{c}}, \bar{c}, T', [(x^i_j, T'^i_j)]_{i \in [m], j \in [n]})$.

Figure 5.8: Encoding algorithm with simulated proofs.

$\mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r, \vec{b} := b^1, \dots, b^m)$:

1. Sample $\vec{\vec{x}} := \vec{x}^1, \dots, \vec{x}^m \leftarrow D_b$, where for $i \in [m]$, $\vec{x}^i = x^i_1, \dots, x^i_n$.
2. Choose $[c'^i_j]_{i \in [m], j \in [n]}$ uniformly at random. Let $\vec{c}' := [c'^i_j]_{i \in [m], j \in [n]}$.
3. Choose $[c'^i]_{i \in [m]}$ uniformly at random. Let $\overline{c}' := [c'^i]_{i \in [m]}$.
4. Set the $m \cdot (n+1)$-bit key $\vec{\vec{k}}' := [\vec{k}'^i]_{i \in [m]} = [c'^i_1, \dots, c'^i_n, c'^i]_{i \in [m]}$.
   For $i \in [m], j \in [n+1]$,
   compute $\hat{k}'^i_j \leftarrow \mathsf{Encrypt}(\textsc{pk}, k'^i_j)$. For $i \in [m]$, let $\hat{\vec{k}}'^i := \hat{k}'^i_1, \dots, \hat{k}'^i_{n+1}$.
5. For $i \in [m], j \in [n]$, simulate, using $\tau^{i,j}_{\mathsf{sim}}$ and r, a non-interactive proof T'^i_j proving $s := ([\hat{\vec{k}}^i]_{i \in [m]}, \vec{c}, \overline{c}) \in \mathcal{L}^{x^i_j}_{i,j}$, relative to $\mathsf{crs}^{\mathsf{NI}}_{i,j}$.
6. Simulate, using $\tau^{0,0}_{\mathsf{sim}}$ and r, a non-interactive proof T' proving $s := ([\hat{\vec{k}}^i]_{i \in [m]}, \vec{c}, \overline{c}) \in \mathcal{L}$, relative to $\mathsf{crs}^{\mathsf{NI}}_{0,0}$.
7. Output $\mathsf{CW} := ([\hat{\vec{k}}'^i]_{i \in [m]}, \vec{c}', \overline{c}', T', [(x^i_j, T'^i_j)]_{i \in [m], j \in [n]})$.

Figure 5.9: Encoding algorithm with simulated proofs and encryptions.

$\mathsf{Ext}(\mathsf{crs}, \textsc{sk}, \mathsf{CW})$:

1. Parse $\mathsf{CW} := ([\hat{\vec{k}}^i]_{i \in [m]}, \vec{c}, \overline{c}, T, [(x^i_j, T^i_j)]_{i \in [m], j \in [n]})$,
2. Output $[\mathsf{Decrypt}(\textsc{sk}, \hat{k}^i_{n+1})]_{i \in [m]}$.

Figure 5.10: Extracting procedure Ext.

$\mathsf{D}'(\mathsf{crs}, [k^i]_{i \in [m]}, \mathsf{CW})$:

1. Parse $\mathsf{CW} := ([\hat{\vec{k}}^i]_{i \in [m]}, , \vec{c}, \overline{c}, T, [(x^i_j, T^i_j)]_{i \in [m], j \in [n]})$,
2. Check that V^{NI} outputs 1 on all proofs $[T^i_j]_{i \in [m], j \in [n]}, T$, relative to the corresponding CRS,
3. For $i \in [m]$, output $b^i := k^i \oplus c^i$.

Figure 5.11: Alternate decoding procedure D', given additional extracted key $[k^i]_{i \in [m]}$ as input.

$g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r)$:

1. Parse $\mathsf{CW} = ([\hat{\vec{k}}^i]_{i\in[m]}, \vec{c}, \bar{c}, T, [(x^i_j, T^i_j)]_{i\in[m], j\in[n]})$, $\mathsf{CW}^* = ([\hat{\vec{k}}^{*i}]_{i\in[m]}, \vec{c}^*, \bar{c}^*, T^*, [(x^{*i}_j, T^{*i}_j)]_{i\in[m], j\in[n]})$.
2. If (1) V^{NI} outputs 1 on all proofs $T^*, [T^{*i}_j)]_{i\in[m], j\in[n]}$, relative to the corresponding CRS; and (2) $([\hat{\vec{k}}^i]_{i\in[m]}, \vec{c}, \bar{c}) = ([\hat{\vec{k}}^{*i}]_{i\in[m]}, \vec{c}^*, \bar{c}^*)$, then output 1. Otherwise output 0.

Figure 5.12: THE PREDICATE $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r)$.

Let $\Psi(p, c, x, y, r, z)$ be defined as a function that takes as input a predicate p, and variables c, x, y, r, z. If $p(c, x, y, r) = 1$, then Ψ outputs the m-bit string $\vec{0}$. Otherwise, Ψ outputs z.

Theorem 5.4.1. *Let* (E, D), E_1, E_2, Ext, D' *and* g *be as defined in Figures 5.7, 5.8, 5.9, 5.10, 5.11 and 5.12. Let* $\mathcal{F}$ *be a computational class. If, for every pair of* m*-bit messages* $\vec{b}_0, \vec{b}_1$ *and if, for every adversary* $\mathcal{A} \in \mathcal{G}$ *outputting tampering functions* $f \in \mathcal{F}$*, all of the following hold:*

- ***Simulation of proofs.***

  1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_0, f(\mathsf{CW}_0), r_0) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1) = 1]$,

  2. $\Psi(g, \mathsf{crs}, \mathsf{CW}_0, f(\mathsf{CW}_0), r_0, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_0); r_0)) \approx$

  $\Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1))$,

  where $(\mathsf{crs}, \mathrm{SK}, \vec{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_0, r_1 *are sampled uniformly at random,* $\mathsf{CW}_0 \leftarrow \mathsf{E}(\mathsf{crs}, \vec{b}_0)$ *and* $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \vec{\tau}_{\mathsf{sim}}, r_1, \vec{b}_0)$.

- ***Simulation of Encryptions.***

1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1]$,

2. $\Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1)) \approx$

   $\Psi(g, \mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2))$,

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_1, r_2 *are sampled uniformly at random,* $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, \vec{b}_0)$ *and* $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, \vec{b}_0)$.

- ***Simulation Soundness.***

  $\Pr_r[\mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2) \neq \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \wedge$

  $g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 0] \leq \mathsf{negl}(n)$,

  where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_2 *is sampled uniformly at random and* $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r, \vec{b}_0)$.

- ***Hardness of $D_{\vec{b}}$ relative to Alternate Decoding.***

  1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_3, f(\mathsf{CW}_3), r_3) = 1]$,

  2. *For every Boolean function, represented by a circuit F over m variables,*

     $F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \approx$

     $F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$,

  where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_2, r_3 *are sampled uniformly at random,* $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, \vec{b}_0)$ *and* $\mathsf{CW}_3 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_3, \vec{b}_1)$.

Then the construction presented in Figure 5.7 is a non-malleable code for class $\mathcal{F}$ against adversaries $\mathcal{A} \in \mathcal{G}$.

We present the proof of theorem 5.4.1 next.

5.4.1 Generic Analysis

Similarly to the one-bit case, we take g to be the predicate that is used in the $\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,m,g}(n)$ tampering experiment. We must argue that for every $m \in \Sigma$ and every attacker $A \in \mathcal{G}$ the output of the experiment $\mathsf{Expt}^{\Pi,\mathcal{F}}_{A,m,g}(n)$ is 1 with at most negligible probability

Assume towards contradiction that for some $A \in \mathcal{G}$ the output of the experiment is 1 with non-negligible probability. Then this means that the probability in the last line of experiment $\mathsf{Expt}^{\Pi,\mathcal{F}}_{A,m,g}(n)$ that $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r) = 1 \wedge \mathsf{D}(\mathsf{crs}, \mathsf{CW}^*; r) \neq m$ is non-negligible. Parse $\mathsf{CW} = ([\hat{\vec{k}}^i]_{i\in[m]}, \vec{\overline{c}}, \overline{c}, T, [(x^i_j, T^i_j)]_{i\in[m],j\in[n]})$, $\mathsf{CW}^* = ([\hat{\vec{k}}^{*i}]_{i\in[m]}, \vec{\overline{c}}^*, \overline{c}^*, T^*, [(x^i_j, T^{*i}_j]_{i\in[m],j\in[n]})$.

Recall that $\mathsf{D}(\mathsf{crs}, \mathsf{CW}; r) = m$. Thus, if the above event occurs, it means that $\mathsf{D}(\mathsf{crs}, \mathsf{CW}; r) \neq \mathsf{D}(\mathsf{crs}, \mathsf{CW}^*; r)$. But since $g(\mathsf{crs}, \mathsf{CW}, \mathsf{CW}^*, r) = 1$, it means that V^{NI} outputs 1 on all proofs $T^*, [T^{*i}_j)]_{i\in[m],j\in[n]}$ and $([\hat{\vec{k}}^i]_{i\in[m]}, \vec{\overline{c}}, \overline{c}) = ([\hat{\vec{k}}^{*i}]_{i\in[m]}, \vec{\overline{c}}^*, \overline{c}^*)$. This, in turn, means that there must be some bit x^i_j, x^{*i}_j that CW and CW^* differ on. But note that by assumption $c^i_j = c^{*i}_j$. Due to the fact that CW is well-formed and perfect correctness of the encryption scheme, it must mean that $c^{*i}_j \notin \mathcal{L}^{x^{*i}_j}_{i,j}$. But recall that by assumption, proof T^{*i}_j verifies correctly. This means that soundness is broken by $A \in \mathcal{G}$. This contradicts the security of the proof system Π^{NI}.

Next, recall that we wish to show that for any $\vec{b}_0, \vec{b}_1$ and any adversary $A \in \mathcal{G}$ outputting tampering function $f \in \mathcal{F}$,

$\{\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,\vec{b}_0,g}\}_{k\in\mathbb{N}} \approx \{\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,\vec{b}_1,g}\}_{k\in\mathbb{N}}$

To do so we consider the following hybrid argument, which proceeds almost identically to the hybrid argument for the one-bit case:

Hybrid 0: The real game, $\mathsf{MediumTamper}^{\Pi,\mathcal{F}}_{A,\vec{b}_0,g}$, relative to g, where the real encoding $\mathsf{CW}_0 \leftarrow \mathsf{E}(\mathsf{crs},\vec{b}_0)$ and the real decoding oracle D are used.

Hybrid 1: Replace the encoding from the previous game with $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs},\overrightarrow{\tau}_{\mathsf{sim}},r_1,\vec{b}_0)$ where r_1 is chosen uniformly at random and g, D use random coins r_1.

Hybrid 2: Replace the encoding from the previous game with $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs},\overrightarrow{\tau}_{\mathsf{sim}},r_2,\vec{b}_0)$, where r_2 is chosen uniformly at random and g, D use random coins r_2.

Hybrid 3: Replace the decoding from the previous game, with $\mathsf{D}'(\mathsf{crs},\mathsf{Ext}(\mathsf{crs},\textsc{sk},f(\mathsf{CW}_2)),f(\mathsf{CW}_2);r_2)$. where r_2 is chosen uniformly at random and g, E_2 use random coins r_2.

Hybrid 4: Same as Hybrid 3, but replace the encoding with $\mathsf{CW}_3 \leftarrow \mathsf{E}_2(\mathsf{crs},\overrightarrow{\tau}_{\mathsf{sim}},r_3,\vec{b}_1)$, where r_3 is chosen uniformly at random and g, D' use random coins r_3.

The proofs of indistinguishability of consecutive hybrid distributions follow identically to the one bit case, except for the final hybrid.

Claim 5.4.1. *Hybrid 3 is computationally indistinguishable from Hybrid 4.*

Proof. First note that for $\gamma \in \{2,3\}$ if $g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1$ then $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_\gamma)), f(\mathsf{CW}_\gamma); r_\gamma)$ always outputs $\vec{0}$ and so

$$\begin{aligned}&\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_\gamma)), f(\mathsf{CW}_\gamma); r_\gamma)\\ &\qquad \equiv \Psi(g, \mathsf{crs}, \mathsf{CW}_\gamma, f(\mathsf{CW}_\gamma), r_\gamma, \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_\gamma)), f(\mathsf{CW}_\gamma); r_\gamma)).\end{aligned}$$

Now, assume towards contradiction that the two distributions $\Psi(g, \mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2, \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2))$ and $\Psi(g, \mathsf{crs}, \mathsf{CW}_3, f(\mathsf{CW}_3), r_3, \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3))$ are distinguishable. By the above, this implies that $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ and $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$ are distinguishable. Note that since D' outputs m bits, this implies that there exists a distinguishing circuit F over m-bit inputs such that

$$\begin{aligned}&\Big|\Pr[F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) = 1]\\ &\qquad - \Pr[F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)] = 1\Big| \geq \mathsf{negl}(n).\end{aligned}$$

But this yields a contradiction to the **Hardness of $D_{\vec{b}}$ relative to Alternate Decoding** property in Theorem 5.4.1. □

5.4.2 Efficient, Multi-Bit NMC for AC^0

Theorem 5.4.2. $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *(presented in Figure 5.7) is an m-bit, computational, non-malleable code in the CRS model against tampering by depth-*$(m^{\log^\delta m}/2-$

c) circuits with unbounded fan-in and size $\delta \cdot \frac{\log m}{\log\log m} - p(n)$ *(where c is constant and* $p(\cdot)$ *is a fixed polynomial), and* m *is such that* $n = m^{3+5\delta}$*, if the underlying components are instantiated in the following way:*

- $\mathcal{E} := (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is a public key encryption scheme with perfect correctness and decryption in* AC^0 *.*
- $\Pi^{\mathsf{NI}} := (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ *is a same-string, weak one-time simulation-sound NIZK with verifier in* AC^0 *.*
- *For* $b \in \{0,1\}$*,* D_b *is the distribution that samples bits* $x_1 \dots x_n$ *uniformly at random, conditioned on* $x_1 \oplus \cdots \oplus x_n = b$*.*

For as in the one-bit case, given Theorem 5.2.2, proof systems Π^{NI} as above exist, under the assumption that same-string, weak one-time simulation-sound NIZK with (arbitrary polynomial-time) deterministic verifier exists. See the beginning of Section 5.3.2 for a discussion of how such NIZK and public key encryption can be instantiated.

Before proving the theorem, we state some claims on Fourier concentration of AC^0 circuits and then prove Claim 5.4.3, which will be used in the proof of Theorem 5.4.2.

Claim 5.4.2 ([117]). AC^0 *circuits of depth* d *and size* k *have at most* $2^{-\Omega(n/(\log k)^{d-1}}$ *of their Fourier mass at level* n *or above.*

Setting $d = (2+\delta) \cdot \frac{\log m}{\log\log m}$, $k = m^{\log^{\delta} m}$, $n = m^{3+5\delta}$, for constant $0 \leq \delta < 1$, and noting that

$$\frac{n}{(\log k)^{d-1}} \geq \frac{n}{(\log k)^{d}} = \frac{m^{3+5\delta}}{(\log m)^{(1+\delta)d}} = \frac{m^{3+5\delta}}{2^{(1+\delta)d\cdot\log\log m}} = \frac{m^{3+5\delta}}{2^{(1+\delta)(2+\delta)\log m}}$$
$$= \frac{m^{3+5\delta}}{m^{2+3\delta+\delta^2}} = m^{1+2\delta-\delta^2} \in \Omega(m^{1+\delta}),$$

We have the following corollary:

Corollary 5.4.1. *An AC^0 circuit of depth $d = (2+\delta)\cdot\frac{\log m}{\log\log m}$ and size $k = m^{\log^{\delta} m}$ has at most $\epsilon \in 2^{-\Omega(m^{1+\delta})}$ of its Fourier mass at level $n := m^{3+5\delta}$ or above.*

We now prove the main technical claim of this section:

Claim 5.4.3. *Let n be security parameter. Let $C \in \mathsf{AC}^0$ be a circuit of depth $d \leq (2+\delta)\cdot\frac{\log m}{\log\log m}$ and size $k \leq m^{\log^{\delta} m}$ that takes inputs $\vec{x}$ of length n bits. Let m be such that $n = m^{3+5\delta}$, where $0 < \delta \leq 1$. For $\gamma \in \{0,1\}$ let $\vec{X}^{\gamma}$ be a random variable distributed as D_{γ}. Then for every Boolean function F over m variables,*

$$|\Pr[F(C(\vec{X}^0)) = 1] - \Pr[F(C(\vec{X}^1)) = 1]| \in 2^{-\Omega(m^{\delta})}.$$

Note, the above claim implies that

$$F(C(\vec{X}^0)) \overset{s}{\approx} F(C(\vec{X}^1)).$$

of Claim 5.4.3. The conclusion of the claim is implied by showing that $|\Pr[F(C(\vec{x})) = 1 \mid \mathsf{PAR}(\vec{x}) = 1] - \Pr[F(C(\vec{x})) = 1 \mid \mathsf{PAR}(\vec{x}) = -1]| \in 2^{-\Omega(m^{\delta})}$, where the probability is taken over choice of $\vec{x}$ from the distribution which sets $\vec{x} \leftarrow D_0$ with probability

1/2 and $\vec{x} \leftarrow D_1$ with probability 1/2. Thus, in order to prove the claim, it is sufficient to show that for every (inefficient) distinguisher F,

$$|E[F \circ C(\vec{x}) \cdot \mathsf{PAR}(\vec{x})]| \in 2^{-\Omega(m^\delta)}.$$

Recall that the correlation of $F \circ C$ with $\mathsf{PAR}(\vec{x})$ is defined as $|E[F \circ C(\vec{x}) \cdot \mathsf{PAR}(\vec{x})]|$. Thus, to complete the proof, we must show that for every (inefficient) F, the correlation of $F \circ C$ with $\mathsf{PAR}(\vec{x})$ is negligible.

Analyzing the correlation of $\chi_S \circ C$ with $\mathsf{PAR}(\vec{x})$. First, note that since each output bit of C, computed by C_i, $i \in [m]$ is in AC^0 it has depth at most $\delta \cdot \frac{\log n}{\log\log n}$.

We next claim that for $S \subseteq [m]$, there is a circuit computing $\chi_S \circ C(\vec{x}) = \chi_S(C_1(\vec{x}), \ldots, C_n(\vec{x}))$ of depth at most $d = (2+\delta) \cdot \frac{\log m}{\log\log m}$ and size at most $k = m^{\log^\delta m}$.

This follows since the circuit for $\chi_S(C_1(\vec{x}), \ldots, C_m(\vec{x}))$ can be constructed by computing $C(\vec{x}) := C_1(\vec{x}), \ldots, C_m(\vec{x})$ in size $m^{\log^\delta m}/2$ and depth $\delta \cdot \frac{\log m}{\log\log m}$ and then feeding this into a circuit that computes parity over (at most) m bits, which (by recursively computing parity over $\log m$ bits in depth 2 and polynomial size), has size $m^{\log^\delta m}/2$ and depth $2\frac{\log m}{\log\log m}$.

By plugging in Claim 5.4.1, we have that $\left(\widehat{\chi_S \circ C}([n])\right)^2 = \epsilon$. Since $|E[\chi_S \circ C(\vec{x}) \cdot \mathsf{PAR}(\vec{x})]| = \widehat{\chi_S \circ C}([n])$, we have that for $S \subseteq [m]$, the correlation of

$\chi_S(C_1(\vec{x}),\ldots,C_n(\vec{x}))$ with $\mathsf{PAR}(\vec{x})$ is at most $\sqrt{\epsilon} \in 2^{-\Omega(m^{1+\delta})}$:

$$|E[\chi_S \circ C(\vec{x}) \cdot \mathsf{PAR}(\vec{x})]| \leq \sqrt{\epsilon}. \tag{5.4.1}$$

Analyzing the correlation of $F \circ C$ with $\mathsf{PAR}(\vec{x})$. Since $F \circ C(\vec{x}) = \sum_{S \subseteq [m]} \hat{F}(S) \cdot \chi_S(C_1(\vec{x}),\ldots,C_n(\vec{x}))$, we have that

$$\begin{aligned}
|E[F \circ C(\vec{x}) \cdot \mathsf{PAR}(\vec{x})]| &= |\sum_{S\subseteq[m]} \hat{F}(S)E[\chi_S(C_1(\vec{x}),\ldots,C_m(\vec{x})) \cdot \mathsf{PAR}(\vec{x})]| \\
&\leq \sum_{S\subseteq[m]} |\hat{F}(S)||E[\chi_S(C_1(\vec{x}),\ldots,C_m(\vec{x})) \cdot \mathsf{PAR}(\vec{x})]| \qquad (5.4.2)\\
&\leq 2^m \cdot \sqrt{\epsilon} \in 2^{-\Omega(m^{\delta})}, \qquad (5.4.3)
\end{aligned}$$

where (5.1) follows by the triangle inequality and (5.2) follows from (5.4.1) and the fact that for all $S \subseteq [n]$, $|\hat{F}(S)| \leq 1$.

So we have shown that $|E[F \circ C(\vec{x}) \cdot \mathsf{PAR}(\vec{x})]|$ is negligible (in m and therefore also in n, since m and n are polynomially related), thus completing the proof. □

We are now ready to complete the proof of the theorem.

of Theorem 5.4.2. The proof proceeds identically to the one-bit proof, until we reach the final property:

Hardness of D_b relative to Alternate Decoding.

1. $\Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1] \approx \Pr[g(\mathsf{crs}, \mathsf{CW}_3, f(\mathsf{CW}_3), r_3) = 1]$,

2. For every Boolean function, represented by a circuit F over m variables,

   $F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \approx$

   $F \circ \mathsf{D}'(\mathsf{Ext}(\mathsf{crs}, \mathrm{SK}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$,

where $(\mathsf{crs}, \mathrm{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_2, r_3 are sampled uniformly at random, $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_2, \vec{b}_0)$ and $\mathsf{CW}_3 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_3, \vec{b}_1)$.

We consider a sequence of distributions where we switch the internal random variables of E_2 from from $\vec{x}^i \leftarrow D_{b_0^i}$, for all $i \in [m]$ to $\vec{x}^i \leftarrow D_{b_1^i}$, for all $i \in [m]$. Namely, for each $i \in \{0, \ldots, m\}$ we consider a distribution where for $j \leq i$, $\vec{x}^j \leftarrow D_{b_1^i}$ and for $j > i$, $\vec{x}^j \leftarrow D_{b_0^i}$.

We must show that (1) and (2) hold for each consecutive pair of distributions. When considering the i-th consecutive pair, fix all random variables except the i-th variable $\vec{X}^i$ to values $\vec{x}^1, \ldots, \vec{x}^{i-1}, \vec{x}^{i+1}, \ldots, \vec{x}^m$. Let $\vec{X}^i$ be a random variable such that with probability $1/2$, $\vec{X}^i \leftarrow D_{b_0^i}$ and with probability $1/2$, $\vec{X}^i \leftarrow D_{b_1^i}$. $\vec{X}^i = \vec{X}^{i,\gamma}$ where $\gamma \leftarrow \{0,1\}$, and let random variable CW^i denote the output of E_2 when using random variables $\vec{x}^1, \ldots, \vec{x}^{i-1}, \vec{X}^i, \vec{x}^{i+1}, \ldots, \vec{x}^m$.

To show (1), assume $\Pr[g(\mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1]$ and $\Pr[g(\mathsf{crs}, \mathsf{CW}_3, f(\mathsf{CW}_3), r_3) = 1]$ differ by a non-negligible amount. This implies that, for some $i \in [m]$, there is a circuit that takes as input $\vec{X}^i$, hardwires all other random variables, and outputs 1 in the case that $g(\mathsf{crs}, \mathsf{CW}^i, f(\mathsf{CW}^i), r) = 1$ and 0 otherwise, implying that it has non-negligible correlation to the parity of its input $\vec{X}^i$. We will show that the above can be computed by an AC^0 circuit with input $\vec{X}^i$, thus contradicting Theorem 3.3.1, which says that an AC^0 circuit has at most negligible correlation with parity of its input $\vec{X}^i$, denoted $\mathcal{P}(\vec{X}^i)$. Details follow.

We construct the distribution of circuits $\mathcal{C}^1_{\mathcal{F}}$. A draw $C \sim \mathcal{C}^1_{\mathcal{F}}$ is done as follows:

1. Sample $(\mathsf{crs}, \text{SK}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$.

2. Sample tampering function $\mathcal{A}(\mathsf{crs}) \rightarrow f$.

3. Sample $\vec{\vec{c}}'$, $\vec{c}'$ uniformly at random,

4. Set the $m \cdot (n+1)$-bit key $\vec{\vec{k}}' := [\vec{k}'^{i}]_{i \in [m]} = [c'^{i}_1, \ldots, c'^{i}_n, c'^{i}]_{i \in [m]}$. For $i \in [m], j \in [n+1]$, compute $\hat{k}'^{i}_j \leftarrow \mathsf{Encrypt}(\text{PK}, k'^{i}_j)$. For $i \in [m]$, let $\hat{\vec{k}}'^{i} := \hat{k}'^{i}_1, \ldots, \hat{k}'^{i}_{n+1}$.

5. Sample r uniformly at random.

6. Sample simulated proofs $[T'^{\beta,i}_j]_{\beta \in \{0,1\}, i \in [m], j \in [n]}$ and T' (as described in Figure 5.8).

7. Sample $\vec{x}^1, \ldots, \vec{x}^{i-1}$ from $D_{b^i_0}$, and $\vec{x}^{i+1}, \ldots, \vec{x}^m$ from $D_{b^i_1}$.

8. Output the following AC^0 circuit C that has the following structure:

   - **hardcoded variables:** crs, SK, f, $[\hat{\vec{k}}'^{i}]_{i \in [m]}$, $\vec{\vec{c}}'$, $\vec{c}'$, r, $[T'^{\beta,i}_j]_{\beta \in \{0,1\}, i \in [m], j \in [n]}$, $\vec{x}^1, \ldots, \vec{x}^{i-1}, \vec{x}^{i+1}, \ldots, \vec{x}^m$.

   - **input:** $\vec{X}^i$.

   - **computes and outputs:**

$$g(\mathsf{crs}, \mathsf{CW}, f(\mathsf{CW}), r).$$

Note that given all the hardwired variables, computing CW is in AC^0 since all it does is, for $j \in [n]$, select the correct simulated proof $T_j^{\prime X_j^i, i}$ based on the corresponding input bit X_j^i. Additionally, f in AC^0 and g in AC^0 , since bit-wise comparison is in AC^0 and V^{SAT} is in AC^0 . Thus, the entire circuit is in AC^0 .

To show (2), assume $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ and $\mathsf{D}'(\mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$ have non-negligible statistical distance. This implies that there exists a distinguisher F (represented by an m-bit Boolean function) such that $F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ is far from $F \circ \mathsf{D}'(\mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$. This implies that, for some $i \in [m]$, the output of $F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}^i)), f(\mathsf{CW}^i); r_i)$ is correlated with the parity of its input $\vec{X}^i$. We will show that $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}^i)), f(\mathsf{CW}^i); r_i)$ can be computed by an AC^0 circuit C (drawn from some distribution $\mathcal{C}$) with input $\vec{X}^i$. We then use Claim 5.4.3, which says that if C is an AC^0 circuit taking inputs of length n bits and F is any m-bit function then the output $F(C(\vec{X}^i))$, conditioned on the parity of $\vec{X}^i$ being 0 is statistically close to the output $F(C(\vec{X}^i))$, conditioned on the parity of $\vec{X}^i$ being 1. This yields a contradiction, since it means that $F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}^i)), f(\mathsf{CW}^i); r_i)$ cannot be correlated with the parity of its input $\vec{X}^i$. Details follow.

We construct the distribution of circuits $\mathcal{C}_{\mathcal{F}}^2$. A draw $C \sim \mathcal{C}_{\mathcal{F}}^2$ is done as follows:

1. Sample $(\mathsf{crs}, \textsc{sk}, \vec{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$.

2. Sample tampering function $\mathcal{A}(\mathsf{crs}) \rightarrow f$.

3. Sample $\vec{c}', \overline{c}'$ uniformly at random,

4. Set the $m \cdot (n+1)$-bit key $\vec{\vec{k}}' := [\vec{k}'^i]_{i \in [m]} = [c_1'^i, \ldots, c_n'^i, c'^i]_{i \in [m]}$. For $i \in [m], j \in [n+1]$, compute $\hat{k}_j'^i \leftarrow \mathsf{Encrypt}(\textsc{pk}, k_j'^i)$. For $i \in [m]$, let $\hat{\vec{k}}'^i := \hat{k}_1'^i, \ldots, \hat{k}_{n+1}'^i$.

5. Sample r uniformly at random.

6. Sample simulated proofs $[T_j'^{\beta,i}]_{\beta \in \{0,1\}, i \in [m], j \in [n]}$ and T' (as described in Figure 5.8).

7. Sample $\vec{x}^1, \ldots, \vec{x}^{i-1}$ from $D_{b_0^i}$, and $\vec{x}^{i+1}, \ldots, \vec{x}^m$ from $D_{b_1^i}$.

8. Output the following AC^0 circuit C that has the following structure:

   - **hardcoded variables:** crs, $\textsc{sk}$, f, $[\hat{\vec{k}}'^i]_{i \in [m]}$, $\vec{c}', \overline{c}'$, r, $[T_j'^{\beta,i}]_{\beta \in \{0,1\}, i \in [m], j \in [n]}$, $\vec{x}^1, \ldots, \vec{x}^{i-1}, \vec{x}^{i+1}, \ldots, \vec{x}^m$.

   - **input:** $\vec{X}^i$.

   - **computes and outputs:**

$$\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}^i)), f(\mathsf{CW}^i); r_i)$$

.

Note that $\mathsf{Ext} \in \mathsf{AC}^0$ since decryption for $\mathcal{E} := (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ in AC^0 . Moreover, as above, given all the hardwired variables, computing CW^i is in AC^0 since all it does is, for $j \in [n]$, select the correct simulated proof $T_j'^{X_j^i, i}$ based on the corresponding input bit X_j^i. Additionally, f in AC^0 and D' is in

AC^0 , since xor of two streams of bits is in AC^0 and V^{SAT} is in AC^0 . Thus, the entire circuit is in AC^0 .

□

Remark 5.4.1. *We wish to highlight that for* AC^0 *tampering, the condition on the encryption scheme used in Sections 5.3.2, and 5.4.2, that* Decrypt *algorithm must be in* AC^0 *can be relaxed. Note that in the proof we need to show that for PPT distinguisher* Dist*, the entire tampering experiment can be simulated in* AC^0 *after we switch the decoding algorithm to alternate decoding* D'*. On a high level the main difference between the decoding algorithm* D *and the alternate decoding algorithm* D' *is that if the NIZK proofs are accepted then,* D' *uses the "trapdoor" (secret key) to decrypt the ciphertext and recover the underlying message. In the proof, we need to argue that* Dist *cannot distinguish between the output of alternate decoding* D' *in the two tampering experiments when the input distribution is switched from* D_0 *to* D_1*. Recall that this is argued by invoking incompressibility of parity function (over* n *bits)* PAR_n *and noting that the length of the output of* D' *is* $m << n$*. We can split the alternate decoding algorithm* D'*, as follows:* $\mathsf{D}' = \mathsf{D}'_1 \circ \mathsf{D}'_2$ *where, (1)* D'_2 *first checks if the NIZK proofs are accepted and if so, simply output the ciphertext corresponding to the underlying message (which is part of the codeword), and then (2)* D'_1 *finds the underlying message by brute-force, instead of decrypting the ciphertext. Now, instead of proving* Dist *cannot distinguish between the tampering experiments when the input distribution is switched from* D_0 *to* D_1*, we need to show that* $\mathsf{Dist} \circ \mathsf{D}'_1$ *cannot distinguish between the tampering experiments when the input distribution is* *switched from D_0 to D_1. However, since the output of $\mathsf{D'}_2$ is the size of m ciphertexts (so $m \cdot \lambda$), by setting n polynomial sufficiently large even an unbounded $\mathsf{Dist} \circ \mathsf{D'}_1$ is not correlated with the parity due to the incompressibility theorem (Theorem 5.2.1). Thus, by appropriate choice of parameters such that n is sufficiently large compared to m number of ciphertexts encrypting a single bit, we no longer need that $\mathsf{Decrypt} \in \mathsf{AC}^0$.*[1]

5.4.3 Tampering with decision trees

Theorem 5.4.3. $\Pi = (\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *(presented in Figure 5.7) is an m-bit, computational, non-malleable code in the CRS model against tampering by depth-d circuits with unbounded fan-in and size $\leq 2^{m^\varepsilon}$ (where d, ε are constants), and m is such that $n = m^{1+\varepsilon}$, if the underlying components are instantiated in the following way:*

- $\mathcal{E} := (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$ *is a public key encryption scheme with perfect correctness and decryption in AC^0 .*
- $\Pi^{\mathsf{NI}} := (\mathsf{CRSGen}^{\mathsf{NI}}, \mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ *is a same-string, weak one-time simulation-sound NIZK with verifier in AC^0 .*
- *For $b \in \{0, 1\}$, D_b is the distribution that samples bits $x_1 \dots x_n$ uniformly at random, conditioned on $x_1 \oplus \dots \oplus x_n = b$.*

The proof of this theorem follows exactly as the proof of Theorem 5.4.2, except we replace Claim 5.4.3 with Claim 5.4.4 below. But first, we present a simple corollary of the theorem.

[1] We thank Vinod Vaikuntanathan and the other attendees at CIS seminar, MIT on February 23^{rrd}, 2018, for insightful discussion which brought this improvement to our notice.

Corollary 5.4.2. *Under the assumptions of Theorem 5.4.3,* $(\mathsf{CRSGen}, \mathsf{E}, \mathsf{D})$ *is an m-bit, computational, non-malleable code against tampering by decision trees of complexity $\leq m^{\varepsilon}$, where $0 < \varepsilon \leq 1$ is a constant, and $n = m^{1+\varepsilon}$.*

This follows from the above theorem when put together with the fact that decision trees of depth t can be represented as a disjunction of 2^t terms (each term is a path to some 1).

Claim 5.4.4. *Let n be security parameter. Fix some $d \in \mathbb{Z}$. Let $\varepsilon \geq 1/d$. Let m be such that $n = m^{1+\varepsilon}$ Let $C : \{0,1\}^n \to \{0,1\}^m$ be composed of depth-d circuit with unbounded fan-in and size $s = 2^{m^{\varepsilon}}$. For $\gamma \in \{0,1\}$ let $\vec{X}^{\gamma}$ be a random variable distributed as D_{γ}. Then for every Boolean function $F : \{0,1\}^m \to \{0,1\}$ over m variables,*

$$\left|\Pr[F \circ C(\vec{X}^0) = 1] - \Pr[F \circ C(\vec{X}^1) = 1]\right| \leq \frac{1}{2} + \frac{(k+1)}{2\exp(m^{\varepsilon})}.$$

Proof. Let $n = m^{1+\varepsilon}/q$ for $1 > \varepsilon \geq 1/d$. Let ρ be a random restriction over $\{0,1\}^{\ell}$ such that $\Pr[\star] = q = \frac{1}{18^d e^d m^{d\varepsilon}} \leq \frac{1}{18^d e^d m}$. Let $t = m^{\varepsilon}$. Then, by lemma 3.3.1 and a union bound the probability that some output bit of C_{ρ} cannot be represented by a decision tree of depth $t-2$ is at most $ms(9q^{1/d}t)^t$.

$$ms(9q^{1/d}t)^t \leq m2^{m^{\varepsilon}}(\frac{9m^{\varepsilon}}{18em^{\varepsilon}}m^{\varepsilon})^{m^{\varepsilon}} \tag{5.4.4}$$

$$= m\exp(-m^{\varepsilon}). \tag{5.4.5}$$

If C_{ρ} can be represented by a decision tree of depth $t-1$ call C_{ρ} "simple."

Any $F : \{0,1\}^m \to \{0,1\}$ has decision tree complexity at most than m. If we compose this with a simple C_ρ, the resulting decision tree has complexity $< m^{1+\varepsilon} - 1$ [116].

Additionally, by standard Chernoff bounds, the probability that ρ contains more than $4\ell q$ $\star$'s is at most $\exp(-m^\varepsilon)$. Call such a ρ "bad." Note that if this is the case, then C_ρ is a function over at least $m^{1+\varepsilon}$ variables.

If neither event happens, ρ is not bad and C_ρ is simple, then $F \circ C_\rho$ is completely uncorrelated with parity. Otherwise, the correlation is bounded by 1. Therefore, we can simply bound correlation with the probability that either ρ is bad or C_ρ is not simple: $(m+1)\exp(-m^\varepsilon)$. $\square$

5.5 One-Bit NMC Against Streaming Adversaries

We begin by describing constructions of the underlying components required to instantiate the generic constructions in the streaming adversaries setting.

In the following, we assume that the tampering class $\mathcal{F}$ corresponds to streaming adversaries with memory $o(n'')$. We then choose parameter $n \in \omega(n'')$ and parameter $n' \in \omega(n)$. n is the parameter for the hard distribution described in Section 5.5.1, n' is the parameter for the encryption scheme described in Section 5.5.2, n'' is the parameter for the weak encryption scheme $(\mathsf{Hide}, \mathsf{Rec})$ *described in Section 5.5.3.*

5.5.1 The Hard Distribution D_b (parameter n)

Let $n = (\mu + 1)^2 - 1$

For $b \in \{0, 1\}$*, a draw from the distribution* D_b *is defined as follows: Choose a parity* χ_S *uniformly at random from the set of all (non-zero) parities over* μ *variables (*$\emptyset \neq S \subseteq [\mu]$*). Choose* $y_1, \ldots, y_\mu \sim \{0,1\}^\mu$ *uniformly at random. Choose* y *uniformly at random, conditioned on* $\chi_S(y) = b$*. Output the following* n*-bit string:* $[(y_i, \chi_S(y_i)]_{i \in [\mu]} || y$.

The hardness of the distribution follows from Theorems 5.2.3 and lemma 5.2.1.

Claim 5.5.1. *Let* A *be a streaming algorithm with* $o(n)$ *space, and* $\alpha > 0$*. Then,*

$$\| \Pr_{x \sim D_0}[A(x) = 0] - \Pr_{x \sim D_1}[A(x) = 0] \| \leq 2^{\alpha n/3}.$$

5.5.2 Encryption scheme $\mathcal{E} = (\mathsf{Encrypt}, \mathsf{Decrypt})$ (parameter $n' \in \omega(n)$)

The Learning Parity problem yields an encryption scheme with semantic security against streaming adversaries with $o(n')$ *storage. We can use this encryption scheme to encrypt the key* k*, bit-by-bit, thus yielding an encryption scheme with the necessary properties.*

To encrypt a bit b*,* $\mathsf{Encrypt}(b)$ *outputs* z*, where* $z \sim D_b$ *and* D_b *is the same as above, except with parameter* n'*.*

To decrypt a ciphertext z*, with* $\Theta(n')$ *storage,* $\mathsf{Decrypt}(z)$ *runs the parity learn-*

ing algorithm to recover b.

Renaming variables and plugging in Claim 5.5.1 from above, we have

Claim 5.5.2. *Let A be a streaming algorithm with $o(n)$ space, and $\alpha > 0$. Then,*

$$\| \Pr_{z \sim \mathsf{Encrypt}(0)}[A(z) = 0] - \Pr_{z \sim \mathsf{Encrypt}(1)}[A(z) = 0] \| \leq 2^{\alpha n/3}.$$

5.5.3 Weak Encryption Scheme (parameter $n'' \in o(n)$)

Let $n'' = (\mu'' + 1)^2 - 1$ Given a bit string $k = k_1, \ldots, k_{\mu''}$ of length μ'' bits, and a vector $y := y_1, \ldots, y_{\mu''}$ of length μ'' bits, let $S_k \subseteq [\mu'']$ denote the set of positions in k that are set to 1. Let $m = m_1, \ldots, m_\ell$ be a bit string of length ℓ bits (where ℓ is polynomial in μ''). let $\chi_{S_k}(y) := \bigoplus_{i \in S_k} y_i$.

On input $m \in \{0,1\}^\ell$ and k as above, $\mathsf{Hide}(k, m)$ chooses random strings $y_1^0, \ldots, y_{\mu''}^0$, $y_1, \ldots, y_\ell \leftarrow U_{\mu''}$ and outputs $([y_i^0, \chi_{S_k}(y_i^0)]_{i \in [\mu'']}, [(y_i, \chi_{S_k}(y_i) \oplus m_i)]_{i \in [\ell]})$.

On input $([y_i^0, \chi_{S_k}(y_i^0)]_{i \in [\mu'']}, [(y_i, \chi_{S_k}(y_i) \oplus m_i)]_{i \in [\ell]})$, Rec uses the first μ'' examples to learn χ_{S_k} and then returns $[m^i]_{i \in [\ell]} := \chi_{S_k}(y_i) \oplus m_i'$.

5.5.4 Non-Interactive Simulatable Proof System (parameter $n'' \in o(n)$)

In the following construction, inputs and proofs have λ parallel components, corresponding to λ parallel invocations of the MPC-in-the-head paradigm. To simplify the exposition, we assume that the bounded, streaming computations read in λ symbols in parallel from each of the λ parallel components and output λ symbols in *parallel for each of the λ parallel components. Note that this increases the required storage by a factor of λ, but since we set $\lambda \ll n''$, the overall storage bound remains below n''.*

We begin by introducing a simplified proof system and proving its soundness. We then present the actual proof system used in our construction. Looking ahead, proving that the ***Simulation Soundness*** *property required by Theorem 5.3.1 holds, will reduce to the soundness of the* simplified *proof system.*

Simplified Proof System Π' *Let λ' be security parameter and ℓ is a constant (e.g. $\ell = 5$).*

P*: On input statement s, encoding $[s_u^1, \ldots, s_u^{\lambda'}]_{u \in [\ell]}$ and witness w:*

1. ***Check that for** $q \in [\lambda']$, $s_1^q \oplus \cdots \oplus s_\ell^q = s$**:*** *Compute streaming hash $h^* := H^h(s)$ and λ' streaming hashes in parallel, $h_q := [H^h(s_1^q \oplus \cdots \oplus s_\ell^q)]_{q \in \lambda'}$, where H is Merkle Damgard and $h \leftarrow \mathcal{H}$, where $\mathcal{H}$ is a universal family of hash function. Check that for all $q \in [\lambda']$, $h_q = h^*$. If not, output $\bot$.*

2. ***Run MPC-in-the-head:*** *For $q \in [\lambda]$, secret share w into ℓ additive shares $(w_1^q, \ldots, w_\ell^q)$ and run*

   $\mathsf{MPC}(P_1(s_1^q, w_1^q) \ldots, P_\ell(s_\ell^q, w_\ell^q)$*, producing views $[\mathsf{View}_u^q]_{q \in [\lambda'], u \in [\ell]}$ (here, each view is a tableau of the parties' computation, as described in the construction of circuit SAT proof system for streaming verifiers in Section 5.2.2).*

   Note that of the input wires to the views, some will be public (corresponding to *the shares of s) and some will be private (corresponding to the shares of w).*

3. ***Encrypt the Views.*** *For $q \in [\lambda'], u \in [\ell]$, choose k_u^q uniformly at random from $\{0,1\}^{\mu''}$. Compute $S_u^q \leftarrow \mathsf{Hide}(k_u^q, \mathsf{View}_u^q)$, where Hide is run with parameter n''. Output proof $T = ([\hat{k}_u^q, S_u^q]_{q\in[\lambda'],u\in[\ell]})$.*

$\underline{\mathsf{V}}$*: On input statement s, encoding $[s_u^1, \dots, s_u^{\lambda'}]_{u\in[\ell]}$ and proof T, parse $T = ([\hat{k}_u^q, S_u^q]_{q\in[\lambda'],u\in[\ell]})$.*

1. ***Generate randomness.*** *Choose randomness $r_1, \dots, r_{\lambda'}$ and hash function $h \leftarrow \mathcal{H}$. For each $q \in [\lambda]$, choose a subset $\mathcal{S}_q \subseteq [\ell]$ using random coins r_q.*

2. ***Check that for** $q \in [\lambda]$**,** $s_1^q \oplus \cdots \oplus s_\ell^q = s$**:*** *Repeat the same steps as P to check that for $q \in [\lambda]$, $s_1^q \oplus \cdots \oplus s_\ell^q = s$ if not, output $\perp$.*

3. ***Prepare hashes of input for later equality checks.*** This is done in parallel to the previous item. *For $q \in [\lambda']$, $u \in \mathcal{S}_q$, compute $h_{q,u} = H^h(s_u^q)$.*

4. ***Open selected views.*** *For $q \in [\lambda]$, $u \in \mathcal{S}_q^1$, recover $k_u^q = \mathsf{Decap}(\hat{k}_u^q)$, recover View_u^q, where $\mathsf{View}_u^q := \mathsf{Rec}(S_u^q)$ (where Rec is run with parameter n'') and corresponding inputs $\widetilde{s}_u^q, \widetilde{w}_u^q$.*

5. ***Check consistency of views.*** This is done in parallel to the previous item. *(1) Check that the opened views are internally consistent (using the verifier described in the construction of circuit SAT proof system for streaming verifiers in Section 5.2.2.2). (2) Check that the opened views are consistent with each other (i.e. same transcript) using similar hashing techniques as above. (3) Check that $h_{q,u} = H^h(\widetilde{s}_u^q)$.*

6. ***Output.*** *If all checks succeed, output* 1*. Otherwise, output* 0*.*

Claim 5.5.3. *Soundness of proof system follows from perfect correctness of the MPC and security of the universal hash function family $\mathcal{H}$.*

The Actual Proof System Π *As above, in the following construction, inputs and proofs have λ parallel components, corresponding to λ parallel invocations of the MPC-in-the-head paradigm. To simplify the exposition, we assume that the bounded, streaming computations read in λ symbols in parallel from each of the λ parallel components and output λ symbols in parallel for each of the λ parallel components. Note that this increases the required storage by a factor of λ, but since we set $\lambda \ll n''$, the overall storage bound remains below n''.*

Let λ be security parameter and ℓ is a constant.

<u>P</u>*: On input statement $s := s_1, \ldots, s_t$, encoding $[s^q_1, \ldots, s^q_\ell]_{q\in[\lambda]}$ and witness w:*

1. ***Check that for** $q \in [\lambda]$**,** $s^q_1 \oplus \cdots \oplus s^q_\ell = s$**:** Compute streaming hash $h^* := H^h(s)$ (with block length λ) and λ streaming hashes (all with block length λ) in parallel, $h_q := [H^h(s^q_1 \oplus \cdots \oplus s^q_\ell)]_{q\in\lambda}$, where H is Merkle Damgard and $h \leftarrow \mathcal{H}$, where $\mathcal{H}$ is a universal family of hash function. Check that for all $q \in [\lambda]$, $h_q = h^*$. If not, output $\perp$.*

2. ***Run MPC-in-the-head:** For $q \in [\lambda]$, secret share w into ℓ additive shares $(w^q_1, \ldots, w^q_\ell)$ and run*

   $\mathsf{MPC}(P_1(s^q_1, w^q_1) \ldots, P_\ell(s^q_\ell, w^q_\ell))$*, producing views* $[\mathsf{View}^q_u]_{q\in[\lambda],u\in[\ell]}$*.*

Note that of the input wires to the views, some will be public (corresponding to the shares of s) and some will be private (corresponding to the shares of w).

3. ***Select the Slots.*** *For each position q, u there are $\ell \cdot 2t$ slots $[S^{z,p}_{q,u}]_{z\in[\ell],p\in[2t]}$, where $t = |s|$. Let $s^q[z,p]$ denote the p-th bit position of the string s^q_z. Let $\mathcal{S}'_{q,z}$ be the set of positions in the string $[s^q[z,p]||\overline{s}^q[z,p]]_{p\in[t]}$ that are set to 1. Note that $|\mathcal{S}'_{q,z}| = t$.*

4. ***Encrypt the Views.*** *For $q \in [\lambda], u \in [\ell], z \in [\ell], p \in [2t]$, choose $k^{z,p}_{q,u}$ uniformly at random from $\{0,1\}^{\mu''}$. For $q \in [\lambda], u \in [\ell], z \in [\ell]$ and $p \in \mathcal{S}'_{q,z}$, compute $S^{z,p}_{q,u} \leftarrow \mathsf{Hide}(k^{z,p}_{q,u}, \mathsf{View}^q_u)$, where Hide is run with parameter n''. For $q \in [\lambda], u \in [\ell], z \in [\ell]$ and $p \notin \mathcal{S}'_{q,z}$, set $S^{z,p}_{q,u} \leftarrow \mathsf{Hide}(k^{z,p}_{q,u}, \vec{0})$. Output proof $T = ([\hat{k}^{z,p}_{q,u}, S^{z,p}_{q,u}]_{q\in[\lambda],u\in[\ell],z\in[\ell],p\in[2t]})$.*

$\underline{\mathsf{V}}$: *On input statement $s := s_1, \dots, s_t$, encoding $[s^q_1, \dots, s^q_\ell]_{q\in[\lambda]}$, and proof T, parse $T = ([\hat{k}^p_{q,u}, S^p_{q,u}]_{q\in[\lambda],u\in[\ell],p\in[2t]})$.*

1. ***Generate Randomness.*** *Choose randomness $(r^1_1, r^2_1) \dots, (r^1_\lambda, r^2_\lambda)$ and hash function $h \leftarrow \mathcal{H}$. Choose subsets $\mathcal{S}^1_q, \mathcal{S}^2_q \subseteq [\ell]$, each of size 2, using random coins (r^1_q, r^2_q).*

2. ***Check that for $q \in [\lambda]$, $s^q_1 \oplus \cdots \oplus s^q_\ell = s$:*** *Repeat the same steps as P to check that for $q \in [\lambda]$, $s^q_1 \oplus \cdots \oplus s^q_\ell = s$ if not, output $\perp$.*

3. ***Prepare hashes of input for later equality checks.*** This is done in parallel to the previous item. *Do the following in parallel: (1) For $q \in [\lambda]$, $u \in \mathcal{S}^1_q$, compute $h^1_{q,u} = H^h(s^q_u)$ in a streaming fashion, using block size λ and*

space $O(\lambda^2)$. (2) For $q \in [\ell]$, $u \in \mathcal{S}^2_q$, compute $h^2_{q,u} = H^h(s^q_u)$ in a streaming fashion, using block size λ and space $O(\lambda^2)$.

4. ***Open selected views and check consistency across slots.*** *For $q \in [\lambda]$, $u \in \mathcal{S}^1_q$, do the following: (1) For each $z \in [\mathcal{S}^2_q]$, $p \in [2t]$, recover $k^{z,p}_{q,u} = \mathsf{Decap}(\hat{k}^p_{q,u})$. (2) For each $z \in [\mathcal{S}^2_q]$, recover $\mathsf{View}^{z,p}_{q,u}$, where $\mathsf{View}^{z,p}_{q,u} := \mathsf{Rec}(S^{z,p}_{q,u})$, and Rec is run with parameter n''. Let $[\mathsf{View}^{z,p}_{q,u}]_{p \in \mathcal{S}'^z_{q,u}}$ be the views (out of $[2t]$) that do not decrypt to $\vec{0}$ (i.e. $\mathcal{S}'^z_{q,u}$ is the set of slots that are filled). Let $s'^z_{q,u}$ denote the vector corresponding to $\mathcal{S}'^z_{q,u}$. (3) Use hashing as above to check that for each q, r all the recovered views $\mathsf{View}^{z,p}_{q,u}$ are identical. (4) Let View^q_u denote the contents of these identical views and let $(\widetilde{s}^q_u, \widetilde{w}^q_u)$ be the corresponding inputs.*

5. ***Check consistency of views.*** This is done in parallel to the previous item. *(1) For $q \in [\lambda]$, $u \in \mathcal{S}^1_q$ check that the view View^q_u is internally consistent (using the verifier described in the construction of circuit SAT proof system for streaming verifiers in Section 5.2.2.2). (2) For $q \in [\lambda]$, Check that the views $[\mathsf{View}^q_u]_{u \in \mathcal{S}'_q}$ are consistent with each other (i.e. same transcript) using similar hashing techniques as above. (3) For each $u \in \mathcal{S}^1_q$, check that $h^1_{q,u} = H^h(\widetilde{s}^q_u)$. (4) For each $u \in \mathcal{S}^2_q$, check that $h^2_{q,u} = H^h(\widetilde{s}^q_u)$.*

6. ***Output.*** *If all checks succeed, output 1. Otherwise, output 0.*

<u>Sim</u>*: On input statement $s := s_1, \ldots, s_t$, encoding $[s^q_1, \ldots, s^q_\ell]_{q \in [\lambda]}, \ldots, [s^q_1, \ldots, s^q_\ell]_{q \in [\lambda]}$:*

1. ***Check that for $q \in [\lambda]$, $s^q_1 \oplus \cdots \oplus s^q_\ell = s$:*** *Compute streaming hash $h^* :=$*

$H^h(s)$ and λ streaming hashes in parallel, $h_q := [H^h(s_1^q \oplus \cdots \oplus s_\ell^q)]_{q \in \lambda}$, where H is Merkle Damgard and $h \leftarrow \mathcal{H}$, where $\mathcal{H}$ is a universal family of hash function. Check that for all $q \in [\lambda]$, $h_q = h^*$. If not, output $\perp$.

2. ***Run MPC-in-the-head Simulation:*** *For $q \in [\lambda]$, choose subset $\mathcal{S}_q^1 \subseteq [\ell]$ using random coins r_q^1. run $\mathsf{Sim}^{\mathsf{MPC}}$ to produce the views of parties P_u, $u \in \mathcal{S}_q^1$ (note that each of these parties has public input s_u^q) producing views $[\mathsf{View}_u^q]_{q \in [\lambda], u \in \mathcal{S}_q}$.*

3. ***Select the Slots.*** *For each position q, u there are $\ell \cdot 2t$ slots $[q, u, z, p]_{z \in [\ell], p \in [2t]}$, where $t = |s|$. Let $s^q[z, p]$ denote the p-th bit position of the string s_z^q. Let $\mathcal{S}'_{q,z}$ be the set of positions in the string $[s^q[z, i] || \overline{s}^q[z, i]]_{i \in [t]}$ that are set to 1. Note that $|\mathcal{S}'_{q,z}| = t$.*

4. ***Encrypt the Views.*** *Choose subset $\mathcal{S}_q^2 \subseteq [\ell]$ using random coins r_q^2. For $q \in [\lambda], u \in [\ell], z \in [\ell], p \in [2t]$, choose $k_{q,u}^{z,p}$ uniformly at random from $\{0,1\}^{\mu''}$. For $q \in [\lambda], u \in [\mathcal{S}_q^1], z \in [\mathcal{S}_q^2]$ and $p \in \mathcal{S}'_{q,z}$, compute $S_{q,u}^{z,p} \leftarrow \mathsf{Hide}(k_{q,u}^{z,p}, \mathsf{View}_u^q)$, where Hide is run with parameter n''. For $q \in [\lambda]$ and u, z, p such that $u \notin [\mathcal{S}_q^1]$ OR $z \notin [\mathcal{S}_q^2]$ OR $p \notin \mathcal{S}'_{q,z}$, set $S_{q,u}^{z,p} \leftarrow \mathsf{Hide}(k_{q,u}^{z,p}, \vec{0})$. Output proof $\pi = ([\hat{k}_{q,u}^{z,p}, S_{q,u}^{z,p}]_{q \in [\lambda], u \in [\ell], z \in [\ell], p \in [2t]})$.*

Remark 5.5.1. *Note that if the simulated proof and encoding $[s_1^q, \ldots, s_\ell^q]_{q \in [\lambda]}$ are generated at the same time, then we can* first *produce the simulated proof π* independently *of s. This can be done because the simulated proof depends only on $[s_u^q]_{q \in [\lambda], u \in [\mathcal{S}_q^1 \cup \mathcal{S}_q^2]}$, which can be chosen uniformly at random (since our parameter settings ensure that $|\mathcal{S}_q^1| + |\mathcal{S}_q^2| = \ell - 1$). Given the simulated proof π and the choice* *of $[s^q_y]_{q\in[\lambda],y\in[\mathcal{S}^1_q\cup\mathcal{S}^2_q]}$, we can then output the entire encoding $[s^q_1[1],\dots,s^q_\ell[1]]_{q\in[\lambda]},\dots,[s^q_1[t'],\dots,s^q_\ell[t']]_{q\in[\lambda]}$ and proof π in a* streaming fashion, *given input s in a streaming fashion, requiring only $O(\lambda^2)$ memory. This is done by hardwiring $[s^q_y]_{q\in[\lambda],y\in[\mathcal{S}^1_q\cup\mathcal{S}^2_q]}$, and, in a block-by-block streaming fashion (with block length λ), outputting, in parallel, the i-th block of each share for each $q\in[\lambda]$, along with the missing share: $[s\oplus\left(\bigoplus_{y\in[\mathcal{S}^1_q\cup\mathcal{S}^2_q]} s^q_y\right)]_{q\in\lambda}$.*

Remark 5.5.2. *Note that for two statements $s_1\neq s_2$ and their proofs π_{s_1}, π_{s_2}, for each q, there exists a pair (z^*_q,p^*_q) such that for each $u\in[\ell]$, slot $[q,u,z^*_q,p^*_q]$ contains encryptions of $\vec{0}$ in π_{s_1} and encryptions of View^q_u in π_{s_2}. Moreover, for any two statements $s_1\neq s_2$ and every $q\in[\lambda]$, the probability over choice of r^2 that $z^*_q\in[\mathcal{S}^2_q]$, (which means that slots $[q,u,z^*,p^*]_{u\in\mathcal{S}^1_q}$ will be checked by V) is at least $1/\ell$.*

Claim 5.5.4. *Let $\mathcal{A}$ be an* unbounded *adversary that takes as input random variable $S^1_{\mathsf{tamp}}||S^2_{\mathsf{tamp}}$. Let S^1,S^2 denote the random variables corresponding to the initial contents of $\mathcal{A}$'s input (before tampering).*

Let f be a streaming tampering function with memory $o(n'')$ that reads in random variable $S^1||I||S^2$ (chunk-by-chunk), where $I=\mathsf{Hide}(k,m)$ is an encoding of m with random key k and parameter n'', and (in a streaming fashion) outputs the random variable $S^1_{\mathsf{tamp}}||I_{\mathsf{tamp}}||S^2_{\mathsf{tamp}}$ (chunk-by-chunk). For $i\in[3]$, let $f(S^1||I||S^2)[i]$ denote the i-th chunk outputted by f.

Then for any m_0,m_1, when I encodes m_0 vs. I encodes m_1 the resulting output distributions of $\mathcal{A}(f(S^1||I||S^2)[1],f(S^1||I||S^2)[3])$ are statistically close.

Proof. If the claim is false, then there exists a distinguisher D. Using D, $\mathcal{A}$, f, we can now construct a streaming branching program with space $o(n'')$ that distinguishes whether I encodes 0 or 1. We do so in the following way:

1. Fix the random variables $S^1 = s^1$ and $S^2 = s^2$
2. Construct a branching program $BP_{s^1,s^2,D,\mathcal{A},f}$ that hardcodes s^1, s^2 and emulates $f(s^1||I||s^2)$. Note the following about the emulation:
   - $S^1_{\mathsf{tamp}} = s^1_{\mathsf{tamp}} = f(s^1||I||s^2)[1]$ and the entire inner state of f up to the moment right before it starts reading I can be hardcoded into the transition function for BP.
   - from this point on, we can emulate $f(s^1||I||s^2)$ using space $o(n'')$ until the moment that f finishes reading I.
   - from this point on, we can determine the output of $\mathcal{A}(s^1_{\mathsf{tamp}}, f(S^1||I||S^2)[3])$ without requiring any more memory. To do this, we use the fact that $s^1, s^1_{\mathsf{tamp}}, s^2$ are hardcoded and simply precompute the output of $\mathcal{A}(s^1_{\mathsf{tamp}}, f(s^1||I||s^2)[3])$ for each of the possible $2^{o(n'')}$ internal states of f (by using the internal state of f at the moment that f finishes reading I to compute $S^2_{\mathsf{tamp}} = s^2_{\mathsf{tamp}}$ and then running $\mathcal{A}$ on $(s^1_{\mathsf{tamp}}, s^2_{\mathsf{tamp}})$) and then output whatever D outputs. This implies that given the internal state of f at the moment f finishes reading I, we can immediately transition to the output level of the branching program, without requiring additional state.

3. Note that BP succeeds with the same probability as D.

$\square$

Theorem 5.5.1. $\Pi = (\mathsf{E}, \mathsf{D})$ *(presented in Figure 5.1) is a one-bit, unconditional non-malleable code against streaming adversaries with space* $o(n'')$*, if the underlying components are instantiated in the following way:*

- $\mathcal{E} := (\mathsf{Encrypt}, \mathsf{Decrypt})$ *is the encryption scheme described in Section 5.5.2 (with parameter* $n' := n'(n)$*).*
- $\Pi^{\mathsf{NI}} := (\mathsf{P}^{\mathsf{NI}}, \mathsf{V}^{\mathsf{NI}}, \mathsf{Sim}^{\mathsf{NI}})$ *the simulatable proof system with streaming verifier described in Section 5.5.4 with parameter* $n'' := n''(n)$*.*
- *For* $b \in \{0, 1\}$*,* D_b *is the distribution described in Section 5.5.1 (with paramter* n*).*

Note that no CRS or computational assumptions are needed for this result. Indeed we can assume that the adversary $\mathcal{A}$ *outputting tampering function* f *is computationally unbounded. To maintain consistency, we continue to use the variables* crs*,* SK*,* $\overrightarrow{\tau}_{\mathsf{sim}}$ *but we simply assume that all of them are set to* $\perp$*.*

Proof. To prove the theorem, we need to show that the necessary properties from Theorem 5.3.1 hold. We next go through these one by one.

- **Simulation of proofs.**

  1. $\Pr[g(\mathsf{CW}_0, f(\mathsf{CW}_0), r_0) = 1] \approx \Pr[g(\mathsf{CW}_1, f(\mathsf{CW}_1), r_1), = 1]$,

2. $\Psi(g, \mathsf{crs}, \mathsf{CW}_0, f(\mathsf{CW}_0), r_0, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_0); r_0)) \approx$

   $\Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1)),$

where $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_1, r_2 are sampled uniformly at random, $\mathsf{CW}_0 \leftarrow \mathsf{E}(\mathsf{crs}, \vec{b}_0)$ and $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r_1, \vec{b}_0)$.

To prove this, we must switch from all real proofs (as outputted by E) to all simulated proofs (as outputted by E). Looking closer at the construction from Section 5.5.4, to switch from a real to a simulated proof, we must go through a sequence of hybrids starting from honestly generated proofs from E and ending with simulated proofs from E_1. In hybrid $H_{[q,u,p]}$ we switch to using encoding algorithm $\mathsf{E}^{[q,u,p]}$, which works the same way as the encoding in the previous hybrid, except when generating the proofs, if $r \notin \mathcal{S}_q$, it sets random variable $S^p_{q,u}$ to $S^p_{q,u} \leftarrow \mathsf{Hide}(k^p_{q,u}, \vec{0})$. Note that for $u \in \mathcal{S}_q$, $H_{[q,u,p]}$ is identical to the previous hybrid. Let $\mathsf{CW}^{[q,u,p]}$ denote the random variable representing the codeword in each hybrid distribution. We use Claim 5.5.4 to show that for every fixed random string r and $u \notin \mathcal{S}_q$, the output of g in consecutive hybrids is indistinguishable and the output of Ψ in consecuitve hybrids is indistinguishable. To see this, note that we set $\mathcal{A}$ from Claim 5.5.4 to be equal to D, $f = f$, S^1 denotes the codeword up to the $[q, u, p]$ position, S^2 denotes the codeword after the $[q, u, p]$ position, and $I := S^p_{q,u}$. The key is that V^{NI} (which checks the proofs during computation of g and Ψ) with random coins $r := r_1, \ldots r_q$ will not check slot $[q, u, p]$ when determining its output and so the conditions of Claim 5.5.4 are satisfied.

- **Simulation of Encryption.**

  1. $\Pr[g(\mathsf{CW}_1, f(\mathsf{CW}_1), r_1) = 1] \approx \Pr[g(\mathsf{CW}_2, f(\mathsf{CW}_2), r_2), = 1]$,

  2. $\Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1)) \approx$

     $\Psi(g, \mathsf{crs}, \mathsf{CW}_2, f(\mathsf{CW}_2), r_2, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_2); r_2))$,

  where $(\mathsf{crs}, \textsc{sk}, \vec{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_1, r_2 are sampled uniformly at random,

  $\mathsf{CW}_1 \leftarrow \mathsf{E}_1(\mathsf{crs}, \vec{\tau}_{\mathsf{sim}}, r_1, \vec{b}_0)$ and $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \vec{\tau}_{\mathsf{sim}}, r_2, \vec{b}_0)$.

  To see this, we will show that $g(\mathsf{CW}_1, f(\mathsf{CW}_1), r_1)$, $\Psi(g, \mathsf{crs}, \mathsf{CW}_1, f(\mathsf{CW}_1), r_1, \mathsf{D}(\mathsf{crs}, f(\mathsf{CW}_1); r_1))$ can be computed in a streaming fashion with memory $o(n')$, while distinguishing encryptions of k_i from encryptions of k'_i in a streaming fashion requires memory $\Omega(n')$ (see Claim 5.5.2). We will show that each of $\mathsf{E}_1/\mathsf{E}_2, f, \mathsf{D}, g$ can be computed in a streaming fashion with memory $o(n')$. This implies that their (parallel) composition can also be computed in a streaming fashion with memory $o(n')$.

  To see that this is true for $\mathsf{E}_1/\mathsf{E}_2$, we use the observation from Remark 5.5.1. It is true for f by definition of the tampering class $\mathcal{F}$. D consists of (1) determining b such that $\vec{x}$ is in the support of D_b and (2) running the verifier for Π. Note that (1) can be done in a streaming fashion using $\Theta(n)$ bits of memory. Since we choose $n' = \omega(n)$, the required memory is $o(n')$. (2) can be done in a streaming fashion with space $o(n')$, since the only memory intensive part of the verification is running Rec. Similar to the above, we set parameters of $\mathsf{Hide}/\mathsf{Rec}$ such that this can be done using $\Theta(n'')$ bits of memory, where $n'' = o(n)$. Finally, g consists of a bit-wise comparison of two strings obtained in a streaming fashion and running the verifier for Π, both of which can be done in a streaming fashion with memory $o(n')$. Thus, we have shown that each of each of $\mathsf{E}_1/\mathsf{E}_2, f, \mathsf{D}, g$ can be computed in a streaming fashion with memory $o(n')$.

- **Simulation Soundness.**

  $\Pr_r[\mathsf{D}(f(\mathsf{CW}_2); r_2) \neq \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \wedge$

  $g(\mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 0] \leq \mathsf{negl}(n)$,

  where $(\mathsf{crs}, \textsc{sk}, \overrightarrow{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_2 is sampled uniformly at random and $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \overrightarrow{\tau}_{\mathsf{sim}}, r, 0)$.

  We give a reduction from the above property with Π instantiated with security parameter λ to the soundness of Π' with security parameter $\lambda' = \lambda/2\ell$.

  First, if $g(\mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 0$ then it must be the case that either the verification of the proofs rejects (in both D and D', since they are the same) or $s_1 \neq s_2$, where s_1 is the statement for the proofs in CW and s_2 is the statement for the proofs in $f(\mathsf{CW})$, and $s_2 \notin \mathcal{L}$. Therefore, by Remark 5.5.2, for each q, there exists a pair (z_q^*, p_q^*) such that for each $u \in [\ell]$, slot $[q, u, z_q^*, p_q^*]$ contains encryptions of $\vec{0}$ in π_{s_1} and (is supposed to contain) encryptions of View_u^q in π_{s_2}.

  We now consider the distribution over slots $([q, u, z_q^*, p_q^*]_{q \in [\lambda], u \in [\ell]})$. Note that by Claim 5.5.4 the distribution over these slots only is statistically close in the case that f gets as input a codeword with a simulated proof, versus a proof where all $S^{z,p}_{q,u}$ encrypt $\vec{0}$.

Therefore, our reduction R will construct a simulated proof $\pi' = ([\hat{k}^{z,p}_{q,u}, S^{z,p}_{q,u}]_{q \in [\lambda], u \in [\ell], z \in [\ell], p \in [2t]})$, for s_1 where all $S^{z,p}_{q,u}$ encrypt $\vec{0}$. Note that this simulated proof π' has *no dependence* on $(r^1_1, r^2_1) \ldots, (r^1_\lambda, r^2_\lambda)$, since all slots encrypt $\vec{0}$ so there is no information at all in the proof. R will then extract a proof $\pi'' = ([\hat{k}^{z,p}_{q,u}, S^{z,p}_{q,u}]_{q \in [\lambda], r \in [\ell], z \in [\ell], p \in [2t]})$ for some statement $s_2 \neq s_1$ from the tampered codeword. It will now choose random coins r^2_q and sets $\mathcal{S}^2_q \subseteq [\ell]$, for $q \in [\lambda]$ (using random coins r^2_q). Using Remark 5.5.2, we know that the probability over choice of random coins r^2_q that $z^*_q \in \mathcal{S}^2_q$ is at least $1/\ell$. Therefore, with all but negligible probability, there is a set $\mathcal{Q} \subseteq [\lambda]$ of size at least $1/2\ell \cdot \lambda$ such that $z^*_q \in \mathcal{S}^2_q$ for all $q \in \mathcal{Q}$. Moreover, note that if w.h.p. over choice of $r^1_1, \ldots, r^2_\lambda$, all checks for $[\hat{k}^{z,p}_{q,u}, S^{z,p}_{q,u}]_{q \in Q, u \in [\ell], z \in [\ell], p \in [2t]}$ pass then it must be the case that $[\hat{k}^{z^*_q, p^*_q}_{q,u}, S^{z^*_q, p^*_q}_{q,u}]_{q \in [Q], u \in [\ell]}$ is a proof for statement s_2 for proof system Π' for which $\mathsf{V}^{\Pi'}$ accepts w.h.p. But this breaks the soundness of proof system Π' with security parameter $\lambda' = |Q| \geq \lambda/2\ell$.

- **Hardness of D_b relative to Alternate Decoding.**

1. $\Pr[g(\mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1] \approx \Pr[g(\mathsf{CW}_3, f(\mathsf{CW}_3), r_3) = 1]$,

2. $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \approx \mathsf{D}'(\mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$,

where $(\mathsf{crs}, \textsc{sk}, \vec{\tau}_{\mathsf{sim}}) \leftarrow \mathsf{CRSGen}(1^n)$, $f \leftarrow \mathcal{A}(\mathsf{crs})$, r_2, r_3 are sampled uniformly at random, $\mathsf{CW}_2 \leftarrow \mathsf{E}_2(\mathsf{crs}, \vec{\tau}_{\mathsf{sim}}, r_2, 0)$ and $\mathsf{CW}_3 \leftarrow \mathsf{E}_2(\mathsf{crs}, \vec{\tau}_{\mathsf{sim}}, r_3, 1)$.

Let $\vec{X}$ denote a random variable where $\vec{X} \leftarrow D_0$ with probability $1/2$ and

$\vec{X} \leftarrow D_1$ with probability $1/2$ and let random variable CW denote the output of E_2 when $\vec{X}$ replaces $\vec{x}$.

To show (1), assume $\Pr[g(\mathsf{CW}_2, f(\mathsf{CW}_2), r_2) = 1]$ and $\Pr[g(\mathsf{CW}_3, f(\mathsf{CW}_3), r_3) = 1]$ differ by a non-negligible amount. This implies that a circuit that takes as input $\vec{X}$, hardwires all other random variables, and outputs 1 in the case that $g(\mathsf{CW}, f(\mathsf{CW}), r) = 1$ and 0 otherwise, implying that it has non-negligible correlation to the function that outputs b such that $\vec{X}$ is in the support of D_b. We will show that the above can be computed by a streaming adversary with storage $o(n)$ and input $\vec{X}$, thus contradicting Claim 5.5.1. Indeed, this follows since the output of E_2 can be computed in a streaming fashion using a similar trick to the AC^0 case, f can be computed by streaming adversaries with storage $o(n)$ by definition of tampering class $\mathcal{F}$, and verification for Π can also be computed in a streaming fashion with memory $\Theta(n'')$, where $n'' = o(n)$. So the composition of the three can also be computed by streaming adversaries with storage $o(n)$.

To show (2), assume $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2)$ and $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3)$ have non-negligible statistical distance. This implies that a circuit that takes as input $\vec{X}$, hardwires all other random variables, and outputs $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW})), f(\mathsf{CW}); r_2)$ has non-negligible correlation to the function that outputs b such that $\vec{X}$ is in the support of D_b. We will show that $\mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\textsc{sk}, f(\mathsf{CW})), f(\mathsf{CW}); r_2)$ can be computed by a streaming adversary with storage $o(n)$ and input $\vec{X}$, thus contradicting

Claim 5.5.1. To show this, note first that the output k_{n+1} of $\mathsf{Ext}(\textsc{sk}, f(\mathsf{CW}))$ can be given as non-uniform advice since it does not depend on $\vec{X}$. This is the case because the key is extracted by looking at the first part of the tampered codeword, which is independent of $\vec{X}$. Since the tampering function is streaming as well, it means that the output of the tampering function was determined independently of $\vec{X}$.

Now, we must show that E_2, f, and $\mathsf{D}'(\mathsf{crs}, k_{n+1}, \cdot; r_2)$ can all be computed in a streaming fashion. We have already argued that E_2, f can be computed in a streaming fashion. Note that $\mathsf{D}'(\mathsf{crs}, k_{n+1}, \cdot; r_2)$ simply decrypts (by xor'ing k_{n+1} with c) and checks all proofs using the verifier of Π, which can be done in a streaming fashion, with space $\Theta(n'') = o(n)$.

□

5.5.5 Multi-Bit NMC Against Streaming Adversaries

The result from the previous section extends trivially for any number m of bits. Moreover, when we increase the number of bits m, all other parameters (n, n', n'') can remain the same and do not need to be increased as in our previous multi-bit constructions. To see this, note that the only additional property that needs to be proved in the multi-bit case is that for every Boolean function, represented by a circuit F over m variables,

$F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_2)), f(\mathsf{CW}_2); r_2) \approx$

$F \circ \mathsf{D}'(\mathsf{crs}, \mathsf{Ext}(\mathsf{crs}, \textsc{sk}, f(\mathsf{CW}_3)), f(\mathsf{CW}_3); r_3).$

But in the bounded, streaming model, F as above can be computed without requiring any additional memory beyond what is required in the one-bit case. To see this, recall that the streaming adversary can receive the decryptions *of the m ciphertexts in the tampered codeword as non-uniform advice, since tampering on this part of the codeword does not depend on the values of $[\vec{x}^i]_{i \in [m]}$. Thus, the streaming adversary needs only to check the $m \cdot n + 1$ proofs, in a streaming fashion, in order to determine the output of D': If all proofs verify correctly, the output of D' will consist of the hardcoded, "candidate" bits; otherwise, D' will output $\vec{0}$. Thus, the streaming adversary can compute the output of D' using the same amount of space as in the one-bit case. Now, F needs to be applied to the output of D'. But note that computing F does not require any additional space. Indeed, given the state of the streaming adversary at the moment the output of D' is determined, we can simply hardcode the output of F in the transition function. Thus, no additional memory is required.*

5.6 Figure to explain MPC in head from section 5.5.4

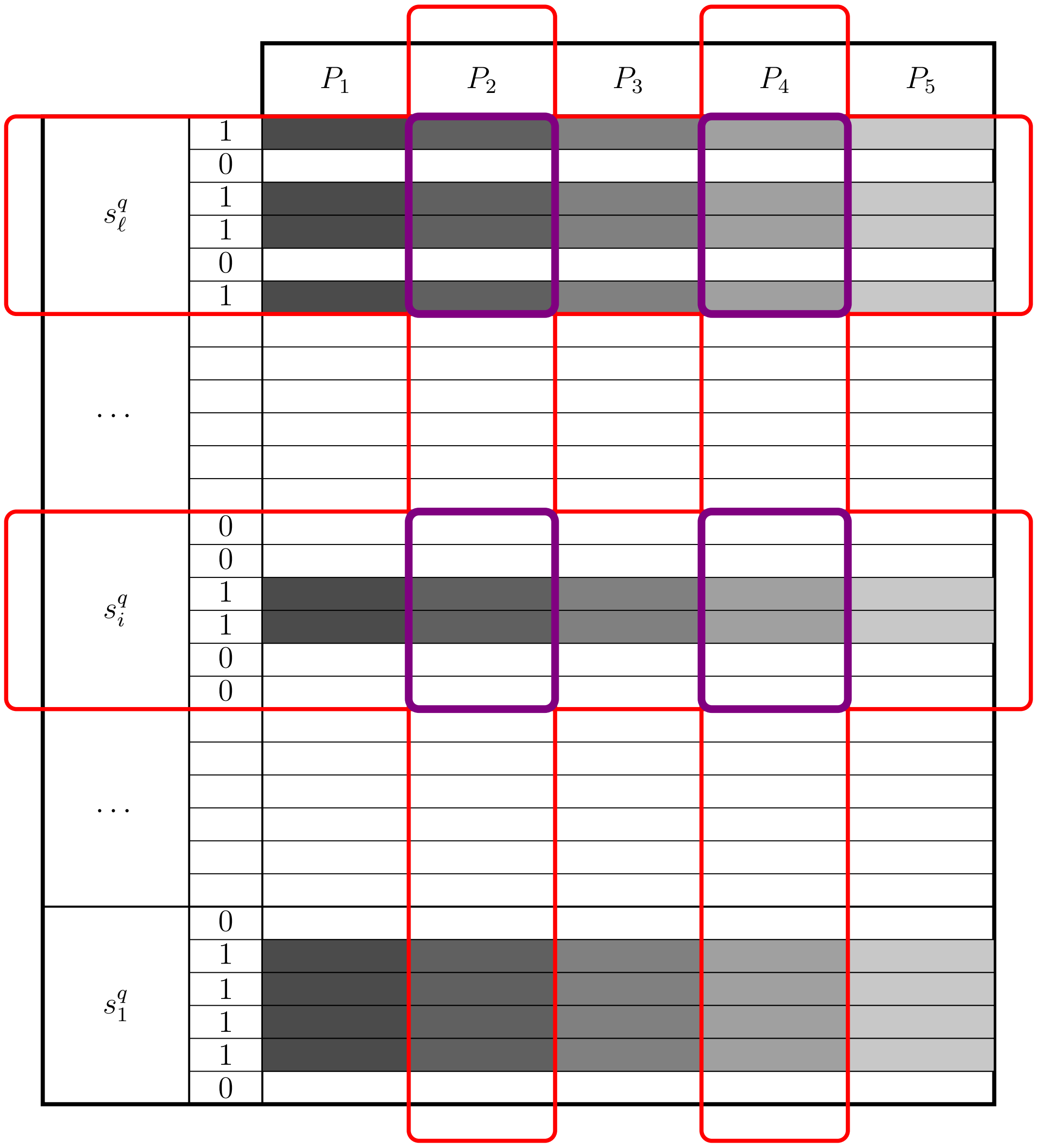

Figure 5.13: **A pictorial representation of the Prover's output in the NI Simulatable Proof System Π.** Let $\ell = 5$ be the number of parties, and λ be the security parameter. In the q-th iteration, each party P_i for $i \in [\ell]$ has inputs (w_i^q, s_i^q). We encode each s_i^q as $s_i^q || \overline{s}_i^q$, where $\overline{s}_i^q$ is the bit-wise complement of s_i^q. For example 001 is encoded as 001100. For each bit of the encoding of s_i^q, if the bit is 1 then each party P_i places a weak encryption of its view, view_i, in the corresponding slot (represented by filled-in rectangles of various shades of gray in the figure). Otherwise, if the bit is 0 then each party places a weak encryption of all 0's in the corresponding slot, (represented by blank rectangles in the figure). During verification, the verifier checks in the first step that input $s = s_1^q \oplus s_2^q \oplus \ldots \oplus s_\ell^q$. To check the consistency of the views, the verifier selects 2 columns (P_2 and P_4) and 2 rows (s_i^q and s_ℓ^q) at random and does the following: (1) checks that s_i^q and s_ℓ^q are consistent with the values read in the first step (2) runs Rec on the weakly encrypted views and checks that the resulting views are consistent with each other and internally.

Chapter 6: Tight Upper and Lower Bounds for Leakage-Resilient, Locally Decodable and Updatable Non-Malleable Codes

6.1 Introduction

As discussed in chapter 1 standard non-malleable codes are useful for protecting small amounts of secret data stored on a device (such as a cryptographic secret key) but unfortunately are not suitable in settings where, say, an entire database must be protected. In a recent result, [54] proposed a new notion called locally decodable and updatable non-malleable codes, which informally speaking, provides the security guarantees of a non-malleable code while also allowing for efficient random access. In more detail, we consider a message $m = m_1, \ldots, m_n$ consisting of n blocks, and an encoding algorithm $\mathsf{E}(m)$ that outputs a codeword $\hat{C} = \hat{c}_1, \ldots, \hat{c}_{\hat{n}}$ consisting of $\hat{n}$ blocks.

As observed by [54], achieving these locality properties requires a modification of the previous definition of non-malleability: Suppose a tampering function f only modifies one block of the codeword, then it is likely that the output of the decoding algorithm, D, remains unchanged in most locations. (Recall D gets as input an index $i \in [n]$ and will only access a few blocks of the codeword to recover the i-th block *of the message, so it may not detect the modification.) In this case, the (overall) decoding of the tampered codeword $f(\hat{C})$ (i.e. $(\mathsf{D}^{f(\hat{C})}(1), \ldots, \mathsf{D}^{f(\hat{C})}(n))$) can be highly related to the original message, which intuitively means it is highly malleable.*

To handle this issue, [54] consider a more fine-grained experiment. Informally, they require that for any tampering function f (within some class), there exists a simulator that, after every update instruction, computes a vector of decoded messages $\vec{m}^*$, and a set of indices $\mathcal{I} \subseteq [n]$. Here $\mathcal{I}$ denotes the coordinates of the underlying messages that have been tampered with. If $\mathcal{I} = [n]$, then the simulator thinks that the decoded messages are $\vec{m}^*$, which should be unrelated to the original message as well as the messages placed in each position by the updater. On the other hand, if $\mathcal{I} \subsetneq [n]$, the simulator thinks that all the messages not in $\mathcal{I}$ remain unchanged (equivalent to the most recent values *placed there by the simulator or the original message, if no update has occurred in that position), while those in $\mathcal{I}$ become $\perp$. This intuitively means the tampering function can do only one of the following cases:*

1. *It destroys a block (or blocks) of the underlying messages while keeping the other blocks unchanged, OR*

2. *If it modifies a block of the underlying message to a valid encoding, then it must have modified* all *blocks to encodings of unrelated messages, thus destroying the original message.*

It turns out, as shown by [54], that the above is sufficient for achieving tamper-resilience for RAM computations. Specifically, the above (together with an ORAM scheme) yields a compiler for any RAM program with the guarantee that any ad-

versary who gets input/output access to the compiled RAM program Π *running on compiled database* D *who can additionally apply tampering functions* $f \in \mathcal{F}$ *to the database* D *adaptively throughout the computation, learns no more than what can be learned given only input/output access to* Π *running on database* D*.*

Rewind attacks *As discussed earlier in chapter 1, when considering both leakage and tampering attacks (even just a single leakage query followed in a later round by a single tampering query) so-called* rewind attacks *become possible. In a rewind attack, the attacker does the following (1)* ***leak*** *information on only a "few" blocks of memory in rounds* $1, \ldots, i$*; (2)* ***wait*** *during rounds* $i+1, \ldots, j$ *until these memory locations are (with high probability) modified by the "updater" (the entity that models the honest computation on the data); (3)* ***re-write*** *the old information into these memory locations in round* $j+1$*, with the goal of causing the state of the computation to be* rewound*. Rewind attacks can be thwarted by ensuring that when the old information is written back, it becomes* inconsistent *with other positions of the codeword and an error is detected. On the other hand, a bad outcome of a rewind attack occurs if when decoding certain blocks of memory, with non-negligible probability, the old values from round* i *are recovered and no error is detected. This is a problem since such an outcome cannot be simulated by a simulator as required in the security definition: The decoding of these blocks depends on the original message and yet is no longer equal to "same" (since the values decoded are not the most recent values placed in those positions by the updater).*

6.1.1 Our Results

Our results show that any construction of locally decodable and updatable non-malleable codes in a threat model that allows for a rewind attack as above will require "high locality." Specifically, we show tight upper and lower bounds: (1) Every such construction will require super-constant locality, moreover; (2) Super-constant locality is sufficient for achieving constructions in the same threat model as [54] (which, as discussed, allows for rewind attacks). Throughout the paper, we assume that the decode and update procedures are non-adaptive *in the sense that the next block of the codeword accessed during decode/update does not depend on the contents of the previous blocks accessed. For non-adaptive decode and update we consider two settings: In the first, simpler setting, which we call* non-adaptive decode and update with deterministic accesses *we assume that once the encoding scheme* $\Pi = (\mathsf{E}, \mathsf{D}, \mathsf{UP})$ *is specified, for each* $n \in \mathbb{N}$*, the sets of codeword blocks* $S_i := S_i^{\mathsf{D}} \cup S_i^{\mathsf{UP}}$ *accessed in order to decode/update the* i*-th message block,* $i \in [n]$*, are fixed. This is a natural requirement, which holds true for the encoding scheme of [54]. This is the setting that was considered in the conference version of this paper [53]. In this work, we also consider a second, more complex setting, which we call* non-adaptive decode and update with randomized accesses*. Here we assume that the locations accessed by decode/update can depend on random coins* $r \in \{0,1\}^{\rho}$*, in addition to the message index* i*. Specifically, once the encoding scheme* $\Pi = (\mathsf{E}, \mathsf{D}, \mathsf{UP})$ *is specified, for each* $n \in \mathbb{N}$*, the sets of codeword blocks* $S_{i,r} := S_{i,r}^{\mathsf{D}} \cup S_{i,r}^{\mathsf{UP}}$ *corresponding to the locations accessed in order to decode/update the* i*-th message block,* $i \in [n]$*, with* random coins

r, are fixed. Note that since the access pattern depends on the random coins r, the blocks of the codeword accessed each time location i is decoded/updated may differ. Nevertheless, the access pattern is still non-adaptive *since the next accessed location depends only on the message index i to be decoded/updated and on random coins r, but* does not *depend on the contents of previously read blocks.*

We show the following:

Theorem 6.1.1 (Informal)**.** *Let λ be security parameter and let $\Pi = (\mathsf{E}, \mathsf{D}, \mathsf{UP})$ be a locally decodable and updatable non-malleable code that has* non-adaptive decode and update with deterministic accesses *in a threat model which allows for a rewind attack, which takes messages over alphabet Σ and outputs codewords over alphabet $\widehat{\Sigma}$, where $|\Sigma|, |\widehat{\Sigma}| \in \text{poly}(\lambda)$. Then, for $n = \text{poly}(\lambda)$, Π must have locality $\delta(n) \in \omega(1)$.*

Moreover, for every $\delta(n) \in \omega(1)$, there exists a computationally secure, locally decodable and updatable non-malleable code $\Pi = (\mathsf{E}, \mathsf{D}, \mathsf{UP})$ that has non-adaptive decode and update with deterministic accesses*, in a threat model which allows for a rewind attack, which takes messages over alphabet Σ and outputs codewords over alphabet $\widehat{\Sigma}$, where $|\Sigma|, |\widehat{\Sigma}| \in \text{poly}(\lambda)$ and such that for $n = \text{poly}(\lambda)$, Π has locality $\delta(n)$.*

Specifically, for the positive result, the construction of leakage resilient locally decodable updatable codes is secure against the same classes of tampering and leakage functions, $\mathcal{F}$, $\mathcal{G}$, as the construction of [54], but improves the locality from $O(\log n)$ to $\delta(n)$, for any $\delta(n) \in \omega(1)$.

In this work, we extend our lower bound beyond the setting considered in the *conference version [53], to the setting where the locations accessed by decode and update may be* randomized*:*

Theorem 6.1.2 (Informal)**.** *Let λ be security parameter and let $\Pi = (\mathsf{E}, \mathsf{D}, \mathsf{UP})$ be a locally decodable and updatable non-malleable code that has* non-adaptive decode and update with randomized accesses *which takes messages over alphabet Σ and outputs codewords over alphabet $\widehat{\Sigma}$, where $|\Sigma|, |\widehat{\Sigma}| \in \mathrm{poly}(\lambda)$, in a threat model which allows for a rewind attack. Then, for $n = \mathrm{poly}(\lambda)$, Π has locality $\delta(n) \in \omega(1)$.*

We emphasize that, for both lower bounds, our attacks work even in a threat model which allows only a single *bit of leakage in each round.*

We also note that since our attacks are efficient, the lower bounds are applicable in the computational setting and therefore also rule out constructions based on computational assumptions.

We leave as an open question extending our lower bound to the setting where the access pattern of decode and update, in addition to being randomized, may also be adaptive (i.e. the next position accessed by decode and/or update depends on the values read in the previous positions). In Section 6.1.2.3, we discuss the difficulties of extending our technique to the setting in which decode and update are adaptive.

6.1.2 Our Techniques

6.1.2.1 Lower Bound for Deterministic Access Patterns

We assume that there exists a locally decodable and updatable non-malleable code with non-adaptive decode and update and constant locality, c, for all message *lengths* $n = \mathrm{poly}(\lambda)$ *(where* n *is the number of blocks in the message). We then arrive at contradiction by showing that for every constant* c*, there exists a constant* $c' > c$*, such that the security guarantee cannot hold when encoding messages of length* $\mathcal{X}^{c'}$ *number of blocks, where* $\mathcal{X} \in \mathrm{poly}(\lambda)$ *is the bit length of the codeword blocks. Specifically, for messages of length* $n := \mathcal{X}^{c'} \in \mathrm{poly}(\lambda)$ *number of blocks, we will present an explicit (efficient) attacker and an explicit updater for which there cannot exist a simulator as required by the definition of locally decodable and updatable non-malleable codes.*

The attack we present is a rewind *attack, as discussed before. Intuitively, the main difficulty of designing the attack is to determine* which *positions of the codeword are to be leaked and subsequently re-wound to their original values so that with high probability in the real game, the corresponding message block will decode (with no error detected) to the original value in that position, as opposed to the most recently updated value. For purposes of our attack, we assume that the original message is either equal to* 0 *in all* n *blocks or equal to* 1 *in all* n *blocks.*

Sunflower Lemma *Informally speaking, a sunflower of size* k *is a collection of* k *sets* $\{S_1, \ldots, S_k\}$ *such that the intersection of any pair is equal to the core* core*, i.e. for sets* S_i *and* S_j*,* $S_i \cap S_j = \mathsf{core}$*. There exists* k *petals,* $S_i \setminus \mathsf{core}$*, and it is required that none of them are empty. Figure 6.1 shows a sunflower and the gray area represents a sample set* S_i*.*

We now describe how sunflowers are employed in our setting. We begin by considering the following collection of sets ($\{S_1, \ldots, S_{\hat{n}}\}$*) (which is not necessarily*

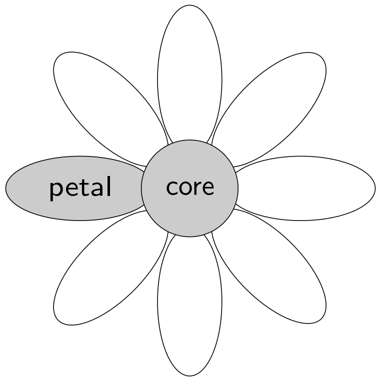

Figure 6.1: Illustration of a Sunflower

a sunflower): For $i \in [n]$, let the sets $S_i \subseteq [\hat{n}]$ correspond to the blocks (where each block has size $\mathcal{X} \in \text{poly}(\lambda)$ bits) of the codeword accessed in order to decode/update the i-th block of the message. Note that by the locality assumption, the size of each set S_i is $|S_i| = c$. The celebrated Sunflower Lemma of Erdős and Rado [68] tells us that any sufficiently large collection of sets (such as $\{S_1, \ldots, S_{\hat{n}}\}$), must contain a sunflower, whose size depends on the number of sets and the size of each set. Specifically, since the size of each set in $\{S_1, \ldots, S_{\hat{n}}\}$ is constant in our setting, the Sunflower Lemma of Erdős and Rado [68] implies that we can choose constant c' large enough such that when the message is of length $n := \mathcal{X}^{c'}$ number of blocks, we are guaranteed to have a sunflower $\mathsf{SF} := \{S_{i_0}, S_{i_1}, \ldots, S_{i_k}\}$, where $i_0, \ldots, i_k \in [n]$, of size $k+1$, where $k \gg \mathcal{X} \cdot c$. The collection of sets forming a sunflower are of the form S_{i_j} for $j \in [k]$ and the intersection of any pair is equal to the core core, i.e. $S_{i_j} \cap S_{i_\ell} = \mathsf{core}$ for all $j \neq \ell$. There also exist k petals $S_{i_j} \setminus \mathsf{core}$, and, as mentioned, none of them are empty. See Sections 6.2.1, 6.2.2 for more details.

The Compression Function *In the following, we explicitly define a compression function that is implicitly computed during a run of the security experiment. We* *define this function since its stability properties (discussed below) will be crucial for the analysis of our attack. Given a* fixed *initial codeword* $\hat{C}$ *and sunflower* SF *(as defined above) we define a (randomized) compression function* $F_{\hat{C}} : \{0, 1, \mathsf{same}\}^k \to \{0, 1\}^{\mathcal{X} \cdot c}$ *which takes as input values* $x_1, \ldots, x_k \in \{0, 1, \mathsf{same}\}$ *indicating how to update (or not) the corresponding message block* i_j, $j \in [k]$, *where* S_{i_j} *is in the sunflower. Specifically, for* $j = 1$ *to* k*: If* $x_j = \mathsf{same}$, *message block* i_j *does not get updated. Otherwise the update algorithm,* $\mathsf{UP}^{\hat{C}}(i_j, x_j)$ *is executed. The output of the function* $F_{\hat{C}}$ *is the contents of the sunflower core,* core, *after all the updates have been completed. Note that* core *can consist of at most* c *codeword blocks since* $\mathsf{core} \subseteq S_{i_j}$ *for all* $j \in [k]$*. Therefore, the output length of* $F_{\hat{C}}$ *is at most* $\mathcal{X} \cdot c$ *bits. Note that this means that* $F_{\hat{C}}$ *is a compression function, since we chose* $k \gg \mathcal{X} \cdot c$*. Now this, in turn, means that the output of* $F_{\hat{C}}$ *cannot contain all of the information in its input and satsifies certain stability properties. Indeed, it can be shown (cf. [61]) that with high probability over the choice of* $j^* \in [k]$, *the two distributions* $F_{\hat{C}}(X_1, \ldots, X_{j^*-1}, \mathsf{same}, X_{j^*+1}, \ldots, X_k)$ *and* $F_{\hat{C}}(X_1, \ldots, X_{j^*-1}, X_{j^*}, X_{j^*+1}, \ldots, X_k)$ *are statistically close when each* X_j, $j \in [k]$ *is chosen uniformly at random from* $\{0, 1, \mathsf{same}\}$*. See Sections 6.2.1, 6.2.3, 6.2.4 for more details.*

The Attacker and the Updater *The* attacker *first finds the sunflower* $\mathsf{SF} := \{S_{i_0}, S_{i_1}, \ldots, S_{i_k}\}$ *in polynomial time and then chooses* $j^* \in [k]$ *at random. In the first round (or multiple rounds if the attacker is allowed only a single bit of leakage) the attacker leaks the contents of the positions in* $\hat{C}$ *corresponding to the decoding of* i_{j^*}

($S_{i_{j^*}}$), minus the contents of the blocks in the core of the sunflower. We denote the entire leaked information by y_{j^*}. The attacker then writes those same values, y_{j^*}, back in the $k+1$-st round. The updater *chooses values $x_1, \ldots, x_k \in \{0, 1, \mathsf{same}\}$ and in each round from* 1 *to* k*, requests the corresponding update (i.e. update message block i_j to* 0*, if $x_j = 0$, update to* 1 *if $x_j = 1$ and do not update this block at all, if $x_j = \mathsf{same}$). See Section 6.2.5 for more details.*

Putting it All Together *Note that the input to the decoding algorithm when decoding position i_{j^*} is exactly: $(y_{j^*}, F_{\hat{C}_0}(X_1, \ldots, X_{j^*-1}, X_{j^*}, X_{j^*+1}, \ldots, X_k))$ (the contents of the positions in $\hat{C}$ corresponding to decoding of i_{j^*}, minus the contents of the blocks in the core of the sunflower, and the core itself). Additionally, note that since $\{S_{i_0}, S_{i_1}, \ldots, S_{i_k}\}$ form a sunflower, if $x_{j^*} = \mathsf{same}$, then the rewind attack has* no effect *(since the blocks in $S_{i_{j^*}} \setminus \mathsf{core}$ were not accessed during any update request) and so decode on input $(y_{j^*}, F_{\hat{C}_0}(X_1, \ldots, X_{j^*-1}, \mathsf{same}, X_{j^*+1}, \ldots, X_k))$ must correctly output* 1 *if the original encoding was* 1 *and* 0 *if the original encoding was* 0 *(without outputting $\perp$). Since $F_{\hat{C}}$ is a compression function, it means that with high probability decode on input $(y_{j*}, F_{\hat{C}}(X_1, \ldots, X_{j^*-1}, X_{j^*}, X_{j^*+1}, \ldots, X_k))$ will output* 1 *if the original encoding was* 1 *and* 0 *if the original encoding was* 0*, regardless of the value of X_{j^*}. Intuitively, since the output of decode now depends on the* original *message block in the i_{j^*}-th position, as opposed to the* most recently updated *value, the simulator must fail in at least one of the two cases (either when the original message was* 0 *or* 1*) and so the encoding scheme cannot satisfy the non-malleability definition. See Section 6.2.6 for more details.*

6.1.2.2 Lower Bound for Randomized Access Patterns

We next present a technical overview of the results that are new to this work and did not appear in the conference version [53]. In the following, we highlight the main technical challenges that arise in this setting and the approaches for addressing them.

- *We need to define a sunflower for the randomized access pattern setting. The natural way to do this is to consider the collection of sets $S_{i,r} \subseteq [\hat{n}]$, for $i \in [n]$, $r \in \{0,1\}^\rho$, corresponding to the blocks of the codeword accessed in order to decode/update the i-th block of the message with randomness r. Note that by the locality assumption, the size of each set $S_{i,r}$ is still $|S_{i,r}| = c$. Thus, we can still find a sufficiently large sunflower contained in the collection of sets above. However, when launching a rewind attack, while we can still randomly select (i, r_1), leak information about blocks corresponding to some set S_{i,r_1}, and write back to S_{i,r_1} at a later point, it is now possible that the decode algorithm will actually access a different set of blocks S_{i,r_2} during decoding in the final round, corresponding to some other randomness r_2. The key observation is that since the number of blocks $\hat{n}$ in the codeword is polynomial in security parameter λ, and since the total number of blocks accessed by each decode/update is a constant, c, then for each $i \in [n]$, there must be* some *set of c blocks $\mathcal{R}_i$ such that the same set $\mathcal{R}_i$ is accessed by the decode algorithm for position i with probability at least $1/\hat{n}^c$, which is inverse polynomial in the security parameter. We thus consider the collection of sets $\{S_{1,r_1^*}, \ldots, S_{n,r_n^*}\}$, where for $i \in [n]$, r_i^* is the* *randomness corresponding to the set $\mathcal{R}_i$ described above. We may now apply the Sunflower Lemma to obtain a sunflower* $\mathsf{SF} := \{S_{i_0,r^*_{i_0}}, S_{i_1,r^*_{i_1}}, \ldots, S_{i_k,r^*_{i_k}}\}$*, where* $i_0, \ldots, i_k \in [n]$*, of size* $k+1$*, where* $k \gg \mathcal{X} \cdot c^2$*, contained in the collection of sets. For technical reasons, we also choose a slightly smaller sunflower,* SF^-*, which is a subset of* SF*. We omit this part of the analysis from the overview and refer the reader to Section 6.3.1 for more details.*

- *Analogously to our previous rewind attack, we would like the attacker to select an index* $j^* \in [k]$*, leak the contents of the positions in* $\hat{C}$ *corresponding to decoding of* i_{j^*} *with random coins* $r^*_{i_{j^*}}$ $(S_{i_{j^*},r^*_{i_{j^*}}})$*, minus the contents of the blocks in the core of the sunflower, and then write those same values back in some later round (call this the* $k'+1$*-st round). Crucial to our analysis of the attack, is that if no update occurs on position* i_{j^*}*, then the rewind attack makes no change at all to the state of the codeword in the* $k'+1$*-st round. However, this is now not straightforward to argue. The reason is that the other* k' *updates that occur are now randomized and may overwrite locations contained in the set* $S_{i_{j^*},r^*_{i_{j^*}}} \setminus \mathsf{core}$*. In more detail, note that the adversary does not control the randomness of the updater and so cannot ensure that the updater accesses only the sets of the sunflower (corresponding to random coins* $r^*_{i_j}$*) during the updates. Indeed, since the updater samples its own randomness, it is extremely unlikely that this will happen. To address this issue, note that we now choose a larger sunflower* $\mathsf{SF} := \{S_{i_0,r^*_{i_0}}, S_{i_1,r^*_{i_1}}, \ldots, S_{i_k,r^*_{i_k}}\}$ *of size* $k+1$*, where* $k \gg \mathcal{X} \cdot c^2$*, whereas in the deterministic access pattern case we* *chose a sunflower of size* $k+1$ *where* $k \gg \mathcal{X} \cdot c$*. The fact that* k *is larger allows us to now choose a random subset of the sunflower* SF*, which is also a sunflower and still has sufficiently large size. Specifically, we choose a random subset* $\mathcal{T} \subseteq [k]$ *of size* k'*, such that* $\mathcal{T} := \{z_1, \ldots, z_{k'}\}$ *and consider updates to the sets* $S_{i_{z_j}}$*,* $j \in [k']$*. Now, due to the structure of the sunflower (which guarantees that all the sets* $(S_{i_j, r^*_{i_j}} \setminus \mathsf{core})$ *are pairwise disjoint) we will argue that (when* k' *is sufficiently small relative to* k*), with high probability over the random choice of* $\mathcal{T}$*,* $j^* \in [k']$*, and randomness of the updater, if no update occurs on position* $i_{z_{j^*}}$ *then there is no intersection between the updated sets and the chosen set* $S_{i_{z_{j^*}}, r^*_{z_{j^*}}} \setminus \mathsf{core}$*. See Sections 6.3.3, 6.3.5 for more details.*

- *Recall that the analysis of the previous case proceeds by showing that after the rewind attack occurs, the probability that the decode algorithm on position* $i_{z^*_j}$ *outputs the original value, is "close" whether or not an update occurred on position* $i_{z_{j^*}}$*. Unfortunately, the argument we used in the previous section only allows us to bound the difference in probabilities by a constant. This is no longer sufficient to complete the proof in the randomized case, since recall that in the randomized case, we can only argue that, when no update occurs, the decode algorithm on position* $i_{z_{j^*}}$ *outputs the original value with probability* $1/\hat{n}^c$*–when the attacker gets lucky and correctly guesses the random coins used by the decoder. This means that our previous techniques cannot rule out the case that when an update occurs, the decode algorithm outputs the original value with probability* 0*. Therefore, we must define a different event and show* *that this event is both observable given the output of the experiment, and occurs with* $1/\operatorname{poly}(\lambda)$ *probability. We then show that conditioned on this observable event occurring, the probability that the decode algorithm outputs the original value at another randomly chosen location* i_{ℓ^*} *is high. See Section 6.3.5 for more details.*

6.1.2.3 Difficulty of Extending Techniques to the Adaptive Setting

In both the deterministic and randomized case, our techniques crucially rely on using the access patterns to define a sunflower. Specifically, we consider a collection of sets $\{S_i\}$*, where each set* S_i *contains the indeces of the blocks of the codeword accessed in order to decode/update the* i*-th block of the message. If access patterns are adaptive, then as updates occur, the sets* S_i *may change, depending on the contents of the updates. This means that the structure of the sunflower may keep changing in each round of the experiment. Moreover, since the sets* S_i *are now defined in a way that depends on the messages input to the encode and update procedures, they are now unknown to the attacker. Therefore, it is not clear how to define an attack for such a case (since the attacker doesn't know the sets* S_i *and so doesn't know which positions to leak and write back), or which mathematical tools would be useful for analyzing an attack (since our main technical tool–the Sunflower Lemma–assumes a static collection of sets).*

6.1.2.4 Upper Bound

Overview of Locally Decodable and Updatable NMC [54] *In [54], the authors construct locally decodable and updatable non-malleable codes from a symmetric encryption scheme* $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$*, a standard (non-local) non-malleable code* $\mathsf{NMC} = (\mathsf{E}', \mathsf{D}')$*, and a collision resistant hash function family* H*.*

Encoding $\mathsf{E}(m)$*: In order to encode message* $m = (m_1, m_2, \dots, m_n)$*, they first generate the secret key* $\textsc{sk}$ *of by running* Gen *and choose hash function* $h \leftarrow H$*. Then the algorithm encrypts each message block* m_i *using* $\textsc{sk}$ *to get the encrypted blocks* $(e_1, e_2, \dots e_n)$ *corresponding to each message block. They then compute the Merkle tree* T_h *of these encrypted blocks using the collision-resistant hash function* h*, along with the root of the merkle tree* (T_h) *as* (R_h)*. They finally encode the secret key* $\textsc{sk}$*, the description of the hash function* h*, and the root* R_h *using the standard non-malleable code to generate* $c \leftarrow \mathsf{E}'(\textsc{sk}, h, R_h)$ *and output the encoded message* $C = (c, e_1, e_2, \dots, e_n, T_h)$*.*

Decoding $\mathsf{D}^C(i)$*: In order to decode the* i*-th block of* C*, they first recover* $(\textsc{sk}, h, R_h) = \mathsf{D}'(c)$*. Then the algorithm checks the consistency of the path* path *to leaf* e_i *in tree* T_h *and outputs* $m_i = \mathsf{Decrypt}(\textsc{sk}, e_i)$*. If any of the consistency checks, decoding of* c*, or decryption fails then the algorithm outputs* $\perp$*.*

Update $\mathsf{UP}^C(i, m'_i)$*: In order to update a message block* m_i *with* m'_i *the algorithm first recovers* $(\textsc{sk}, h, R_h) = \mathsf{D}'(c)$ *along with the path* path *to leaf* e_i *in tree* T_h*. If* D' *outputs* $\perp$ *then the algorithm updates the first block of* C *with* $\perp$ *denoting failure. Otherwise, computes the fresh encryption* e'_i *and updates the path* path' *and root*

R'_h in the merkle tree T_h by replacing the leaf e_i with e'_i and computing the path to the root. Finally, update c' as $c' \leftarrow \mathsf{E}'(\textsc{sk}, h, R'_h)$. The construction of [54] provides computational security against polynomial size tampering functions $f = (f_1, f_2)$ such that $f_1 \in \mathcal{F}$ tampers with c, and the underlying non-local NMC $(\mathsf{E}', \mathsf{D}')$ is secure against $\mathcal{F}$, whereas f_2 tampers with rest of the codeword arbitrarily. For more details please refer to Section 6.4.

Here we take advantage of the fact that codeword blocks are large–$\mathcal{X} \in \Omega(\lambda^{1/\mu})$ number of bits, for constant $0 < \mu < 1$–number of bits—to replace the Merkle Tree used in the original construction of [54] with an alternative data structure we call a t-slice Merkle Tree. Note that the $\Omega(\log \lambda)$ locality of the construction of [54] came from the fact that an entire path (and siblings) of the binary Merkle tree from root to leaf of length $\log(n)$ had to be traversed for each decode and update instruction. Our new data structure is a $t := \mathcal{X}^{1-\mu}$-ary tree and uses as a building block a collision resistant hash function $h : \{0,1\}^{\mathcal{X}} \to \{0,1\}^{\mathcal{X}^\mu}$ (note h has output length $\mathcal{X}^\mu \in \Omega(\lambda)$) and so, for messages of length $n = \mathrm{poly}(\lambda)$ blocks, an entire path of the tree from root to leaf will always have length at most $\delta(n)$, for any $\delta(n) \in \omega(1)$. Moreover, the root of the tree can be updated and verified without reading any of the siblings along the path from root to leaf, due to the use of a hash function with a specific structure. This allows us to achieve a locally decodable and updatable non-malleable codes with locality $\delta(n)$, for any $\delta(n) \in \omega(1)$. See Section 6.4 for more details.

6.2 Lower Bound

In this section we prove the following theorem:

Theorem 6.2.1. *Let λ be security parameter and let $\Pi = (\mathsf{E}, \mathsf{D}, \mathsf{UP})$ be a locally decodable and updatable non-malleable code that has* non-adaptive decode and update with deterministic accesses *which takes messages over alphabet Σ and outputs codewords over alphabet $\widehat{\Sigma}$, where $\log|\Sigma|, \log|\widehat{\Sigma}| \in \text{poly}(\lambda)$, in a threat model which allows for a rewind attack. Then, for $n := n(\lambda) \in \text{poly}(\lambda)$, Π has locality $\delta(n) \in \omega(1)$.*

We denote by $\mathcal{X} := \log|\widehat{\Sigma}| \in \text{poly}(\lambda)$ the number of bits in each block of the codeword. For purposes of the lower bound, we can take $\mathcal{X}$ to be any polynomial in λ (or smaller).

In the following, we assume that $\Pi = (\mathsf{E}, \mathsf{D}, \mathsf{UP})$ is a locally decodable and updatable non-malleable code with non-adaptive decode and update with deterministic accesses *and with* constant *locality. We then present an efficient rewind attacker along with an updater that break the security of Π, thus proving the theorem.*

6.2.1 Attack Preliminaries

Definition 6.2.1 (Sunflower)**.** *A sunflower (or Δ-system) is a collection of sets S_i for $1 \leq i \leq k$ such that the intersection of any two set is core Y, i.e. $S_i \cap S_j = \mathsf{core}$ for all $i \neq j$. There exists k petals $S_i^- := S_i \setminus \mathsf{core}$ and it's required that none of them are empty. A family of pairwise disjoint sets form a sunflower with an empty* *core.*

The following famous lemma is due to Erdős and Rado.

Lemma 6.2.1 (Sunflower Lemma [68]). *Let $\mathcal{F}$ be family of sets each of cardinality s. If $|\mathcal{F}| > s!(k-1)^s$ then $\mathcal{F}$ contains a sunflower with k petals.*

Definition 6.2.2 (Statistical Distance). *Let $\mathcal{D}_1$ and $\mathcal{D}_2$ be two distribution over a shared universe of outcomes. let $supp(\mathcal{D})$ be the set of values assumed by $\mathcal{D}$ with nonzero probability, and let $\mathcal{D}(u) := \Pr[\mathcal{D} = u]$. The statistical distance of $\mathcal{D}_1$ and $\mathcal{D}_2$ is defined as*

$$||\mathcal{D}_1 - \mathcal{D}_2||_{stat} := \frac{1}{2} \sum_{u \in supp(\mathcal{D}_1) \cup supp(\mathcal{D}_2)} |\mathcal{D}_1(u) - \mathcal{D}_2(u)|.$$

Definition 6.2.3 (Distributional Stability [61]). *Let $\mathcal{U}$ be a finite universe and $t, n \geq 1$ be integers. Let $\mathcal{D}_i$ for $1 \leq i \leq t$ be a collection of t mutually independent distributions over $\{0,1\}^n$ and F be a possibly-randomized mapping $F(x^1, \ldots, x^t) : \{0,1\}^{n \times t} \to \mathcal{U}$, for $j \in [t]$ let*

$$\gamma_j := \mathop{\mathbb{E}}_{y \sim \mathcal{D}_j} [||F(\mathcal{D}_1, \ldots, \mathcal{D}_{j-1}, y, \mathcal{D}_{j+1}, \ldots, \mathcal{D}_t) - F(\mathcal{D}_1, \ldots, \mathcal{D}_t)||_{stat}].$$

F is δ-distributionally stable for $\delta \in [0,1]$ with respect to $\mathcal{D}_1, \ldots, \mathcal{D}_t$ if

$$\frac{1}{t} \sum_{j=1}^{t} \gamma_j \leq \delta.$$

Lemma 6.2.2 (Compression Functions are Distributionally Stable [61]). *Let $R(x^1, \ldots, x^t) : \{0,1\}^{n \times t} \to \{0,1\}^{\leq t'}$ be any possibly-randomized mapping, for any $n, t, t' \in \mathbb{N}^+$. R is δ-distributionally stable with respect to any independent input distributions $\mathcal{D}_1, \ldots, \mathcal{D}_t$, where it may take either of the following two bounds:*

1. $\delta := \sqrt{\frac{\ln 2}{2} \cdot \frac{t'+1}{t}}$
2. $\delta := 1 - 2^{-\frac{t'}{t} - 3}$.

6.2.2 Applying the Sunflower Lemma

For $i \in [n]$, the sets $S_i \subseteq [\hat{n}]$ correspond to the blocks (each of size $\mathcal{X}$) of the codeword accessed in order to update/decode m_i (i.e. the set $S_i := S_i^{\mathsf{D}} \cup S_i^{\mathsf{UP}}$, where S_i^{D}, S_i^{UP} are the sets of blocks accessed by the decode and update procedures, respectively). By hypothesis, we have that for $i \in [n]$, $|S_i| = c$, for constant c. Choose $n = \mathcal{X}^{c'} \in \mathrm{poly}(\lambda)$, where c' is a constant such that

$$\mathcal{X}^{c'} > c! \cdot (22,500 \cdot c \cdot \mathcal{X})^c$$

Then by the Sunflower Lemma, $\{S_1, \ldots, S_n\}$ contains a sunflower with $k + 1 := 22,500 \cdot c \cdot \mathcal{X} + 1$ petals and core core. Let $\mathsf{SF} := \{S_{i_0}, S_{i_1}, \ldots, S_{i_k}\}$, where $i_0, \ldots, i_k \in [n]$. For codeword $\hat{C}$, let $\mathsf{core}(\hat{C})$ denote the content of the set of blocks that make up the core of the sunflower. For set S_ℓ, $\ell \in [n]$, let $\mathsf{set}_\ell(\hat{C})$ denote the content of the blocks in set S_ℓ, and let $\mathsf{set}_\ell^-(\hat{C})$ denote the content of the blocks in set S_ℓ^-, where $S_\ell^- := S_\ell \setminus \mathsf{core}$.

6.2.3 The Compression Functions

Given a fixed *initial codeword $\hat{C}$, sunflower* $\mathsf{SF} := \{S_{i_0}, \dots, S_{i_k}\}$*, where $i_0, \dots, i_k \in [n]$ (as defined above) with $k + 1 := 22,500 \cdot c \cdot \mathcal{X} + 1$ petals, define the following (randomized) function $F_{\hat{C}} : \{0, 1, \mathsf{same}\}^k \to \{0, 1\}^{\mathcal{X} \cdot c}$ as follows:*

- *On input $x_1, \dots, x_k \in \{0, 1, \mathsf{same}\}$*
- *For $j = 1$ to k:*
  - *If $x_j = \mathsf{same}$, run $\mathsf{UP}^{\hat{C}^{(j)}}(i_0, 0)$.*
  - *Otherwise run $\mathsf{UP}^{\hat{C}^{(j)}}(i_j, x_j)$.*

  where $\hat{C}^{(j)}$ denotes the codeword immediately before the j-th update.
- *Run $\mathsf{UP}^{\hat{C}^{(k+1)}}(i_0, 0)$.*
- *Output the contents of $\mathsf{core}(\hat{C}^{(k+1)+})$, where $\hat{C}^{(k+1)+}$ denotes the codeword immediately after the $k + 1$-st update.*

6.2.4 Closeness of Distributions

For $\ell \in [k]$, let X_ℓ be a random variable distributed as X, where X is distributed as $U_{\{0,1,\mathsf{same}\}}$, i.e. its value is chosen uniformly from the set $\{0, 1, \mathsf{same}\}$. Let $\hat{C}_0 \leftarrow \mathsf{E}(0 \dots 0)$ and $\hat{C}_1 \leftarrow \mathsf{E}(1 \dots 1)$. We prove the following claim, which will be useful in the subsequent analysis.

Claim 6.2.1. *For every $\hat{C}_0 \leftarrow \mathsf{E}(0 \dots 0)$ and $\hat{C}_1 \leftarrow \mathsf{E}(1 \dots 1)$, we have that:*

- *The statistical distance between* $F_{\hat{C}_0}(X_1, \ldots, X_{j-1}, \mathsf{same}, X_{j+1}, \ldots, X_k)$ *and* $F_{\hat{C}_0}(X_1, \ldots, X_k)$ *is at most* 0.1*, with probability at least* 0.8 *over* $j \sim [k]$*.*

- *The statistical distance between* $F_{\hat{C}_1}(X_1, \ldots, X_{j-1}, \mathsf{same}, X_{j+1}, \ldots, X_k)$ *and* $F_{\hat{C}_1}(X_1, \ldots, X_k)$ *is at most* 0.1*, with probability at least* 0.8 *over* $j \sim [k]$*.*

Proof. *First, by Lemma 6.2.2 and the fact that* $F_{\hat{C}}$ *is a compression function, we have that for every codeword* $\hat{C}$*:*

$$\frac{1}{k}\sum_{j=1}^{k} \mathop{\mathbb{E}}_{x\sim X}[||F_{\hat{C}}(X_1, \ldots, X_{j-1}, x, X_{j+1}, \ldots, X_k) - F_{\hat{C}}(X_1, \ldots, X_k)||_{stat}] < \sqrt{\frac{c \cdot \mathcal{X}}{k}}.$$

By linearity of expectation, we have

$$\mathop{\mathbb{E}}_{x\sim X}\left[\frac{1}{k}\sum_{j=1}^{k}(||F_{\hat{C}}(X_1, \ldots, X_{j-1}, x, X_{j+1}, \ldots, X_k) - F_{\hat{C}}(X_1, \ldots, X_k)||_{stat})\right] < \sqrt{\frac{c \cdot \mathcal{X}}{k}}.$$

Now, by Markov's inequality, we have that

$$\frac{1}{k}\sum_{j=1}^{k}(||F_{\hat{C}}(X_1, \ldots, X_{j-1}, \mathsf{same}, X_{j+1}, \ldots, X_k) - F_{\hat{C}}(X_1, \ldots, X_k)||_{stat}) < 3\sqrt{\frac{c \cdot \mathcal{X}}{k}}.$$

Applying Markov's inequality again, we have that with probability at least 0.8 *over choice of* $j \sim [k]$*,*

$$||F_{\hat{C}}(X_1, \ldots, X_{j-1}, \mathsf{same}, X_{j+1}, \ldots, X_k) - F_{\hat{C}}(X_1, \ldots, X_k)||_{stat} < 15 \cdot \sqrt{\frac{c \cdot \mathcal{X}}{k}} = 0.1,$$

where the final equality holds since we take $k + 1 := 22,500 \cdot c \cdot \mathcal{X} + 1$*. Finally,* *since the above holds for every $\hat{C}$, the claim is immediate. we have that for every $\hat{C}_0 \leftarrow \mathsf{E}(0 \ldots 0)$, and $\hat{C}_1 \leftarrow \mathsf{E}(1 \ldots 1)$:*

- *The statistical distance between $F_{\hat{C}_0}(X_1, \ldots, X_{j-1}, \mathsf{same}, X_{j+1}, \ldots, X_k)$ and $F_{\hat{C}_0}(X_1, \ldots, X_k)$ is at most 0.1, with probability at least 0.8 over $j \sim [k]$.*
- *The statistical distance between $F_{\hat{C}_1}(X_1, \ldots, X_{j-1}, \mathsf{same}, X_{j+1}, \ldots, X_k)$ and $F_{\hat{C}_1}(X_1, \ldots, X_k)$ is at most 0.1, with probability at least 0.8 over $j \sim [k]$.*

This concludes the proof of the claim.

6.2.5 The Attack

In this section we describe the polynomial-time attacker and updater:

Description of attacker:

- *Find the Sunflower $\mathsf{SF} := \{S_{i_0}, \ldots, S_{i_k}\}$, where $i_0, \ldots, i_k \in [n]$ and $k+1 := 22,500 \cdot c \cdot \mathcal{X} + 1$, contained in $\{S_1, \ldots, S_n\}$ in polynomial time.*[1]
- *Choose $j^* \sim [k]$*
- *In the first round, submit leakage function $\ell(\hat{C})$ defined as $\ell(\hat{C}) := \mathsf{set}^{-}_{i_{j^*}}(\hat{C})$ which returns Leaked, i.e. the contents of the positions in $\hat{C}$ corresponding to decoding of i_{j^*}, minus the contents of the blocks in the core of the sunflower.*[2]

[1]It is not hard to see that the original inductive proof of the Sunflower lemma by Erdős and Rado yields an efficient (polynomial in (n, c, k)) algorithm for finding the sunflower. For further information, see for example [**?**].

[2]If the attacker may leak only a single bit per round, we instead add here $r < \mathcal{X} \cdot c$ number of rounds where in each round the attacker leaks a single bit from $\mathsf{set}^{-}_{i_{j^*}}(\hat{C})$. During each of these rounds, the updater requests a "dummy" update, $\mathsf{UP}^{\hat{C}^{(j)}}(i_0, 0)$.

- *Wait until the $(k+2)$-nd round. In the $(k+2)$-nd round, choose tampering function f which replaces the contents of* $\mathsf{set}^{-}_{i_{j*}}(\hat{C}^{(k+1)+})$*—where* $\hat{C}^{(k+1)+}$ *denotes the contents of the codeword immediately after the $(k+1)$-st update– corresponding to decoding of i_{j^*}, minus the contents of the blocks in the core of the sunflower, with the values,* Leaked*, that were leaked via ℓ.*

Description of Updater:

- *Choose* $x_1, \ldots, x_k \sim \{0, 1, \mathsf{same}\}^k$.
- *For $j = 1$ to k:*
  - *If* $x_j = \mathsf{same}$*, request* $\mathsf{UP}^{\hat{C}^{(j)}}(i_0, 0)$
  - *Otherwise request* $\mathsf{UP}^{\hat{C}^{(j)}}(i_j, x_j)$

where $\hat{C}^{(j)}$ denotes the codeword immediately before the j-th update.

- *In round $k+1$, request* $\mathsf{UP}^{\hat{C}^{(k+1)}}(i_0, 0)$.

6.2.6 Attack Analysis

We begin with some notation and basic facts. Let J^* be the random variable corresponding to choice of j^* in the attack described above. For $j \in [k]$, let UP_{i_j} *be the event that location i_j gets updated and let* $\overline{\mathsf{UP}_{i_j}}$ *be the event that location i_j does not get updated. Recall that $\vec{m} \in \{0^n, 1^n\}$ denotes the original message. For $j \in [n]$, m_j denotes the original message in block j. $m_j^{(t)}$ denotes the decoding of $\hat{C}^{(t)}$ in the j-th position, where $\hat{C}^{(t)}$ denotes the codeword immediately before the t-th* *update.* $m_j^{(t)+}$ *denotes the decoding of* $\hat{C}^{(t)+}$ *in the* j*-th position, where* $\hat{C}^{(t)+}$ *denotes the codeword immediately after the* t*-th update. We have the following properties, which can be verified by inspection:*

Fact 6.2.1. *(a) For* $j \in [k]$, $\Pr[\mathsf{UP}_{i_j} \mid m_{i_j} = 0] = \Pr[\mathsf{UP}_{i_j} \mid m_{i_j} = 1] = 0.67$*;* $\Pr[\overline{\mathsf{UP}_{i_j}} \mid m_{i_j} = 0] = \Pr[\overline{\mathsf{UP}_{i_j}} \mid m_{i_j} = 1] = 0.33$*.*

(b) For $j \in [k]$*, if the* i_j*-th block of original message was a* $m_{i_j} = 0$*, then conditioned on an update occurring on block* i_j*,* $m_{i_j}^{(k+1)} = 0$ *with probability* 0.5 *and* $m_{i_j}^{(k+1)} = 1$ *with probability* 0.5*. Conditioned on no update occurring on block* i_j*,* $m_{i_j}^{(k+1)} = 0$ *with probability* 1*.*

(c) For $j \in [k]$*, if the* i_j*-th block of original message was a* $m_{i_j} = 1$*, then conditioned on an update occurring on block* i_j*,* $m_{i_j}^{(k+1)} = 1$ *with probability* 0.5 *and* $m_{i_j}^{(k+1)} = 0$ *with probability* 0.5*. Conditioned on no update occurring on block* i_j*,* $m_{i}^{(k+1)} = 1$ *with probability* 1*.*

(d) For $j \in [k]$*,* $m_{i_j}^{(k+1)} = m_{i_j}^{(k+1)+}$*, since in the* $k+1$*-st round only the position* i_0 *gets updated.*

We next present the main technical lemma of this section:

Lemma 6.2.3. *For the attack and updater specified in Section 6.2.5:*

Case 1: *If the original message was* $\vec{m} = \vec{0}$*, then with probability at least* 0.7*,* $m_{i_{J^*}}^{(k+2)} = 0$*.*

Case 2: *If the original message was* $\vec{m} = \vec{1}$*, then with probability at least* 0.7*,* $m_{i_{J^*}}^{(k+2)} = 1$*.*

We first show how to use Lemma 6.2.3 to complete the proof of Theorem 6.2.1 and then present the proof of Lemma 6.2.3.

Proof (of Theorem 6.2.1.). *We show that the above claim implies that the candidate scheme is not secure under Definition 3.2.3 and Definition 3.2.4. Definition 3.2.4 requires the existence of a simulator* Sim *which (for the above attack and updater) outputs one of* $\{\mathsf{same}, \perp\} \cup \{0,1\}^\kappa$ *for the decoding of each position* $i \in [n]$ *in each round* $j \in [k+2]$. *We denote by* $m_{i,\mathsf{Sim}}^{(j)}$ *the output of* Sim *in round* j *for position* i. *Recall that if* Sim *outputs* same *in round* j *for position* i, *then the output of the experiment in the corresponding position, denoted* $\widetilde{m}_{i,\mathsf{Sim}}^{(j)}$, *is set to* $\widetilde{m}_{i,\mathsf{Sim}}^{(j)} := m_i^{(j-1)+}$.

We begin by defining the following notation for each $j \in [k]$*:*

$$p_{up,j}^0 := \Pr[m_{i_j,\mathsf{Sim}}^{(k+2)} = \mathsf{same} \mid m_{i_j} = 0 \wedge \mathsf{UP}_{i_j}]$$

$$p_{up,j}^1 := \Pr[m_{i_j,\mathsf{Sim}}^{(k+2)} = \mathsf{same} \mid m_{i_j} = 1 \wedge \mathsf{UP}_{i_j}]$$

$$p_{\overline{up},j}^0 := \Pr[m_{i_j,\mathsf{Sim}}^{(k+2)} = \mathsf{same} \mid m_{i_j} = 0 \wedge \overline{\mathsf{UP}_{i_j}}]$$

$$p_{0,j}^0 := \Pr[m_{i_j,\mathsf{Sim}}^{(k+2)} = 0 \mid m_{i_j} = 0]$$

$$p_{0,j}^1 := \Pr[m_{i_j,\mathsf{Sim}}^{(k+2)} = 0 \mid m_{i_j} = 1]$$

Note that since Sim *does not see the original message, we have that for each* $j \in [k]$*:*

$$(a)\, p_{up,j}^0 = p_{up,j}^1 \qquad (b)\, p_{0,j}^0 = p_{0,j}^1. \tag{6.2.1}$$

Additionally we have, for each $j \in [k]$::

$$
\begin{aligned}
&\Pr[m^{(k+2)}_{i_j,\mathsf{Sim}} = \mathsf{same} \wedge m^{(k+1)}_{i_j} = 0 \wedge \mathsf{UP}_{i_j} \mid m_{i_j} = 0] \\
&\quad = \Pr[\mathsf{UP}_{i_j} \mid m_{i_j} = 0] \cdot \Pr[m^{(k+2)}_{i_j,\mathsf{Sim}} = \mathsf{same} \mid m_{i_j} = 0 \wedge \mathsf{UP}_{i_j}] \\
&\qquad \cdot \Pr[m^{(k+1)}_{i_j} = 0 \mid m_{i_j} = 0 \wedge \mathsf{UP}_{i_j}] \\
&\quad = 0.67 \cdot p^0_{up,j} \cdot 0.5, \qquad (6.2.2)
\end{aligned}
$$

Where the first equality follows since $(m^{(k+2)}_{i_j,\mathsf{Sim}} = \mathsf{same} \mid m_{i_j} = 0 \wedge \mathsf{UP}_{i_j})$ *and* $(m^{(k+1)}_{i_j} = 0 \mid m_{i_j} = 0 \wedge \mathsf{UP}_{i_j})$ *are independent events and the last line follows from Fact 6.2.1, items (a) and (b). Similarly, for each $j \in [k]$:*

$$
\begin{aligned}
&\Pr[m^{(k+2)}_{i_j,\mathsf{Sim}} = \mathsf{same} \wedge m^{(k)}_{i_j} = 0 \wedge \mathsf{UP}_{i_j} \mid m_{i_j} = 1] \\
&\quad = \Pr[\mathsf{UP}_{i_j} \mid m_{i_j} = 1] \cdot \Pr[m^{(k+2)}_{i_j,\mathsf{Sim}} = \mid m_{i_j} = 1 \wedge \mathsf{UP}_{i_j}] \\
&\qquad \cdot \Pr[m^{(k+1)}_{i_j} = 0 \mid m_{i_j} = 1 \wedge \mathsf{UP}_{i_j}] \\
&\quad = 0.67 \cdot p^1_{up,j} \cdot 0.5 \\
&\quad = 0.67 \cdot p^0_{up,j} \cdot 0.5, \qquad (6.2.3)
\end{aligned}
$$

where the second to last line follows from Fact 6.2.1, items (a) and (c), and the last *line follows due to (6.2.1)(a). Moreover, we have for each $j \in [k]$:*

$$\begin{aligned}
&\Pr[m^{(k+2)}_{i_j,\mathsf{Sim}} = \mathsf{same} \wedge m^{(k+1)}_{i_j} = 0 \wedge \overline{\mathsf{UP}_{i_j}} \mid m_{i_j} = 0] \\
&\quad = \Pr[\overline{\mathsf{UP}_{i_j}} \mid m_{i_j} = 0] \cdot \Pr[m^{(k+2)}_{i_j,\mathsf{Sim}} = \mathsf{same} \mid m_{i_j} = 0 \wedge \overline{\mathsf{UP}_{i_j}}] \\
&\quad = 0.33 \cdot p^0_{\overline{up},j}, \qquad (6.2.4)
\end{aligned}$$

where the last line follows from Fact 6.2.1, item (a). Finally, for each $j \in [k]$:

$$\Pr[m^{(k+2)}_{i_j,\mathsf{Sim}} = \mathsf{same} \wedge m^{(k+1)}_{i_j} = 0 \wedge \overline{\mathsf{UP}_{i_j}} \mid m_{i_j} = 1] = 0, \qquad (6.2.5)$$

since if no update occurs on position i_j then $m^{(k+1)}_{i_j} = 1$.

Given Lemma 6.2.3, in order for Sim *to succeed, if the original message was $\vec{m} = \vec{0}$, then $m^{(k+2)}_{i_{J^*},\mathsf{Sim}}$ must be equal to 0 with probability (nearly) 0.7, whereas if the original message was $\vec{m} = \vec{1}$, then $m^{(k+2)}_{i_{J^*},\mathsf{Sim}}$ must be equal to 1 with probability (nearly) 0.7. Thus we have that:*

$$\begin{aligned}
0.7 - \mathsf{negl}(\lambda) &= \sum_{j\in[k]} \Pr[J^* = j] \cdot \Pr[\widetilde{m}^{(k+2)}_{i_j,\mathsf{Sim}} = 0 \mid m_{i_j} = 0] \\
&= \sum_{j\in[k]} \frac{1}{k} \cdot (\Pr[m^{(k+2)}_{i_j,\mathsf{Sim}} = \mathsf{same} \wedge m^{(k+1)}_{i_j} = 0 \wedge \mathsf{UP}_{i_j} \mid m_{i_j} = 0] \\
&\qquad + \Pr[m^{(k+2)}_{i_j,\mathsf{Sim}} = \mathsf{same} \wedge m^{(k+1)}_{i_j} = 0 \wedge \overline{\mathsf{UP}_{i_j}} \mid m_{i_j} = 0] \\
&\qquad + \Pr[m^{(k+2)}_{i_j,\mathsf{Sim}} = 0 \mid m_{i_j} = 0]) \\
&= \sum_{j\in[k]} \frac{1}{k} \cdot (0.67 \cdot p^0_{up,j} \cdot 0.5 + 0.33 \cdot p^0_{\overline{up},j} + p^0_{0,j}), \qquad (6.2.6)
\end{aligned}$$

where the second equality follows due to the fact that if $m^{(k+2)}_{i_j,\mathsf{Sim}} = \mathsf{same}$ *then* $\widetilde{m}^{(k+2)}_{i_j} = m^{(k+1)+}_{i_j}$ *and Fact 6.2.1, item (d) (which says that* $m^{(k+1)+}_{i_j} = m^{(k+1)}_{i_j}$*), and the last line follows due to (2) and (4). On the other hand we have:*

$$\begin{aligned}
0.3 + \mathsf{negl}(\lambda) &= \sum_{j\in[k]} \Pr[J^* = j] \cdot \Pr[\widetilde{m}^{(k+1)}_{i_j,\mathsf{Sim}} = 0 \mid m_{i_j} = 1] \\
&= \sum_{j\in[k]} \frac{1}{k} \cdot (\Pr[m^{(k+2)}_{i_j,\mathsf{Sim}} = \mathsf{same} \wedge m^{(k+1)}_{i_j} = 0 \wedge \mathsf{UP}_{i_j} \mid m_{i_j} = 1] \\
&\qquad + \Pr[\Pr[m^{(k+2)}_{i_j,\mathsf{Sim}} = \mathsf{same} \wedge m^{(k+1)}_{i_j} = 0 \wedge \overline{\mathsf{UP}_{i_j}} \mid m_{i_j} = 1] \\
&\qquad + \Pr[\Pr[m^{(k+2)}_{i_j,\mathsf{Sim}} = 0 \mid m_{i_j} = 1]) \\
&= \sum_{j\in[k]} \frac{1}{k} \cdot (0.67 \cdot p^0_{up,j} \cdot 0.5 + p^1_{0,j}) \\
&= \sum_{j\in[k]} \frac{1}{k} \cdot (0.67 \cdot p^0_{up,j} \cdot 0.5 + p^0_{0,j}). \qquad (6.2.7)
\end{aligned}$$

where the second equality follows due to the fact that if $m^{(k+2)}_{i_j,\mathsf{Sim}} = \mathsf{same}$ *then* $\widetilde{m}^{(k+2)}_{i_j} = m^{(k+1)+}_{i_j}$ *and Fact 6.2.1, item (d) (which says that* $m^{(k+1)+}_{i_j} = m^{(k+1)}_{i_j}$*), the second to last line follows due to (3) and (5) and the last line follows due to (1)(b). But subtracting (7) from (6), this implies that* $0.33 \cdot \sum_{j\in[k]} \frac{1}{k} \cdot p^{0}_{\overline{up},j} \geq 0.4$*, which is impossible since for each* $j \in [k]$, $p^{0}_{\overline{up},j} \leq 1$. *Thus we have reached contradiction and so the theorem is proved.*

We conclude by proving the Lemma.

Proof (of Lemma 6.2.3)**.** *Let* $y^0_{j^*} := \mathsf{set}^-_{i_{j^*}}(\hat{C}_0)$ *and* $y^1_{j^*} := \mathsf{set}^-_{i_{j^*}}(\hat{C}_1)$ *The proof relies on the fact that, in the* $(k{+}2)$*-nd round, decode takes as input* $\mathsf{D}(y^0_{j^*}, F_{\hat{C}_0}(X_1,\ldots,X_k))$ *in Case* 1 *and* $\mathsf{D}(y^1_{j^*}, F_{\hat{C}_1}(X_1,\ldots,X_k))$ *in Case* 2*, Now note that, due to the struc-*

ture of the Sunflower, updates to positions $i_0, \ldots, i_{j^*-1}, i_{j^*+1}, \ldots, i_k$ *do not modify the blocks in* $S^-_{i_{j^*}}$ *(corresponding to the contents of* $\mathsf{set}^-_{i_{j^*}}(\hat{C}_0)$ *or* $\mathsf{set}^-_{i_{j^*}}(\hat{C}_1)$*). So* $\mathsf{D}(y^0_{j^*}, F_{\hat{c}_0}(X_1, \ldots, X_{j^*-1}, \mathsf{same}, X_{j^*+1}, \ldots, X_k)) = 0$ *with overwhelming probability and* $\mathsf{D}(y^1_{j^*}, F_{\hat{c}_1}(X_1, \ldots, X_{j^*-1}, \mathsf{same}, X_{j^*+1}, \ldots, X_k)) = 1$ *with overwhelming probability, since when* $X_j = \mathsf{same}$*, the rewind attack has no effect and decode outputs the original message.*

Moreover, due to Claim 6.2.1 and the fact that $y^0_{j^*}$ *(resp.* $y^1_{j^*}$*) is fully determined by* $\hat{C}_0$ *(resp.* $\hat{C}_1$*), which is part of the description of the compression function* $F_{\hat{C}_0}$ *(resp.* $F_{\hat{C}_1}$*), we have that for every* $\hat{C}_0 \leftarrow \mathsf{E}(0 \ldots 0)$ *and* $\hat{C}_1 \leftarrow \mathsf{E}(1 \ldots 1)$*:*

1. *The statistical distance between* $(y^0_j, F_{\hat{C}_0}(X_1, \ldots, X_{j^*-1}, \mathsf{same}, X_{j^*+1}, \ldots, X_k))$ *and* $(y^0_j, F_{\hat{C}_0}(X_1, \ldots, X_k))$ *is at most* 0.1*, with probability at least* 0.8 *over* $j^* \sim [k]$*.*

2. *The statistical distance between* $(y^1_j, F_{\hat{C}_1}(X_1, \ldots, X_{j^*-1}, \mathsf{same}, X_{j^*+1}, \ldots, X_k))$ *and* $(y^1_j, F_{\hat{C}_1}(X_1, \ldots, X_k))$ *is at most* 0.1*, with probability at least* 0.8 *over* $j^* \sim [k]$*.*

Hence each will not be satisfied with probability at most 0.2*. Now, conditioned on each being satisfied, it can be concluded from (1) that the probability of* $\mathsf{D}(y^0_j, F_{\hat{C}_0}(X_1, \ldots, X_k)) = 1$ *is at most* 0.1*. Similarly from (2),* $\mathsf{D}(y^1_j, F_{\hat{C}_1}(X_1, \ldots, X_k)) = 0$ *with probability at most* 0.1*. Taking a union bound, we have that in each case, the* D *procedure will fail to output the original message with probability at most* 0.3*. This means that with probability at least* 0.7 *over all coins,* $\mathsf{D}(y^0_{j^*}, F_{\hat{C}_0}(X_1, \ldots, X_k)) = 0$*, and with probability at least* 0.7 *over all coins*

$\mathsf{D}(y^1_{j^*}, F_{\hat{C}_1}(X_1, \ldots, X_k)) = 1$, *completing the proof of the claim.*

6.3 Extending Lower Bound to Randomized Decode/Update

In this section we prove the following theorem:

Theorem 6.3.1. *Let* λ *be security parameter and let* $\Pi = (\mathsf{E}, \mathsf{D}, \mathit{UP})$ *be a locally decodable and updatable non-malleable code that has* non-adaptive decode and update with randomized accesses *which takes messages over alphabet* Σ *and outputs codewords over alphabet* $\widehat{\Sigma}$, *where* $\log|\Sigma|, \log|\widehat{\Sigma}| \in \mathrm{poly}(\lambda)$, *in a threat model which allows for a rewind attack. Then, for* $n := n(\lambda) \in \mathrm{poly}(\lambda)$, Π *has locality* $\delta(n) \in \omega(1)$.

As before, we denote by $\mathcal{X} := \log|\widehat{\Sigma}| \in \mathrm{poly}(\lambda)$ *the number of bits in each block of the codeword. For purposes of the lower bound, we can take* $\mathcal{X}$ *to be any polynomial in* λ *(or smaller).*

In the following, we assume that $\Pi = (\mathsf{E}, \mathsf{D}, \mathit{UP})$ *is a locally decodable and updatable non-malleable code with* non-adaptive decode and update with randomized accesses *and with* constant *locality. We then present an efficient rewind attacker along with an updater that break the security of* Π, *thus proving the theorem.*

The formal analysis is at some points quite similar to the analysis in Section 6.2. For completeness, we present the full proof. To aid the reader, we point out which parts of the proof are similar to the previous analysis and which parts are significantly different.

6.3.1 Applying the Sunflower Lemma

As in Section 6.2, we will use the access patterns to define a sunflower, but since the access patterns are now randomized, we will have to take into account the randomness of the decode/update procedures as well as the position $i \in [n]$. For $i \in [n]$, $r \in \{0,1\}^{\rho(\lambda)}$ the sets $S_{i,r} \subseteq [\hat{n}]$ correspond to the blocks (each of size $\mathcal{X}$) of the codeword accessed in order to update/decode m_i with randomness r (i.e. the set $S_{i,r} := S^{\mathsf{D}}_{i,r} \cup S^{\mathsf{UP}}_{i,r}$ where $S^{\mathsf{D}}_{i,r}$, $S^{\mathsf{UP}}_{i,r}$ are the sets of blocks accessed when the decode/update procedures are run with randomness $r \in \{0,1\}^{\rho(\lambda)}$). For $i \in [n]$, let $\mathcal{R}_i$ correspond to a particular access pattern consisting of a set of c blocks that occurs with probability at least $1/\hat{n}^c$ when decoding/updating position i (note that such an access pattern must exist by an averaging argument). For simplicity of notation, we assume that each such set $\mathcal{R}_i$ corresponds to random coins $r^*_i \in \{0,1\}^\rho$, where $\rho := \rho(\lambda) := c \cdot \log(\hat{n}) = O(\log \lambda)$ (i.e. for $i \in [n]$, $S_{i,r^*_i} = \mathcal{R}_i$). We also require a larger sunflower for the randomized case. Consider the system of sets $\{S_{1,r^*_1}, \dots, S_{n,r^*_n}\}$. By hypothesis, we have that for $i \in [n]$, $|S_{i,r^*_i}| = c$, for constant c. Choose $n = \mathcal{X}^{c'} \in \mathrm{poly}(\lambda)$, where c' is a constant such that

$$\mathcal{X}^{c'} > c! \cdot (15,840,011 \cdot c^2 \cdot \mathcal{X})^c$$

Then by the Sunflower Lemma, $\{S_{1,r^*_1}, \dots, S_{n,r^*_n}\}$ contains a sunflower with $k+1 := 15,840,011 \cdot c^2 \cdot \mathcal{X} + 1$ petals and core core*. Let $\mathsf{SF} := \{S_{\ell_0,r^*_{\ell_0}}, S_{\ell_1,r^*_{\ell_1}}, \dots, S_{\ell_k,r^*_{\ell_k}}\}$, where $\ell_0, \dots, \ell_k \in [n]$. As before, for codeword $\hat{C}$, let $\mathsf{core}(\hat{C})$ denote the content of* *the set of blocks that make up the core of the sunflower. Additionally, we slightly extend the previous notation as follows: For set $S_{\ell,r}$, $\ell \in [n]$, let $\mathsf{set}_{\ell,r}(\hat{C})$ denote the content of the blocks in set $S_{\ell,r}$ and for $j \in [k]$, let $\mathsf{set}^{-}_{\ell_j,r^*_{\ell_j}}(\hat{C})$ denote the content of the blocks in set $S^{-}_{\ell_j,r^*_{\ell_j}}$, where $S^{-}_{\ell_j,r^*_{\ell_j}} := S_{\ell_j,r^*_{\ell_j}} \setminus \mathsf{core}$. Let $\mathcal{S}^{hi}$ denote the set of blocks of the codeword that are accessed by $\mathsf{D}[i], \mathit{UP}[i,v]$ with probability at least $1/11c$ over random choice of $i \in [n]$ and over choice of random coins (note that by the non-adaptivity assumption, the set does not depend on the value v input to UP). Note that $|\mathcal{S}^{hi}| \leq 11c^2$. We remove the at most $11c^2$ sets $S_{\ell_j,r^*_{\ell_j}}$ from SF such that $S^{-}_{\ell_j,r^*_{\ell_j}} \cap \mathcal{S}^{hi} \neq \emptyset$. This step was not needed for the analysis in Section 6.2, but will be necessary to complete the analysis in the proof of Lemma 6.3.1. Let $\mathsf{SF}^{-} := \{S_{i_0}, S_{i_1}, \ldots, S_{i_{\widetilde{k}}}\}$ denote the remaining sets. Note that SF^{-} is a sunflower that has size at least $\widetilde{k} + 1 \geq 15,840,000 \cdot c^2 \cdot \mathcal{X} + 1$.*

6.3.2 The Compression Functions

Given a fixed *initial codeword $\hat{C}$, sunflower $\mathsf{SF}^{-} := \{S_{i_0}, \ldots, S_{i_{\widetilde{k}}}\}$, where $i_0, \ldots, i_{\widetilde{k}} \in [n]$ (as defined above) with $\widetilde{k} + 1 := 15,840,000 \cdot c^2 \cdot \mathcal{X} + 1$ petals, and random subset $\mathcal{T} \subseteq [\widetilde{k}]$ of size $k' := 1,440,000 \cdot c \cdot \mathcal{X}$, where $\mathcal{T} = \{z_1, \ldots, z_{k'}\}$ (where we assume the z_i's are ordered in lexicographic order), define the following (randomized) function $F_{\hat{C},\mathcal{T},r} : \{0, 1, \mathsf{same}\}^{k'} \to \{0,1\}^{\mathcal{X} \cdot c}$ as follows:*

- *On input $x_1, \ldots, x_{k'} \in \{0, 1, \mathsf{same}\}$*
- *Parse $r := r_1, \ldots, r_{k'+1}$*
- *For $j = 1$ to k':*

  - *If* $x_j = \mathsf{same}$*, run* $\mathsf{UP}^{\hat{C}^{(j)}}(i_0, 0; r_j)$.

  - *Otherwise run* $\mathsf{UP}^{\hat{C}^{(j)}}(i_{z_j}, x_j; r_j)$.

  where $\hat{C}^{(j)}$ *denotes the codeword immediately before the* j*-th update.*

- *Run* $\mathsf{UP}^{\hat{C}^{(k'+1)}}(i_0, 0; r_{k'+1})$.

- *Output the contents of* $\mathsf{core}(\hat{C}^{(k'+1)+})$*, where* $\hat{C}^{(k'+1)+}$ *denotes the contents of the codeword immediately after the* $k'+1$*-st update.*

Note that the subset $\mathcal{T}$ *was not needed in the analysis in Section 6.2, and is added here since it will be used in the attack in Section 6.3.4 and in the proof of Lemma 6.3.1. Otherwise, the compression function defined above is essentially the same as before, except the randomness for encode/decode is "hardwired" into the compression function.*

6.3.3 Closeness of Distributions

For $\ell \in [k']$*, let* X_ℓ *be a random variable distributed as* X*, where* X *is distributed as* $U_{\{0,1,\mathsf{same}\}}$*, i.e. its value is chosen uniformly from the set* $\{0, 1, \mathsf{same}\}$*. Let* $\hat{C}_0 \leftarrow \mathsf{E}(0\ldots0)$ *and* $\hat{C}_1 \leftarrow \mathsf{E}(1\ldots1)$*. For* $z_j \in \mathcal{T}$*, let* $y_j^0 := \mathsf{set}_{i_{z_j}}(\hat{C}_0) \setminus \mathsf{core}(\hat{C}_0)$ *denote the contents of the positions in* $\hat{C}_0$ *corresponding to decoding of* i_{z_j}*, minus the contents of the blocks in the core of the sunflower. Similarly, let* $y_j^1 := \mathsf{set}_{i_{z_j}}(\hat{C}_1) \setminus \mathsf{core}(\hat{C}_1)$ *denote the contents of the positions in* $\hat{C}_1$ *corresponding to decoding of* i_{z_j}*, minus the contents of the blocks in the core of the sunflower. We prove the following claim (similar to Claim 6.2.1 in Section 6.2, but with different* *parameters), which will be useful in the subsequent analysis.*

Claim 6.3.1. *For every r and every $\hat{C}_0 \leftarrow \mathsf{E}(0\dots0)$ and $\hat{C}_1 \leftarrow \mathsf{E}(1\dots1)$, we have that:*

- *With probability at least 0.9 over $j \sim [k']$, the statistical distance between $F_{\hat{C}_0}(X_1,\dots,X_{j-1},\mathsf{same},X_{j+1},\dots,X_{k'})$ and $F_{\hat{C}_0}(X_1,\dots,X_{j-1},1,X_{j+1},\dots,X_{k'})$ is at most 0.1.*

- *With probability at least 0.9 over $j \sim [k']$, the statistical distance between $F_{\hat{C}_1}(X_1,\dots,X_{j-1},\mathsf{same},X_{j+1},\dots,X_{k'})$ and $F_{\hat{C}_1}(X_1,\dots,X_{j-1},0,X_{j+1},\dots,X_{k'})$ is at most 0.1.*

Proof. *First, by Lemma 6.2.2 and the fact that $F_{\hat{C}}$ is a compression function, we have that for every codeword $\hat{C}$:*

$$\frac{1}{k'}\sum_{j=1}^{k'} \mathop{\mathbb{E}}_{x\sim X}[||F_{\hat{C}}(X_1,\dots,X_{j-1},x,X_{j+1},\dots,X_{k'}) - F_{\hat{C}}(X_1,\dots,X_{k'})||_{stat}] < \sqrt{\frac{c\cdot\mathcal{X}}{k}}$$

By linearity of expectation, we have

$$\mathop{\mathbb{E}}_{x\sim X}\left[\frac{1}{k'}\sum_{j=1}^{k'}(||F_{\hat{C}}(X_1,\dots,X_{j-1},x,X_{j+1},\dots,X_{k'}) - F_{\hat{C}}(X_1,\dots,X_{k'})||_{stat})\right] < \sqrt{\frac{c\cdot\mathcal{X}}{k'}}.$$

Now, by Markov's inequality, we have that

$$\frac{1}{k'}\sum_{j=1}^{k'}(||F_{\hat{C}}(X_1,\dots,X_{j-1},\mathsf{same},X_{j+1},\dots,X_{k'}) - F_{\hat{C}}(X_1,\dots,X_{k'})||_{stat}) < 3\sqrt{\frac{c\cdot\mathcal{X}}{k'}}.$$

Applying Markov's inequality again, we have that with probability at least 0.95 over *choice of* $j \sim [k]$,

$$||F_{\hat{C}}(X_1, \ldots, X_{j-1}, \mathsf{same}, X_{j+1}, \ldots, X_{k'}) - F_{\hat{C}}(X_1, \ldots, X_{k'})||_{stat} < 60 \cdot \sqrt{\frac{c \cdot \mathcal{X}}{k'}} = 0.05,$$

where the final equality holds since we take $k' + 1 := 1,440,000 \cdot c \cdot \mathcal{X} + 1$.

Similarly, we can repeat the above analysis to obtain that with probability at least 0.95 *over choice of* $j \sim [k']$,

$$||F_{\hat{C}}(X_1, \ldots, X_{j-1}, 1, X_{j+1}, \ldots, X_{k'}) - F_{\hat{C}}(X_1, \ldots, X_{k'})||_{stat} < 60 \cdot \sqrt{\frac{c \cdot \mathcal{X}}{k'}} = 0.05,$$

and that with probability at least 0.95 *over choice of* $j \sim [k']$,

$$||F_{\hat{C}}(X_1, \ldots, X_{j-1}, 0, X_{j+1}, \ldots, X_{k'}) - F_{\hat{C}}(X_1, \ldots, X_{k'})||_{stat} < 60 \cdot \sqrt{\frac{c \cdot \mathcal{X}}{k'}} = 0.05.$$

Finally, since the above hold for every $\hat{C}$*, we have by a union bound that for every* $\hat{C}_0 \leftarrow \mathsf{E}(0 \ldots 0)$*, and* $\hat{C}_1 \leftarrow \mathsf{E}(1 \ldots 1)$*:*

- *With probability at least* 0.9 *over* $j \sim [k']$*, the statistical distance between* $F_{\hat{C}_0}(X_1, \ldots, X_{j-1}, \mathsf{same}, X_{j+1}, \ldots, X_{k'})$ *and* $F_{\hat{C}_0}(X_1, \ldots, X_{j-1}, 1, X_{j+1}, \ldots, X_{k'})$ *is at most* 0.1.

- *With probability at least* 0.9 *over* $j \sim [k']$*, the statistical distance between* $F_{\hat{C}_1}(X_1, \ldots, X_{j-1}, \mathsf{same}, X_{j+1}, \ldots, X_{k'})$ *and* $F_{\hat{C}_1}(X_1, \ldots, X_{j-1}, 0, X_{j+1}, \ldots, X_{k'})$ *is at most* 0.1.

This concludes the proof of the claim.

The following claim can be verified by inspection:

Claim 6.3.2. *The following two distributions $\mathcal{D}$, $\mathcal{D}'$ are equivalent:*

$\underline{\mathcal{D}}$:

1. *Choose r uniformly at random from $\{0,1\}^{(k'+1)\rho}$*
2. *Choose $\vec{x}$ uniformly at random from $\{0,1,\mathsf{same}\}^{k'}$*
3. *Choose $\mathcal{T} \subseteq [\widetilde{k}]$ uniformly at random from all subsets of $[\widetilde{k}]$ of size k'. Set $\mathcal{T} = \{z_1, \ldots, z_{k'}\}$, where the elements are in lexicographic order.*
4. *Choose j^* uniformly at random from $[k']$.*
5. *Choose ℓ^* uniformly at random from $[n]$*
6. *Output $(r, \vec{x}, \mathcal{T}, j^*, \ell)$.*

$\underline{\mathcal{D}'}$:

1. *Choose $\hat{r} := \hat{r}_1, \ldots, \hat{r}_{k'}, r_{k'+1}$ uniformly at random from $\{0,1\}^{(k'+1)\rho}$.*
2. *Choose $\hat{\vec{x}} := \hat{x}_1, \ldots, \hat{x}_{k'}$ uniformly at random from $\{0,1,\mathsf{same}\}^{k'}$.*
3. *Choose $\mathcal{T}^- := \{z^1 \ldots, z^{k'-1}\}$ uniformly at random from all subsets of $[k]$ of size $k'-1$.*
4. *Choose z^* uniformly at random from $[\widetilde{k}] \setminus \mathcal{T}^-$. Set $\mathcal{T} := \mathcal{T}^- \cup \{z^*\}$.*
5. *Choose ℓ^* uniformly at random from $[n]$.*
6. *Set j^* to the lexicographic index of element z^* in the set $\mathcal{T}$. Let $\{z_1, \ldots, z_{k'}\}$ denote the lexicographical ordering of $\mathcal{T}$ and note that $z_{j^*} := z^*$.*

7. *Set* $r := \hat{r}_{\pi(1)}, \ldots, \hat{r}_{\pi(k')}, r_{k'+1}$ *and* $\vec{x} := \hat{x}_{\pi(1)}, \ldots, \hat{x}_{\pi(k')}$*, where* $\pi(i) := j$ *iff* z^j *is the element with lexicographic index* i *in the set* $\mathcal{T}$*.*

8. *Output* $(r, \vec{x}, \mathcal{T}, j^*, \ell)$*.*

Again, the above distributions did not appear in the analysis in Section 6.2, and are added here since they will be used in the proof of Lemma 6.3.1.

6.3.4 The Attack

In this section we describe the polynomial-time attacker and updater. At a high-level, the attack is similar to the one from Section 6.2.5. However, the definition of the sunflower is slightly different, the attack employs the additional set $\mathcal{T}$*, and the parameters of the attack are different.*

Description of attacker:

- *For* $i \in [n]$*, find the sets* $\mathcal{R}_i$ *corresponding to the set of* c *blocks that is accessed with probability at least* $1/\hat{n}^c$ *by* D *and* $\mathtt{UP}$ *for position* i *and the corresponding random coins* $r_i^* \in \{0,1\}^\rho$*, where* $\rho := c \cdot \log(\hat{n}) = O(\log \lambda)$ *in polynomial time.*[3]

- *Find the Sunflower (with petals containing blocks from the set* $\mathcal{S}^{hi}$ *removed)* $\mathsf{SF}^- := \{S_{i_0, r_0^*}, \ldots, S_{i_{\widetilde{k}}, r_{i_{\widetilde{k}}}^*}\}$*, where* $i_0, \ldots, i_{\widetilde{k}} \in [n]$ *and* $\widetilde{k} + 1 := 8,640,000 \cdot c^2 \cdot$

[3]This can be done by, for each $i \in [n]$, taking a sufficiently large (but still polynomial) set of random coins $r \in \{0,1\}^\rho$, running D and $\mathtt{UP}$ for position i on an arbitrary codeword $\hat{C}$ for each r in the set, and selecting the access pattern, $\mathcal{R}_i$, that appears a sufficient number of times. By standard Chernoff bounds, a sufficiently good set, $\mathcal{R}_i$, will be selected with high probability.

$\mathcal{X}+1$, contained in $\{S_{1,r_1^*},\dots,S_{n,r_n^*}\}$ in polynomial time time.[4]

- *Choose a random subset $\mathcal{T} \subseteq [\widetilde{k}]$ of size k' and choose $j^* \sim [k']$.*
- *In the first round, submit leakage function $\ell(\hat{C})$ defined as $\ell(\hat{C}) := \mathsf{set}^{-}_{i_{z_j^*}, r^*_{z_{j^*}}}(\hat{C})$ which returns* Leaked*, i.e. the contents of the positions in $\hat{C}$ corresponding to decoding of $i_{z_j^*}$ with randomness $r^*_{z_j^*}$, minus the contents of the blocks in the core of the sunflower.*[5]
- *Wait until the $(k'+2)$-nd round. In the $(k'+2)$-nd round, choose tampering function f which replaces the contents of $\mathsf{set}^{-}_{i_{z_j^*}, r^*_{z_{j^*}}}(\hat{C}^{(k+1)+})$, i.e. the positions in $\hat{C}^{(k+1)+}$—where $\hat{C}^{(k+1)+}$ denotes the contents of the codeword immediately after the $(k+1)$-st update–corresponding to decoding of $i_{z_{j^*}}$ with randomness $r^*_{z_{j^*}}$, minus the contents of the blocks in the core of the sunflower, with the values,* Leaked*, that were leaked via ℓ.*

Description of Updater:

- *Choose $x_1,\dots,x_{k'} \sim \{0,1,\mathsf{same}\}^{k'}$.*
- *For $j=1$ to k':*
  - *If $x_j = \mathsf{same}$, request $\mathsf{UP}^{\hat{C}^{(j)}}(i_0, 0)$*

[4]As mentioned previously, the original inductive proof of the Sunflower lemma by Erdős and Rado yields an efficient (polynomial in $(n,c,\widetilde{k})$) algorithm for finding the sunflower.

[5]If the attacker may leak only a single bit per round, we instead add here $r < \mathcal{X}\cdot c$ number of rounds where in each round the attacker leaks a single bit from $\mathsf{set}^{-}_{i_{z_j^*}, r^*_{z_{j^*}}}(\hat{C})$. During each of these rounds, the updater requests a "dummy" update, $\mathsf{UP}^{\hat{C}^{(j)}}(i_0, 0)$. In order for the analysis to go through, we must increase the size of k so that the ratio of $\frac{(k'+1+c\cdot\mathcal{X})c}{k-k'} \le 0.2$ (in the current analysis we only need that $\frac{(k'+1)\cdot c}{\widetilde{k}-k'} \le 0.2$).

– *Otherwise request* $\mathsf{UP}^{\hat{C}^{(j)}}(i_{z_j}, x_j)$

where $\hat{C}^{(j)}$ *denotes the codeword immediately before the* j*-th update.*

- *In round* $k'+1$*, request* $\mathsf{UP}^{\hat{C}^{(k'+1)}}(i_0, 0)$.

6.3.5 Attack Analysis

We begin with some notation and basic facts. Let J^* *be the random variable corresponding to choice of* j^* *in the attack described above. Let* L^* *be a random variable chosen uniformly at random from* $[n] \setminus \mathcal{T}$*. For* $z_j \in \mathcal{T}$*, let* $\mathsf{UP}_{i_{z_j}}$ *be the event that location* i_{z_j} *gets updated and let* $\overline{\mathsf{UP}_{i_{z_j}}}$ *be the event that location* i_{z_j} *does not get updated. Recall that* $\vec{m} \in \{0^n, 1^n\}$ *denotes the original message. For* $j \in [n]$*,* m_j *denotes the original message in block* j*.* $m_j^{(t)}$ *denotes the decoding of* $\hat{C}^{(t)}$ *in the* j*-th position, where* $\hat{C}^{(t)}$ *denotes the codeword immediately before the* t*-th update.* $m_j^{(t)+}$ *denotes the decoding of* $\hat{C}^{(t)+}$ *in the* j*-th position, where* $\hat{C}^{(t)+}$ *denotes the codeword immediately after the* t*-th update. We have the following properties, which can be verified by inspection:*

Fact 6.3.1. *(a) For* $j \in [k']$*,* $\Pr[\mathsf{UP}_{i_{z_j}} \mid \vec{m} = \vec{0}] = \Pr[\mathsf{UP}_{i_{z_j}} \mid \vec{m} = \vec{1}] = 0.67$*;* $\Pr[\overline{\mathsf{UP}_{i_{z_j}}} \mid \vec{m} = \vec{0}] = \Pr[\overline{\mathsf{UP}_{i_{z_j}}} \mid \vec{m} = \vec{1}] = 0.33$.

(b) For $j \in [k']$*, if the original message was* $\vec{0}$*, then conditioned on an update occurring on block* i_{z_j}*,* $m_{i_{z_j}}^{(k'+1)} = 0$ *with probability* 0.5 *and* $m_{i_{z_j}}^{(k'+1)} = 1$ *with probability* 0.5*. Conditioned on no update occurring on block* i_{z_j}*,* $m_{i_{z_j}}^{(k'+1)} = 0$ *with probability* 1*.*

(c) *For $j \in [k']$, if the original message was $\vec{1}$, then conditioned on an update occurring on block i_{z_j}, $m_{i_{z_j}}^{(k'+1)} = 1$ with probability 0.5 and $m_{i_{z_j}}^{(k')+} = 0$ with probability 0.5. Conditioned on no update occurring on block i_{z_j}, $m_i^{(k'+1)} = 1$ with probability 1.*

(d) Sim *must output* same *for each position $i \in [n]$ in rounds $1, \ldots, k'+1$, since no tampering of the codeword occurs before round $k'+2$.*

(e) *For $j \in [k']$, $m_{i_{z_j}}^{(k'+1)} = m_{i_{z_j}}^{(k'+1)+}$, since in the $k'+1$-st round only the position i_0 gets updated.*

The following is the main technical lemma of this section. Note that the lemma differs significantly from the analogous lemma (Lemma 6.2.3) in Section 6.2.6. The lemma below considers the probability of the decoding of position i_{L^*} being equal to 0 or 1, conditioned on another event occurring in position $i_{z_{J^*}}$, *whereas Lemma 6.2.3 considered only the probability of the decoding of position i_{J^*} being equal to 0 or 1.*

Lemma 6.3.1. *For the attack and updater specified in Section 6.3.4, the probability that all three of the following events occur simultaneously is at least $1/\operatorname{poly}(\lambda)$, both when the original message was $\vec{m} = \vec{0}$ and when the original message was $\vec{m} = \vec{1}$: (a) position $i_{z_{J^*}}$ gets updated, (b) $m_{i_{z_{J^*}}}^{(k'+1)} = m_{i_{z_{J^*}}}^{(k'+1)+} \neq m_{i_{z_{J^*}}}^{(k'+2)}$ and (c) $m_{i_{z_{J^*}}}^{(k'+2)} \neq \perp$.*

Moreover, we have the following:

Case 1: *If the original message was $\vec{m} = \vec{0}$, then conditioned on (a), (b), and (c) occurring, we have that $m_{i_{L^*}}^{(k'+2)} = 0$ with probability at least 0.9.*

Case 2: *If the original message was $\vec{m} = \vec{1}$, then conditioned on (a), (b), and (c)* *occurring,* $m_{i_{L^*}}^{(k'+2)} = 1$ *with probability at least* 0.9.

We first show how to use Claim 6.3.1 to complete the proof of Theorem 6.3.1 and then present the proof of Claim 6.3.1.

Proof (of Theorem 6.2.1.)**.** *We show that the above claim implies that the candidate scheme is not secure under Definition 3.2.3 and Definition 3.2.4. Definition 3.2.4 requires the existence of a simulator* Sim *which (for the above attack and updater) outputs one of* $\{\mathsf{same}, \perp\} \cup \{0,1\}^\kappa$ *for the decoding of each position* $i \in [n]$ *in each round* $j \in [k'+2]$. *We denote by* $m_{i,\mathsf{Sim}}^{(j)}$ *the output of* Sim *in round* j *for position* i. *Recall that if* Sim *outputs* same *in round* j *for position* i*, then the output of the experiment in the corresponding position, denoted* $\widetilde{m}_{i,\mathsf{Sim}}^{(j)}$*, is set to* $\widetilde{m}_{i,\mathsf{Sim}}^{(j)} := m_i^{(j-1)+}$. *We begin by defining the following notation for each pair* $j \in [k']$, $\ell \in [n]$*:*

$$p_{up,j,\ell}^{1,0} := \Pr[m_{i_{z_j},\mathsf{Sim}}^{(k'+2)} = 1, m_{\ell,\mathsf{Sim}}^{(k'+2)} = 0 \mid \vec{m} = \vec{0} \wedge \mathsf{UP}_{i_{z_j}}]$$

$$p_{up,j,\ell}^{1,1} := \Pr[m_{i_{z_j},\mathsf{Sim}}^{(k'+2)} = 1, m_{\ell,\mathsf{Sim}}^{(k'+2)} = 0 \mid \vec{m} = \vec{1} \wedge \mathsf{UP}_{i_{z_j}}]$$

$$p_{up,j,\ell}^{0,0} := \Pr[m_{i_{z_j},\mathsf{Sim}}^{(k'+2)} = 0, m_{\ell,\mathsf{Sim}}^{(k'+2)} = 0 \mid \vec{m} = \vec{0} \wedge \mathsf{UP}_{i_{z_j}}]$$

$$p_{up,j,\ell}^{0,1} := \Pr[m_{i_{z_j},\mathsf{Sim}}^{(k'+2)} = 0, m_{\ell,\mathsf{Sim}}^{(k'+2)} = 0 \mid \vec{m} = \vec{1} \wedge \mathsf{UP}_{i_{z_j}}]$$

$$p'^{1,0}_{up,j} := \Pr[m_{i_{z_j},\mathsf{Sim}}^{(k'+2)} = 1 \mid \vec{m} = \vec{0} \wedge \mathsf{UP}_{i_{z_j}}]$$

$$p'^{1,1}_{up,j} := \Pr[m_{i_{z_j},\mathsf{Sim}}^{(k'+2)} = 1 \mid \vec{m} = \vec{1} \wedge \mathsf{UP}_{i_{z_j}}]$$

$$p'^{0,0}_{up,j} := \Pr[m^{(k'+2)}_{i_{z_j},\mathsf{Sim}} = 0 \mid \vec{m} = \vec{0} \wedge \mathsf{UP}_{i_{z_j}}]$$

$$p'^{0,1}_{up,j} := \Pr[m^{(k'+2)}_{i_{z_j},\mathsf{Sim}} = 1 \mid \vec{m} = \vec{1} \wedge \mathsf{UP}_{i_{z_j}}]$$

Note that since Sim *does not see the original message, we have that for each* $j \in [k']$, $\ell \in [n]$*:*

$$(a)\, p^{1,0}_{up,j,\ell} = p^{1,1}_{up,j,\ell} \qquad (b)\, p^{0,0}_{up,j,\ell} = p^{0,1}_{up,j,\ell} \qquad (c)\, p'^{1,0}_{up,j} = p'^{1,1}_{up,j} \qquad (d)\, p'^{0,0}_{up,j} = p'^{0,1}_{up,j}. \tag{6.3.1}$$

Since the output of the ideal experiment ($\mathbf{Ideal}_{\mathsf{Sim},\mathcal{U},\vec{0}}$*) is computationally indistinguishable from the output of the real experiment (*$\mathbf{CTamperLeak}_{\mathcal{A},\mathcal{U},\vec{0}}$*), if the original message* $\vec{m} = \vec{0}$*, then by Lemma 6.3.1, we must have:*

$$
\begin{aligned}
0.9 - \mathsf{negl}(\lambda) &\leq \Pr[\widetilde{m}^{(k'+2)}_{i_{L^*},\mathsf{Sim}} = 0 \mid \mathsf{UP}_{i_{z_{J^*}}} \wedge \widetilde{m}^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq \widetilde{m}^{(k'+1)}_{i_{z_{J^*}},\mathsf{Sim}} \\
&\qquad \wedge \widetilde{m}^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq \bot \wedge \vec{m} = \vec{0}] \\
&= \Pr[\widetilde{m}^{(k'+2)}_{i_{L^*},\mathsf{Sim}} = 0 \mid \mathsf{UP}_{i_{z_{J^*}}} \wedge \widetilde{m}^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq m^{(k'+1)}_{i_{z_{J^*}}} \\
&\qquad \wedge \widetilde{m}^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq \bot \wedge \vec{m} = \vec{0}] \qquad (6.3.2) \\
&= \Pr[\widetilde{m}^{(k'+2)}_{i_{L^*},\mathsf{Sim}} = 0 \mid \mathsf{UP}_{i_{z_{J^*}}} \wedge m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq m^{(k'+1)}_{i_{z_{J^*}}} \\
&\qquad \wedge m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \in \{0,1\} \wedge \vec{m} = \vec{0}] \qquad (6.3.3) \\
&= \Pr[m^{(k'+2)}_{i_{L^*},\mathsf{Sim}} = 0 \mid \mathsf{UP}_{i_{z_{J^*}}} \wedge m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq m^{(k'+1)}_{i_{z_{J^*}}} \\
&\qquad \wedge m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \in \{0,1\} \wedge \vec{m} = \vec{0}] \qquad (6.3.4) \\
&= \frac{\begin{array}{c}\Pr[m^{(k'+2)}_{i_{L^*},\mathsf{Sim}} = 0 \wedge m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq m^{(k'+1)}_{i_{z_{J^*}}} \\ \wedge\, m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \in \{0,1\} \mid \vec{m} = \vec{0} \wedge \mathsf{UP}_{i_{z_{J^*}}}]\end{array}}{\Pr[m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq m^{(k'+1)}_{i_{z_{J^*}}} \wedge m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \in \{0,1\} \mid \vec{m} = \vec{0} \wedge \mathsf{UP}_{i_{z_{J^*}}}]} \\
&= \frac{\sum_{j,\ell} \Pr[J^* = j \wedge L^* = \ell] \cdot 1/2 \cdot (p^{1,0}_{up,j,\ell} + p^{0,0}_{up,j,\ell})}{\sum_j \Pr[J^* = j] \cdot 1/2 \cdot (p'^{1,0}_{up,j} + p'^{0,0}_{up,j})} \qquad (6.3.5) \\
&= \frac{\sum_{j,\ell} \Pr[J^* = j \wedge L^* = \ell] \cdot (p^{1,0}_{up,j,\ell} + p^{0,0}_{up,j,\ell})}{\sum_j \Pr[J^* = j] \cdot (p'^{1,0}_{up,j} + p'^{0,0}_{up,j})} \qquad (6.3.6)
\end{aligned}
$$

Where (9) follows since by Fact 6.3.1, Sim *outputs* same *for each position in rounds* $1, \ldots, k'+1$ *and so* $\widetilde{m}^{(k'+1)}_{i_{z_{J^*}},\mathsf{Sim}} = m^{(k'+1)}_{i_{z_{J^*}}}$. *(10) follows since if* $(\widetilde{m}^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq m^{(k'+1)+}_{i_{z_{J^*}}} \wedge \widetilde{m}^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq \bot)$*, then* Sim *cannot output* same *or* $\bot$ *in position* $i_{z_{J^*}}$ *and so* Sim *must output* $m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} = \widetilde{m}^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \in \{0,1\}$. *(11) follows since if* Sim *does not output* same *or* $\bot$ *in position* $i_{z_{J^*}}$ *then* Sim *also cannot output* same *in position* i_{L^*} *(by definition of the ideal experiment* $\mathbf{Ideal}_{\mathsf{Sim},\mathcal{U},\vec{0}}$*). And (12) follows since the event that position* $i_{z_{J^*}}$ *gets updated to* 0 *(resp. updated to* 1*) occurs with probability* $1/2$, *conditioned on* $(\mathsf{UP}_{i_{z_{J^*}}} \wedge \vec{m} = \vec{0})$ *(see Fact 6.3.1) and is independent of the event that* Sim *outputs* 1 *(resp. outputs* 0*) in position* $i_{z_{J^*}}$.

Similarly, by Lemma 6.3.1 we also must have:

$$\begin{aligned}
0.1 + \mathsf{negl}(\lambda) \geq \Pr[&\widetilde{m}^{(k'+2)}_{i_{L^*},\mathsf{Sim}} = 0 \mid \mathsf{UP}_{i_{z_{J^*}}} \wedge \widetilde{m}^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq \widetilde{m}^{(k'+1)}_{i_{z_{J^*}},\mathsf{Sim}} \\
&\wedge \widetilde{m}^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq \bot \wedge \vec{m} = \vec{1}] \\
= \Pr[&\widetilde{m}^{(k'+2)}_{i_{L^*},\mathsf{Sim}} = 0 \mid \mathsf{UP}_{i_{z_{J^*}}} \wedge \widetilde{m}^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq m^{(k'+1)}_{i_{z_{J^*}}} \\
&\wedge \widetilde{m}^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq \bot \wedge \vec{m} = \vec{1}] \\
= \Pr[&\widetilde{m}^{(k'+2)}_{i_{L^*},\mathsf{Sim}} = 0 \mid \mathsf{UP}_{i_{z_{J^*}}} \wedge m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq m^{(k'+1)}_{i_{z_{J^*}}} \\
&\wedge m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \in \{0,1\} \wedge \vec{m} = \vec{1}] \\
= \Pr[&m^{(k'+2)}_{i_{L^*},\mathsf{Sim}} = 0 \mid \mathsf{UP}_{i_{z_{J^*}}} \wedge m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq m^{(k'+1)}_{i_{z_{J^*}}} \\
&\wedge m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \in \{0,1\} \wedge \vec{m} = \vec{1}] \\
= &\frac{\begin{array}{c}\Pr[m^{(k'+2)}_{i_{L^*},\mathsf{Sim}} = 0 \wedge m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq m^{(k'+1)}_{i_{z_{J^*}}} \\ \wedge m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \in \{0,1\} \mid \vec{m} = \vec{1} \wedge \mathsf{UP}_{i_{z_{J^*}}}]\end{array}}{\Pr[m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \neq m^{(k'+1)}_{i_{z_{J^*}}} \wedge m^{(k'+2)}_{i_{z_{J^*}},\mathsf{Sim}} \in \{0,1\} \mid \vec{m} = \vec{1}] \wedge \mathsf{UP}_{i_{z_{J^*}}}} \\
= &\frac{\sum_{j,\ell} \Pr[J^* = j \wedge L^* = \ell] \cdot 1/2 \cdot (p^{1,1}_{up,j,\ell} + p^{0,1}_{up,j,\ell})}{\sum_j \Pr[J^* = j] \cdot 1/2 \cdot (p'^{1,1}_{up,j} + p'^{0,1}_{up,j})} \\
= &\frac{\sum_{j,\ell} \Pr[J^* = j \wedge L^* = \ell] \cdot (p^{1,1}_{up,j,\ell} + p^{0,1}_{up,j,\ell})}{\sum_j \Pr[J^* = j] \cdot (p'^{1,1}_{up,j} + p'^{0,1}_{up,j})}
\end{aligned}$$

But by (6.3.1) we must have

$$\frac{\sum_{j,\ell} \Pr[J^* = j \wedge L^* = \ell] \cdot (p^{1,0}_{up,j,\ell} + p^{0,0}_{up,j,\ell})}{\sum_j \Pr[J^* = j] \cdot (p'^{1,0}_{up,j} + p'^{0,0}_{up,j})} = \frac{\sum_{j,\ell} \Pr[J^* = j \wedge L^* = \ell] \cdot (p^{1,1}_{up,j,\ell} + p^{0,1}_{up,j,\ell})}{\sum_j \Pr[J^* = j] \cdot (p'^{1,1}_{up,j} + p'^{0,1}_{up,j})},$$

thus leading to contradiction.

We conclude by proving the Lemma.

Proof (of Lemma 6.3.1)**.** *We prove that (a), (b), (c) occur with probability at least* $1/\operatorname{poly}(\lambda)$ *when* $\vec{m} = \vec{0}$ *and we prove that if the original message was* $\vec{m} = \vec{0}$*, then conditioned on (a), (b), (c) occurring, we have that* $m_{i_{L^*}}^{(k'+2)} = 0$ *with probability at least* 0.9*. The proof for the case* $\vec{m} = \vec{1}$ *is entirely analogous.*

Recall that random variables $\vec{X} := X_1, \ldots, X_{k'}$ *are i.i.d. variables distributed uniformly in* $\{0, 1, \mathsf{same}\}$*. We define the random variables* $\mathsf{CW}(r, \hat{C}_0, \mathcal{T}, \vec{X}, \mathsf{var})$, $\widetilde{\mathsf{CW}}(r, \hat{C}_0, \mathcal{T}, \vec{X}, \mathsf{var})$ *as follows, where* CW *represents the value of the codeword immediately before the tampering occurs in round* $k'+2$, $\widetilde{\mathsf{CW}}$ *represents the value of the codeword immediately after the tampering occurs in round* $k'+2$*.*

```
$\underline{\mathsf{CW}(r, \hat{C}_0, \mathcal{T}, \vec{X}, j^*, \mathsf{var})}$:

  1. Parse $r = r_1, \ldots, r_{k'+1}$
  2. Set $X_{j^*} := \mathsf{var}$, where $\mathsf{var} \in \{\mathsf{same}, 1\}$.
  3. For $j = 1$ to $k'$:
       • If $X_j = \mathsf{same}$, run $\mathsf{UP}^{\hat{C}_0^{(j)}}(i_0, 0; r_j)$.
       • Otherwise run $\mathsf{UP}^{\hat{C}_0^{(j)}}(i_j, X_j; r_j)$.
     where $\hat{C}_0^{(j)}$ denotes the codeword immediately before the $j$-th update.
  4. Run $\mathsf{UP}^{\hat{C}_0^{(k'+1)}}(i_0, 0; r_{k'+1})$.
  5. Output the contents of $\hat{C}_0^{(k'+1)+}$, where $\hat{C}_0^{(k'+1)+}$ denotes the codeword immediately after the $k'+1$-st update.
```

```
$\widetilde{\mathsf{CW}}(r, \hat{C}_0, \mathcal{T}, \vec{X}, j^*, \mathsf{var})$:

  1. Set $y^0_{j^*} := \mathsf{set}^-_{i_{j^*}, r^*_{j^*}}(\hat{C}_0)$
  2. Parse $r = r_1, \ldots, r_{k'+1}$
  3. Set $X_{j^*} := \mathsf{var}$, where $\mathsf{var} \in \{\mathsf{same}, 1\}$.
  4. For $j = 1$ to $k'$:
       • If $X_j = \mathsf{same}$, run $\mathsf{UP}^{\hat{C}_0^{(j)}}(i_0, 0; r_j)$.
       • Otherwise run $\mathsf{UP}^{\hat{C}_0^{(j)}}(i_j, X_j; r_j)$.
     where $\hat{C}_0^j$ denotes the codeword immediately before the $j$-th update.
  5. Replace the current contents of $\mathsf{set}^-_{i_{j^*}, r^*_{j^*}}(\hat{C}_0^{(k'+)})$ with $y^0_{j^*}$, where $\hat{C}_0^{(k'+)}$ denotes
     the codeword immediately after the $k'$-th update, yielding codeword $\hat{C}_0^{(k'+1)}$.
  6. Run $\mathsf{UP}^{\hat{C}_0^{(k'+1)}}(i_0, 0; r_{k'+1})$.
  7. Output the contents of $\hat{C}_0^{(k'+2)}$, where $\hat{C}_0^{(k'+2)}$ denotes the codeword immediately
     before the $k'+2$-st update.
```

Note that for any setting of $\vec{X} := X_1, \ldots, X_{k'}$ and any setting of var,

$$\mathsf{core}(\mathsf{CW}(r, \hat{C}, \mathcal{T}, \vec{X}, j^*, \mathsf{var})) \equiv F_{\hat{C}_0, \mathcal{T}, r}(X_1, \ldots, X_{j^*-1}, \mathsf{var}, X_{j^*+1}, \ldots, X_{k'}) \quad (6.3.7)$$

and

$$\mathsf{core}(\widetilde{\mathsf{CW}}(r, \hat{C}_0, \mathcal{T}, \vec{X}, j^*, \mathsf{var})) \equiv \mathsf{core}(\mathsf{CW}(r, \hat{C}_0, \mathcal{T}, \vec{X}, j^*, \mathsf{var})). \quad (6.3.8)$$

To prove Lemma 6.3.1, we show that (1) with probability at least, $1/5\hat{n}^c = 1/\operatorname{poly}(\lambda)$ *it is the case that position* $i_{z_{j^*}}$ *is updated to* 1 *in round* $j^* \in [k']$ *(and so decodes to* 1 *in round* $k'+1$*) and decodes to* 0 *in round* $k'+2$*; (2) regardless of the decoding of position* $i_{z_{j^*}}$*, position* ℓ^* *decodes to* 0 *with probability at least* 0.9 *in round* $k'+2$.

Showing that w.h.p., position $i_{z_{j^*}}$ is updated to 1 but decodes to 0 after tampering *The proof relies on the fact that, when decoding the tampered codeword, assuming position* $i_{z^*_j}$ *has been updated to* 1 *during the first* k' *rounds, decode for position* $i_{z^*_j}$ *takes the following as input: (a) randomness* r'*;*
(b) $\mathsf{set}^-_{i_{j^*},r'}(\widetilde{\mathsf{CW}}(r,\hat{C}_0,\mathcal{T},\vec{X},j^*,1))$*; (c)* $\mathsf{core}(\widetilde{\mathsf{CW}}(r,\hat{C}_0,\mathcal{T},\vec{X},j^*,1))$.

Recall that $y^0_{j^*} := \mathsf{set}^-_{i_{z_{j^*}},r^*_{z_{j^*}}}(\hat{C}_0)$*, First, note that due to correctness of* **DEC***,*

$$\Pr[\mathsf{D}(\mathsf{set}^-_{i_{z_{j^*}},r^*_{z_{j^*}}}(\mathsf{CW}(r,\hat{C}_0,\mathcal{T},\vec{X},j^*,\mathsf{same}),\mathsf{core}(\mathsf{CW}(r,\mathcal{T},\hat{C}_0,\vec{X},j^*,\mathsf{same}))) = 0] = 1, \tag{6.3.9}$$

Since in experiment $\mathsf{CW}(r,\hat{C}_0,\mathcal{T},\vec{X}),j^*,\mathsf{same})$*, the original message is* $\vec{m} := \vec{0}$*, no update occurs to position* $i_{z_{j^*}}$ *(since* X_{j^*} *is set to* same*) and no tampering occurs.*

We next consider a sequence of hybrid distributions and show that consecutive hybrids are statistically close:

<u>Hybrid 0</u>

$$\left(\mathsf{set}^-_{i_{z_{j^*}},r^*_{z_{j^*}}}\left(\mathsf{CW}\left(r,\hat{C}_0,\mathcal{T},\vec{X},j^*,\mathsf{same}\right)\right),\mathsf{core}\left(\mathsf{CW}\left(r,\mathcal{T},\hat{C}_0,\vec{X},j^*,\mathsf{same}\right)\right)\right)$$

where r is chosen at random from $\{0,1\}^{(k'+1)\rho}$, $\mathcal{T}$ is a set of size k' chosen at random from $[k]$, j^* is chosen at random from $[k']$ and $\vec{X}$ is chosen at random from $\{0,1,\mathsf{same}\}^{k'}$.

Hybrid 1

$$\left(\mathsf{set}^{-}_{i_{z_{j^*}}, r^*_{z_{j^*}}}\left(\mathsf{CW}\left(r, \hat{C}_0, \mathcal{T}, \vec{x}\right), j^*, \mathsf{same}\right), \mathsf{core}\left(\mathsf{CW}\left(r, \mathcal{T}, \hat{C}_0, \vec{X}, \mathsf{same}\right)\right)\right)$$

where

- $\hat{r} := \hat{r}_1, \ldots, \hat{r}_{k'}, r_{k'+1}$ *is chosen uniformly at random from* $\{0,1\}^{(k'+1)\rho}$.
- $\hat{\vec{x}} := \hat{x}_1, \ldots, \hat{x}_{k'}$ *is chosen uniformly at random from* $\{0,1,\mathsf{same}\}^{k'}$.
- $\mathcal{T}^- := \{z^1 \ldots, z^{k'-1}\}$ *is chosen uniformly at random from all subsets of* $[k]$ *of size* $k'-1$.
- z^* *is chosen at random from* $[k] \setminus \mathcal{T}^-$ *and* $\mathcal{T} := \mathcal{T}^- \cup \{z^*\}$.
- ℓ^* *is chosen at random from* $[n]$.

We then set j^* to the lexicographic index of element z^* in the set $\mathcal{T}$, set $r := \hat{r}_{\pi(1)}, \ldots, \hat{r}_{\pi(k')}, r_{k'+1}$, and set $\vec{x} := \hat{x}_{\pi(1)}, \ldots, \hat{x}_{\pi(k')}$, where $\pi(i) := j$ iff z^j is the element with lexicographic index i in the set $\mathcal{T}$.

Claim 6.3.3. *Hybrid 0 and Hybrid 1 are identical.*

This follows immediately from Claim 6.3.2.

Hybrid 2

$$\left(y^0_{j^*}, \mathsf{core}\left(\mathsf{CW}\left(r, \mathcal{T}, \hat{C}_0, \vec{x}, \mathsf{same}\right)\right)\right)$$

where

1. $\hat{r} := \hat{r}_1, \ldots, \hat{r}_{k'}, r_{k'+1}$ *is chosen uniformly at random from* $\{0,1\}^{(k'+1)\rho}$.

2. $\hat{\vec{x}} := \hat{x}_1, \ldots, \hat{x}_{k'}$ *is chosen uniformly at random from* $\{0, 1, \mathsf{same}\}^{k'}$.

3. $\mathcal{T}^- := \{z^1 \ldots, z^{k'-1}\}$ *is chosen uniformly at random from all subsets of* $[k]$ *of size* $k' - 1$.

4. z^* *is chosen at random from* $[k] \setminus \mathcal{T}^-$ *and* $\mathcal{T} := \mathcal{T}^- \cup \{z^*\}$.

5. ℓ^* *is chosen at random from* $[n]$.

We then set j^* *to the lexicographic index of element* z^* *in the set* $\mathcal{T}$, *set* $z^{k'} := z_{j^*} := z^*$, *set* $r := \hat{r}_{\pi(1)}, \ldots, \hat{r}_{\pi(k')}, r_{k'+1}$, *and set* $\vec{x} := \hat{x}_{\pi(1)}, \ldots, \hat{x}_{\pi(k')}$, *where* $\pi(i) := j$ *iff* z^j *is the element with lexicographic index* i *in the set* $\mathcal{T}$.

Claim 6.3.4. *Hybrid* 1 *and Hybrid* 2 *are* 0.8*-close.*

Proof. *We show that for every setting of* $\hat{\vec{x}}, \hat{r}$, *with probability* 0.8 *over choice of* $\mathcal{T}, j^*$

$$\mathsf{set}^-_{i_{z_{j^*}}, r^*_{z_{j^*}}}(\mathsf{CW}(r, \hat{C}_0, \mathcal{T}, j^*, \vec{X}, \mathsf{same})) = y^0_{j^*}, \tag{6.3.10}$$

Note that (6.3.10) holds as long as none of the updates made during experiment $\widetilde{\mathsf{CW}}(r, \hat{C}_0, \mathcal{T}, \vec{X})$ *modify blocks in* $S^-_{i_{z_{j^*}}, r^*_{z_{j^*}}}$. *To show that this is indeed the case, observe that the updates corresponding to positions* $i_0, i_{z^1}, \ldots, i_{z^{k'-1}}$ *touch at most*

$(k'+1)\cdot c$ positions outside the core, core*, and that these positions are completely determined immediately after Step 3 in the sampling procedure above (i.e. immediately after $\hat{r}, \hat{\vec{x}}$ and $\mathcal{T}^-$ are sampled). This is due to the fact that the position corresponding to $i_{z^*} = i_{z_{j^*}}$ will never be updated since* $\mathsf{var} := \mathsf{same}$ *in experiment* $\mathsf{CW}(r, \mathcal{T}, \hat{C}_0, \vec{x}, j^*, \mathsf{same})$ *and the fact that for $j \in [k']$, the accessed positions corresponding to i_{z^j} depend only on $\hat{r}_j$, $\hat{x}_j$, but not on the contents of the codeword. Additionally all of the accessed positions corresponding to i_0 also depend only on $\hat{r}_j$, $\hat{x}_j$, but not on the contents of the codeword. Now, due to the structure of the Sunflower, these positions can modify at most $(k'+1)\cdot c$ petals out of the k petals in the Sunflower. Thus, the probability that z^* chosen at random from $[\widetilde{k}] \setminus \mathcal{T}^-$ in Step 4 hits one of these petals is at most*

$$\frac{(k'+1)\cdot c}{\widetilde{k} - k'} \leq \frac{2\cdot k'\cdot c}{\widetilde{k} - k'\cdot c} \leq \frac{2\cdot 1,440,000\cdot c^2\cdot \mathcal{X}}{14,400,000\cdot c^2\cdot \mathcal{X}} = 0.2.$$

Therefore, with probability 0.8, the blocks in the set $S_{z^*, r^*_{z^*}} = S_{z_{j^*}, r^*_{z_{j^*}}}$ have not been modified at all, which means that

$\mathsf{set}_{i_{z^*}, r^*_{z^*}}(\mathsf{CW}(r, \hat{C}_1, \mathcal{T}, j^*, \vec{X})) = \mathsf{set}_{i_{z_{j^*}}, r^*_{i_{z_{j^*}}}}(\mathsf{CW}(r, \hat{C}_1, \mathcal{T}, j^*, \vec{X})) = y^0_{j^*}.$

Hybrid 3

$$\left(y^0_{j^*}, \mathsf{core}\left(\mathsf{CW}\left(r, \mathcal{T}, \hat{C}_0, \vec{X}, j^*, \mathsf{same}\right)\right)\right)$$

where r is chosen at random from $\{0,1\}^{(k'+1)\rho}$, $\mathcal{T}$ is a set of size k' chosen at random from $[k]$, j^* is chosen at random from $[k']$ and $\vec{X}$ is chosen at random from $\{0, 1, \mathsf{same}\}^{k'}$.

Claim 6.3.5. *Hybrid 2 and Hybrid 3 are identical.*

This follows again by Claim 6.3.2.

Hybrid 4

$$\left(y^0_{j^*}, F_{\hat{C}_0,r,\mathcal{T}}\left(X_1,\ldots,X_{j^*-1},\mathsf{same},X_{j^*+1},X_{k'}\right)\right)$$

where r is chosen at random from $\{0,1\}^{(k'+1)\rho}$, $\mathcal{T}$ is a set of size k' chosen at random from $[k]$, j^* is chosen at random from $[k']$ and $X_1,\ldots,X_{j^*-1},\mathsf{same},X_{j^*+1},X_{k'}$ are chosen at random from $\{0,1,\mathsf{same}\}^{k'}$.

Claim 6.3.6. *Hybrid 3 and Hybrid 4 are identical.*

This follows from Equation (6.3.7).

Hybrid 5

$$\left(y^0_{j^*}, F_{\hat{C}_0,\mathcal{T},r}\left(X_1,\ldots,X_{j^*-1},1,X_{j^*+1},X_{k'}\right)\right)$$

where r is chosen at random from $\{0,1\}^{(k'+1)\rho}$, $\mathcal{T}$ is a set of size k' chosen at random from $[k]$, j^* is chosen at random from $[k']$ and $X_1,\ldots,X_{j^*-1},\mathsf{same},X_{j^*+1},X_{k'}$ are chosen at random from $\{0,1,\mathsf{same}\}^{k'}$.

Claim 6.3.7. *Hybrid 4 and Hybrid 5 are 0.8-close.*

Proof. *We argue that with probability at least 0.9 over $j^* \sim [k]$, the statistical distance between Hybrid 5 and Hybrid 6 is at most 0.1. This follows from Claim 6.2.1 and the fact that $y^0_{j^*}$ is fully determined by $\hat{C}_0$, which is part of the description of the compression function $F_{\hat{C}_0,\mathcal{T},r}$.*

Hybrid 6

$$\left(y^0_{j^*}, \mathsf{core}\left(\widetilde{\mathsf{CW}}\left(r, \hat{C}_0, \mathcal{T}, j^*, \vec{X}, 1\right)\right)\right)$$

where r is chosen at random from $\{0,1\}^{(k'+1)\rho}$, $\mathcal{T}$ is a set of size k' chosen at random from $[k]$, j^* is chosen at random from $[k']$ and $\vec{X}$ is chosen at random from $\{0, 1, \mathsf{same}\}^{k'}$.

Claim 6.3.8. *Hybrid 5 and Hybrid 6 are identical.*

This follows from (6.3.7) and (6.3.8).

Hybrid 7

$$\left(\mathsf{set}^-_{i_{j^*}, r^*_{j^*}}\left(\widetilde{\mathsf{CW}}\left(r, \hat{C}_0, \mathcal{T}, \vec{X}, j^*, 1\right)\right), \mathsf{core}\left(\widetilde{\mathsf{CW}}\left(r, \mathcal{T}, \hat{C}_0, \vec{X}, j^*, 1\right)\right)\right)$$

where r is chosen at random from $\{0,1\}^{(k'+1)\rho}$, $\mathcal{T}$ is a set of size k' chosen at random from $[k]$, j^* is chosen at random from $[k']$ and $\vec{X}$ is chosen at random from $\{0, 1, \mathsf{same}\}^{k'}$.

Claim 6.3.9. *Hybrid 8 and Hybrid 9 are identical.*

This follows from the definition of $\widetilde{\mathsf{CW}}(r, \hat{C}_0, \mathcal{T}, \vec{X}, j^*, 1)$.

Now, conditioned on $r' = r^*_j$, the view of the decode algorithm in position j^* (assuming position j^* got updated to 1) will be exactly

$$\left(\mathsf{set}^-_{i_{j^*}, r^*_{j^*}}\left(\widetilde{\mathsf{CW}}\left(r, \hat{C}_0, \mathcal{T}, \vec{X}, j^*, 1\right)\right), \mathsf{core}\left(\widetilde{\mathsf{CW}}\left(r, \mathcal{T}, \hat{C}_0, \vec{X}, j^*, 1\right)\right)\right).$$

Thus, combining (6.3.13) and Claims 6.3.3, 6.3.4, 6.3.5, 6.3.6, 6.3.7, 6.3.8, *6.3.9 we have that, conditioned on $r' = r^*_{j^*}$ and $X_{j^*} = 1$, the D algorithm outputs 0 for position $i_{z_{j^*}}$ in round $k'+1$ with probability at least 0.6. Thus, the probability that position j^* is updated to 1 in round $j^* \in [k']$ (and so decodes to 1 in round k') and decodes to 0 in round $k'+1$ is*

$$0.6 \cdot \Pr[r' = r^*_{J^*} \wedge X_{J^*} = 1] = 0.6 \cdot 1/\hat{n}^3 \cdot 1/3 = \frac{1}{5\hat{n}^c}.$$

Showing that w.h.p. position ℓ^* decodes to 0 in round $k'+2$ *Let r'' denote the randomness of the decode for position ℓ^* in round $k'+2$. First, as long as $\ell^* \notin \mathcal{T} \cup \{i_0\}$ (which occurs with probability at most $\frac{|\mathcal{T}|+1}{n} = \frac{k'+1}{n} \leq .001$, for sufficiently large n), by correctness we have that:*

$$\Pr\left[\mathsf{D}\left(\mathsf{set}_{i_{\ell^*},r''}\left(\mathsf{CW}\left(r, \hat{C}_0, \mathcal{T}, \vec{X}, j^*, 1\right)\right)\right) = 0\right] = 1. \tag{6.3.11}$$

We want to show that:

$$\Pr\left[\mathsf{D}\left(\mathsf{set}_{i_{\ell^*},r''}\left(\widetilde{\mathsf{CW}}\left(r, \hat{C}_0, \mathcal{T}, \vec{X}, j^*, 1\right)\right)\right) = 0\right] \geq 0.9. \tag{6.3.12}$$

Note that indeed,

$$\left(\mathsf{set}_{i_{\ell^*},r''}\left(\mathsf{CW}\left(r, \hat{C}_0, \mathcal{T}, \vec{X}, j^*, 1\right)\right)\right) \equiv \left(\mathsf{set}_{i_{\ell^*},r''}\left(\widetilde{\mathsf{CW}}\left(r, \hat{C}_0, \mathcal{T}, \vec{X}, j^*, 1\right)\right)\right),$$

(where r'' is chosen uniformly at random, r is chosen uniformly at random, $\mathcal{T}$ is chosen at random and $j^* \sim [k]$), as long as (1) $\ell^* \notin \mathcal{T} \cup \{i_0\}$ and (2) the set of tam-

pered blocks, $S^{-}_{j^*,r^*_{j^*}}$, does not intersect with $S_{\ell^*,r''}$. We have already upperbounded (1) by .001. To upperbound (2), fix $\mathcal{T}, j^*$ and thus the set $S^{-}_{j^*,r^*_{j^*}}$. Since, by construction, $S^{-}_{j^*,r^*_{j^*}} \cap \mathcal{S}^{hi} = \emptyset$, we have that the probability over random choice of $\ell^* \in [n]$ and random choice of r'' that $S_{\ell^*,r''}$ accesses any of the at most c blocks in $S^{-}_{j^*,r^*_{j^*}}$ is at most $c \cdot \frac{1}{11c} \leq 0.091$, then, by a union bound, we have that

$$\Pr\left[\mathsf{D}\left(\mathsf{set}^{-}_{i_{\ell^*},r''}\left(\widetilde{\mathsf{CW}}\left(r, \hat{C}_0, \mathcal{T}, \vec{X}, j^*, 1\right)\right)\right) = 0\right] \geq 0.9. \tag{6.3.13}$$

6.4 Matching Upper Bound

As discussed earlier in Section 6.1.2.4, the locally decodable and updatable NMC of [54] is constructed from a symmetric encryption scheme $\mathcal{E} = (\mathsf{Gen}, \mathsf{Encrypt}, \mathsf{Decrypt})$, a standard non-malleable code $\mathsf{NMC} = (\mathsf{E}', \mathsf{D}')$, and a collision resistant hash function family H. We also stated that the construction of [54] provides computational security against polynomial size tampering functions $f = (f_1, f_2)$ such that $f_1 \in \mathcal{F}$ tampers with c independently, where the underlying non-local NMC $(\mathsf{E}', \mathsf{D}')$ is secure against $\mathcal{F}$. Whereas f_2 tampers with rest of the codeword arbitrarily. In fact, the locally decodable and updatable non-malleable code of [54] provides computational security against a slightly more general tamper class. Specifically, they achieve computational tamper resilience against the tampering class $\bar{\mathcal{F}}$ where $f \in \bar{\mathcal{F}}$ satisfies the following: $f = (f_1, f_2)$ where f_1 gets access to the entire codeword but it is required that f_1 restricted in the first block (corresponding to c) is in tampering class $\mathcal{F}$. As in the previous case f_2 can be arbitrary poly-size tampering *function which tampers with the encryptions and Merkle tree (and does not get access to c). In this work we achieve tamper resilience against the same tampering class as [54], for the formal definition of $\bar{\mathcal{F}}$ refer to Theorem 6.4.2.*

We now show how to construct a locally updatable and decodable non-malleable code with super-constant locality. This is achieved by replacing the Merkle Tree in the construction presented in [54] by a new data structure, t-slice Merkle Tree *which we defined below (see Definition 6.4.1). Intuitively, the locality of updating/decoding in the construction given by Dachman-Soled et al. [54] is lower-bounded by the depth of the Merkle Tree, since, in order to detect tampering, each update/decode instruction must check the consistency of a leaf by traversing the path from leaf to root. Our initial idea is to replace the binary Merkle Tree of depth* $\log(n)$ *with a t-ary Merkle tree (where t is a super-constant function of n defined below) of constant depth. Unfortunately, this simple solution does not quite work. Recall that in order to verify consistency of a leaf in a standard Merkle tree, one needs to access not only the path from leaf to root, but also the* siblings *of each node on the path. This would mean that in the t-ary tree, we would need to access at least* $\Omega(t)$ *sibling nodes, where t is super-constant, thus still requiring super-constant locality. Our solution, therefore, is to construct t-ary Merkle trees of a particular form, where verifying consistency of a leaf can be done by traversing only the path from leaf to root,* without *accessing any sibling nodes. We call such trees t-slice Merkle trees. Details of the construction follow in Definition 6.4.1, and Algorithms 1, 2, 3 and 4. Finally, in Theorem 6.4.1 we show that the t-slice Merkle Tree is collision resistant, which allows us to retain security while replacing the Merkle tree in the construction of [54] with our t-slice*

Merkle Tree. This then leads to our matching upper bound in Theorem 6.4.2.

Definition 6.4.1 (t-slice Merkle Tree)**.** *Let $\mathcal{X}, t \in \mathbb{N}$ and let $\mathcal{X}$ be a multiple of t. Let $h : \{0,1\}^{\mathcal{X}} \to \{0,1\}^{\mathcal{X}/t}$ be a hash function that maps a block of size $\mathcal{X}$ to block of size $\mathcal{X}/t$. Denote a block of data at level j with index i by α_i^j and the input data by $M = (m_1, m_2, \ldots, m_n) \in \{0,1\}^{\mathcal{X} \cdot n}$. Let $\alpha_i^0 := m_{i+1}$ for $0 \leq i \leq n-1$. A t-slice Merkle Tree $\mathsf{Tree}_h^t(M)$ is defined recursively in the following way:*

- *Bottom layer of the tree contains n blocks of data each of size $\mathcal{X}$, i.e., $(\alpha_0^0, \alpha_1^0, \ldots, \alpha_{n-1}^0)$.*
- *To compute the content of non-leaf node at level j with index i set $\alpha_i^j := h(\alpha_{i \cdot t}^{j-1}) \mid\mid \ldots \mid\mid h(\alpha_{((i+1) \cdot t)-1}^{j-1})$.*
- *Once a single block α_0^j remains, set the root of Merkle Tree $\mathsf{Root}_h{}_h^t(M) := \alpha_0^j$ and the height of tree $\mathcal{H} := j+1$ and terminate.*

The internal blocks of Merkle Tree (including the root) are denoted as $\mathsf{Tree}_h^t(M)$.

Definition 6.4.2 (k^{th}-slice)**.** *Let α_i^j be a block in Merkle Tree Tree_h^t. We divide α_i^j into t parts of size $\frac{\chi}{t}$, and call these individual smaller parts as* slices*. Then, for $k \in \{0, 1, \ldots, t-1\}$ the k^{th}-slice, denoted by $\alpha_i^j[k]$ is the* slice *consisting $(\frac{k \cdot \chi}{t}, \ldots, \frac{(k+1) \cdot \chi}{t})$*

.

For reference, we provide a pictorial representation of a t-slice Merkle Tree in Figure 6.2.

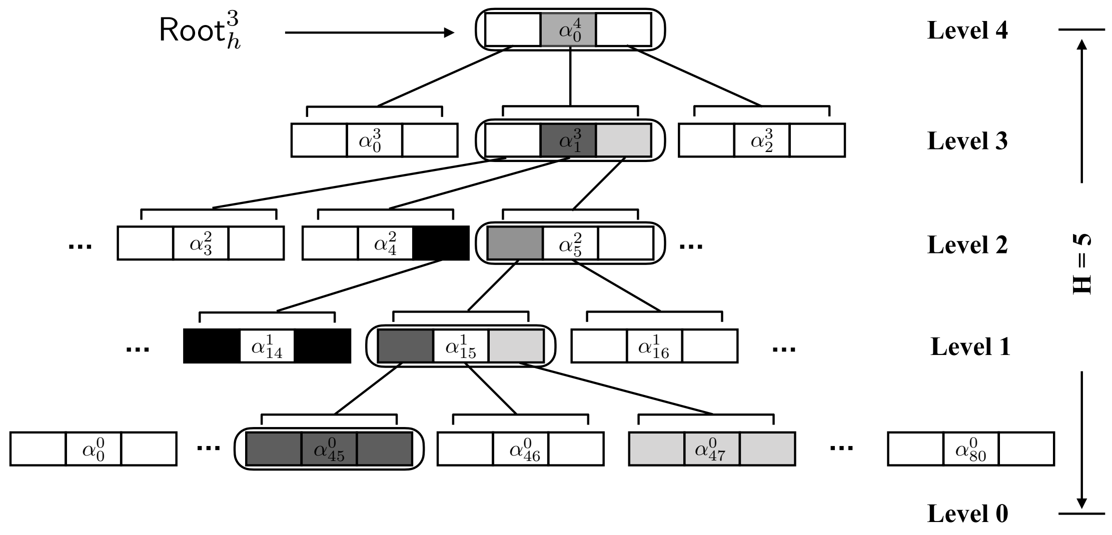

Figure 6.2: Illustration of 3-slice-Merkle Tree: The figure shows a 3-slice Merkle tree with $n = 81$ number of blocks, where each block has size $\mathcal{X}$. The hash function h maps inputs of size $\mathcal{X}$ to outputs of size $\mathcal{X}/3$. As a concrete example, starting from leaf block α_{45}^0, we apply the hash h to obtain the leftmost (0-th) slice of block α_{15}^1. Next, hashing α_{15}^1, we obtain the leftmost (0-th) slice of block α_5^2. Then, hashing α_5^2, we obtain the rightmost (2-nd) slice of block α_1^3. Finally, hashing α_1^3, we obtain the middle (1-st) slice of block α_0^4. Thus, in order to check consistency or update leaf block 45 at the bottom, we need to access only the path to block 45, consisting of $(\alpha_{45}^0, \alpha_{15}^1, \alpha_5^2, \alpha_1^3, \alpha_0^4)$, as described above. Note that no sibling nodes are accessed.

Lemma 6.4.1. *Let $\mathcal{X} \in \Omega(\lambda^{1/\mu})$, $h : \{0,1\}^{\mathcal{X}} \to \{0,1\}^{\mathcal{X}^{\mu}}$, and $t := \mathcal{X}^{1-\mu}$, for constant $0 < \mu < 1$. Assuming $n = \mathrm{poly}(\lambda) := \mathcal{X}^c$ for constant c, the height of the t-slice Merkle Tree will be constant $\mathcal{H} = \frac{c}{1-\mu} + 1$.*

Proof. *In the beginning the message blocks $M = (m_1, m_2, \ldots, m_n)$ are at the leaves of the tree and size of each block is $\mathcal{X}$, i.e. $|m_i| = \mathcal{X}$. After applying a hash function to each of the blocks separately, their size becomes $\mathcal{X}^{\mu}$ and by concatenating $\mathcal{X}^{1-\mu}$ number of hashes a single block of size $\mathcal{X}$ will be formed. In this level there will therefore be $\frac{\mathcal{X}^c}{\mathcal{X}^{1-\mu}} = \mathcal{X}^{c+\mu-1}$ block of size $\mathcal{X}$. Applying hash function to each of them will form new blocks of size $\mathcal{X}^{\mu}$ and there will be $\mathcal{X}^{c+2\mu-2}$ blocks of size $\mathcal{X}$. In general in level i-th there will be $\mathcal{X}^{c+i\mu-i}$ blocks of size $\mathcal{X}$. The root of the t-slice Merkle Tree is of size $\mathcal{X}$ which is just a single block, so the height of the tree is for the case where $\mathcal{X}^{c+i\mu-i} = 1\,(= \mathcal{X}^0)$ resulting in i as $\frac{c}{1-\mu}$. Since i starts from 0 the total height of the tree is given by $\mathcal{H} = \frac{c}{1-\mu} + 1$.*

We next present various algorithms which are used to manipulate the t-slice merkle tree data structure. Specifically, Parent *(see Algorithm 1), returns the parent of a node,* Path *(see Algorithm 2), returns the path from a leaf to the root,* consistencyCheck *(see Algorithm 3), checks the consistence of a path with a given root, and* Update *(see Algorithm 4), updates a leaf of the tree and the corresponding path. It may be helpful for the reader to refer to Figure 6.2 alongside the following algorithms.*

Algorithm 1 Parent($\mathsf{Tree}_h^t, \mathcal{H}, \mathsf{node}$): Returns the parent of the block node in Tree_h^t of height $\mathcal{H}$

```
1: procedure PARENT($\mathsf{Tree}_h^t, \mathcal{H}, \alpha_i^j$)
2:     if $j = \mathcal{H}$ then
3:         print " $\mathsf{Root}_{hh}^{t}$ does not have any parent. "
4:         return NULL
5:     else
6:         $\hat{j} := j + 1$
7:         $\hat{i} := \lfloor \frac{i}{t} \rfloor$        ▷ $\lfloor k \rfloor$ denotes greatest integer less than or equal to $k$.
8:         return $\alpha_{\hat{i}}^{\hat{j}}$
9:     end if
10: end procedure
```

Algorithm 2 Path($\mathsf{Tree}_h^t, \mathcal{H}, \mathsf{node}$): Returns the path to the root of the tree (Root_{hh}^{t}) from the block node as a list of blocks starting at node.

```
1: procedure PATH($\mathsf{Tree}_h^t, \mathcal{H}, \alpha_i^j$)
2:     path := $\phi$        ▷ Initialize an empty list to store the path
3:     $currentNode := \alpha_i^j$

4:     while $currentNode \neq NULL$ do
5:         path.$append(currentNode)$        ▷ Add current block to the path.
6:         $currentNode := PARENT(\mathsf{Tree}_h^t, \mathcal{H}, currentNode)$
7:     end while

8:     return path
9: end procedure
```

Algorithm 3 consistencyCheck($\mathsf{path}, \mathsf{Root}_{hh}^{t}$): Returns boolean value TRUE if given path of nodes in Merkle Tree Tree_h^t is consistent with its root Root_{hh}^{t}, and returns $FALSE$ otherwise.

```
1: procedure consistencyCheck($\mathsf{path}, \mathsf{Root}_{hh}^{t}$)
2:     $\ell := length(\mathsf{path})$   ▷ function $length(\cdot)$ returns the total number of elements in the list.
3:     $valid = \mathsf{TRUE}$   ▷ Initializing the boolean flag to be returned.
4:     for $j = 0$ to $\ell - 2$ do
5:         $\alpha_i^{j+1} := \mathsf{path}_{j+1}$   ▷ Here $\mathsf{path}_j$ is $j^{\text{th}}$ element of the list.
6:         $\alpha_i^{j} := \mathsf{path}_{j}$
7:         $k := i \bmod t$
8:         if $\alpha_i^{j+1}[k] == h(\alpha_i^j)$ then
9:             $valid = valid$ AND TRUE
10:        else
11:            $valid = \mathsf{FALSE}$
12:            break
13:        end if
14:    end for
15:    if $\mathsf{Root}_{hh}^{t} \neq \mathsf{path}_{\ell-1}$ then
16:        $valid = \mathsf{FALSE}$
17:    else
18:        $valid = valid$ AND TRUE
19:    end if

20:    return $valid$
21: end procedure
```

Algorithm 4 Update($\mathsf{Tree}_h^t, \mathcal{H}, i, m'_i$): Updates the tree Tree_h^t by replacing leaf α_i^0 by m'_i and re-computing the necessary internal blocks.

```
1: procedure Update($\mathsf{Tree}_h^t, \mathcal{H}, i, m'_i$)
2:     $\alpha_i^0 := m'_i$
3:     $\mathsf{path} := PATH(\mathsf{Tree}_h^t, \mathcal{H}, \alpha_i^0)$
4:     for $j = 0$ to $\mathcal{H} - 2$ do
5:         $\alpha_\ell^j := \mathsf{path}_j$
6:         $k := \ell \bmod t$
7:         $\mathsf{path}_{j+1}[k] := h(\mathsf{path}_j)$   ▷ Note that here the value is written in the $\mathsf{Tree}_h^t$ and not just in the local variable.
8:     end for
9: end procedure
```

Theorem 6.4.1. *Let $\mathcal{X} \in \Omega(\lambda^{1/\mu})$, $h : \{0,1\}^{\mathcal{X}} \to \{0,1\}^{\mathcal{X}\mu}$, and $t := \mathcal{X}^{1-\mu}$, for constant $0 < \mu < 1$. Assume h is a collision resistant hash function. For any message $M = (m_1, m_2, \ldots, m_n) \in \{0,1\}^{\mathcal{X}\cdot n}$, where $n := \mathcal{X}^c$, consider the corresponding t-slice Merkle Tree, Tree_h^t with root $\mathsf{Root}_{h}{}_h^t$. Then for any $i \in [n]$ and any polynomial time adversary $\mathcal{A}$,*

$$\Pr\left[\begin{array}{c} m_i \neq m_i' \,\wedge\, m_i' = \mathsf{path'}_0 \wedge \\ \text{consistencyCheck}(\mathsf{path}', \mathsf{Root}_{h}{}_h^t) = \mathsf{TRUE} \end{array} \middle| (m_i', \mathsf{path}') \leftarrow \mathcal{A}(M, h)\right] \leq \mathsf{negl}(\lambda)$$

Moreover, given a path path passing the leaf m_i, and a new value m_i', the Update *algorithm computes $\mathsf{Root}_{h}{}_h^t(M')$ in time $\Theta(\mathcal{H})$, where $\mathcal{H} := \frac{c}{1-\mu} + 1$, where $M' = (m_1, \ldots, m_{i-1}, m_i', m_{i+1}, \ldots, m_n)$.*

Proof. *The second part of Theorem 6.4.1 is immediate by inspection of Algorithm 4.*

For the first part of the theorem, we assume towards contradiction that for some message $M = (m_1, m_2, \ldots, m_n)$ with $m_i \in \{0,1\}^{\mathcal{X}}$, there is an efficient adversary $\mathcal{A}$ such that

$$\Pr\left[\begin{array}{c} m_i \neq m_i' \,\wedge\, m_i' = \mathsf{path'}_0 \wedge \\ \text{consistencyCheck}(\mathsf{path}', \mathsf{Root}_{h}{}_h^t) = \mathsf{TRUE} \end{array} \middle| (m_i', \mathsf{path}') \leftarrow \mathcal{A}(M, h)\right] = \epsilon(\lambda)$$

for non-negligible $\epsilon(\cdot)$. Let path be the path returned by Path *(see Algorithm 2) on input $(\mathsf{Tree}_h^t, \mathcal{H}, \alpha_i^0 = m_i)$. Note that $\mathsf{path}_0 = m_i$. We construct adversary $\mathcal{A}'$ which finds a collision in hash function h with non-negligible probability. The procedure is as follows:*

1. *On input h, adversary $\mathcal{A}'$ instantiates $\mathcal{A}$ on input (M, h).*

2. *Adversary $\mathcal{A}$ returns (m'_i, path'), where $\mathsf{path}' := \mathsf{path}'_0, \ldots, \mathsf{path}'_{\mathcal{H}-1}$.*

3. *$\mathcal{A}'$ checks that $\mathsf{path}'_{\mathcal{H}-1} = \mathsf{Root}^{\,t}_{h\,h}(M)$.*

4. *For $j \in [\mathcal{H} - 2]$, if $\mathsf{path}'_{j+1} = \mathsf{path}_{j+1}$ and $\mathsf{path}'_j \neq \mathsf{path}_j$, then $\mathcal{A}'$ returns collision $(\mathsf{path}'_j, \mathsf{path}_j)$.*

We must first argue that there must be some $j \in [\mathcal{H} - 2]$ for which the condition in Item (4) evaluates to TRUE*. Specifically, there must be some $j \in [\mathcal{H} - 2]$ such that $\mathsf{path}'_{j+1} = \mathsf{path}_{j+1}$ and $\mathsf{path}'_j \neq \mathsf{path}_j$. To see why this is so, recall that we require $m'_i \neq m_i$ and so by definition we must have $\mathsf{path}'_0 \neq \mathsf{path}_0$. On the other hand, $\mathsf{path}'_{\mathcal{H}-1} = \mathsf{path}_{\mathcal{H}-1} = \mathsf{Root}^{\,t}_{h\,h}(M)$. Thus, there must be some $j \in [\mathcal{H} - 2]$ such that $\mathsf{path}'_{j+1} = \mathsf{path}_{j+1}$ and $\mathsf{path}'_j \neq \mathsf{path}_j$.*

Next, we argue that for $j \in [\mathcal{H} - 2]$ such that $\mathsf{path}'_{j+1} = \mathsf{path}_{j+1}$ and $\mathsf{path}'_j \neq \mathsf{path}_j$, $(\mathsf{path}'_j, \mathsf{path}_j)$ yields a collision on h. This follows from the definition of the consistencyCheck *algorithm in Algorithm 3* and the fact that, *by assumption,* consistencyCheck$(\mathsf{path}', \mathsf{Root}^{\,t}_{h\,h}) = $ TRUE*. Specifically, we know that both* consistencyCheck$(\mathsf{path}, \mathsf{Root}^{\,t}_{h\,h}) = $ TRUE *and* consistencyCheck$(\mathsf{path}', \mathsf{Root}^{\,t}_{h\,h}) = $ TRUE*. Therefore, during the runs of both algorithms, it must be the case that, for j as above, the condition in Line 8, $\alpha_i^{j+1}[k] == h(\alpha_i^j)$, evaluated to* True*. Let us now look at the settings of α_i^{j+1} and α_i^j during the runs of* consistencyCheck$(\mathsf{path}, \mathsf{Root}^{\,t}_{h\,h}) = $ TRUE *and* consistencyCheck$(\mathsf{path}', \mathsf{Root}^{\,t}_{h\,h}) = $ TRUE*. In Line 5 of the run of* consistencyCheck$(\mathsf{path}, \mathsf{Root}^{\,t}_{h\,h}) = $ TRUE*, α_i^{j+1} is set to $\alpha_i^{j+1} := \mathsf{path}_{j+1}$, while in Line 6, α_i^j is set to $\alpha_i^j := \mathsf{path}_j$. Thus,* *the condition in Line 8 ensures that* $\mathsf{path}_{j+1}[k] = h(\mathsf{path}_j)$*. On the other hand, in Line 5 of the run of* $\text{consistencyCheck}(\mathsf{path}', \mathsf{Root}^t_{hh}) = \mathsf{TRUE}$*,* α_i^{j+1} *is set to* $\alpha_i^{j+1} := \mathsf{path}'_{j+1}$*, while in Line 6,* α_i^j *is set to* $\alpha_i^j := \mathsf{path}'_j$*. Thus, the condition in Line 8 ensures that* $\mathsf{path}'_{j+1}[k] = h(\mathsf{path}'_j)$*. However, recall that we chose* j *such that* ${\mathsf{path}'}_{j+1} = \mathsf{path}_{j+1}$ *and* ${\mathsf{path}'}_j \neq \mathsf{path}_j$*. Therefore, this implies that* $h(\mathsf{path}'_j) = \mathsf{path}'_{j+1}[k] = \mathsf{path}_{j+1}[k] = h(\mathsf{path}_j)$*. We therefore conclude that* $({\mathsf{path}'}_j, \mathsf{path}_j)$ *yields a collision on* h*.*

Thus, the above adversary $\mathcal{A}'$ *will succeed with same probability as the adversary* $\mathcal{A}$ *and breaks collision resistance of* h *with non-negligible probability* $\epsilon(\lambda)$*. Thus, we arrive at contradiction and so the theorem is proved.*

Theorem 6.4.2. *Assume there exists a semantically secure symmetric encryption scheme, and a non-malleable code against the tampering function class* $\mathcal{F}$*, and leakage resilient against the function class* $\mathcal{G}$*. Then there exists a locally decodable and updatable non-malleable code that has* non-adaptive decode and update with deterministic accesses *and is non-malleable against continual attacks of the tampering class*

$$\bar{\mathcal{F}} \overset{\text{def}}{=} \left\{ \begin{array}{l} f : \hat{\Sigma}^{2n+1} \to \hat{\Sigma}^{2n+1} \text{ and } |f| \leq \text{poly}(k), \text{ such that :} \\ f = (f_1, f_2),\ f_1 : \hat{\Sigma}^{2n+1} \to \hat{\Sigma},\ f_2 : \hat{\Sigma}^{2n} \to \hat{\Sigma}^{2n}, \\ \forall (x_2, \ldots, x_{2n+1}) \in \hat{\Sigma}^{2n}, f_1(\,\cdot\,, x_2, \ldots, x_{2n+1}) \in \mathcal{F}, \\ f(x_1, x_2, \ldots, x_{2n+1}) = (f_1(x_1, x_2, \ldots, x_{2n+1}), f_2(x_2, \ldots, x_{2n+1})) . \end{array} \right\},$$

and is leakage resilient against the class

$$\bar{\mathcal{G}} \overset{\text{def}}{=} \left\{ \begin{array}{l} g : \hat{\Sigma}^{2n+1} \to \mathcal{Y} \text{ and } |g| \leq \text{poly}(k), \text{ such that :} \\ g = (g_1, g_2),\ g_1 : \hat{\Sigma}^{2n+1} \to \mathcal{Y}',\ g_2 : \hat{\Sigma}^{2n} \to \hat{\Sigma}^{2n}, \\ \forall\ (x_2, \ldots, x_{2n+1}) \in \hat{\Sigma}^{2n}, g_1(\ \cdot\ , x_2, \ldots, x_{2n+1}) \in \mathcal{G}. \end{array} \right\}.$$

Moreover, for $n := \mathcal{X}^c \in \text{poly}(\lambda)$*, the coding scheme has locality* $\delta(n)$*, for any* $\delta(n) \in \omega(1)$*.*

Our construction is exactly the same as that of Dachman-Soled et al. [54], except we replace their (standard) Merkle tree with our t*-slice Merkle tree with the parameters described above. We note that the only property of the Merkle hash used in the security proof of [54] is the "collision resistance" property, analogous to our Theorem 6.4.1 above for the* t*-slice Merkle tree. Thus, our security proof follows exactly as theirs does and we therefore omit the full proof. On the other hand, as described in Algorithms 4 and 3, updates and consistency checks require time and number of accesses to memory proportional to the height of the tree,* $\mathcal{H}$*, which is* $\frac{c}{1-\mu} + 1$ *for our choice of parameters, as shown in Lemma 6.4.1 above. Since* $n = \mathcal{X}^c \in \text{poly}(\lambda)$*, it means that the height of the tree will always be less than* $\delta(n)$*, for any* $\delta(n) \in \omega(1)$*. On the other hand, [54] used a standard (binary) Merkle tree with height* $\Theta(\log n)$*. Therefore, while [54] requires locality* $\Theta(\log n)$*, we achieve locality* $\delta(n)$*, for any* $\delta(n) \in \omega(1)$*.*

Finally, we give a concrete example of the resulting leakage and tampering classes we can tolerate via Theorem 6.4.2 when instantiating the underlying non-

malleable code with a concrete construction. Specifically, we consider instantiating the underlying non-malleable code with the construction of of Liu and Lysyanskaya [100], which achieves both leakage and tamper resilience for split-state functions. Combining the constructions of [100] and [54] yields codewords consisting of $2n + 1$ *blocks. We next describe the leakage and tampering classes* $\bar{\mathcal{G}}, \bar{\mathcal{F}}$ *that can be tolerated on the* $2n + 1$*-block codeword.* $\bar{\mathcal{G}}$ *consists of leakage functions* g *such that* g *restricted to the first block (i.e.* g_1*) is any (poly-sized) length-bounded split-state function;* g_2 *on the other hand, can leak all other parts.* $\bar{\mathcal{F}}$ *consists of tampering functions* f *such that* f *restricted to the first block (i.e.* f_1*) is any (poly-sized) split-state function. On the other hand* f *restricted to the rest (i.e.* f_2*) is any poly-sized function. We also remark that the function* f_2 *itself can depend on the split-state leakage on the first part.*

Chapter 7: Upper and Lower Bounds for Continuous Non-Malleable Codes

7.1 Introduction

In this chapter we study continuous non-malleable codes (CNMC) and analyze the minimal computational assumptions required to construct CNMC.

Recall, that CNMC with respect to a tampering class $\mathcal{F}$ is informally defined as follows: Given a coding scheme $\Pi = (\mathsf{E}, \mathsf{D})$, where E is the encoding function and D is the decoding function, the adversary interacts with an oracle $\mathcal{O}_\Pi(C)$, parameterized by Π and an encoding of a message m, $C \leftarrow \mathsf{E}(m)$. We refer to the encoding C as the "challenge" encoding. In each round, the adversary submits a tampering function $f \in \mathcal{F}$. The oracle evaluates $C' = f(C)$. If $\mathsf{D}(C') = \perp$, the oracle outputs $\perp$ and a "self-destruct" occurs, aborting the experiment. If $C' = C$, the oracle outputs a special message "same." Otherwise, the oracle outputs C'. We emphasize that the entire tampered codeword is returned to the adversary in this case. A CNMC is secure if for every pair of messages m_0, m_1, the adversary's view in the above game is computationally indistinguishable when the message is m_0 or m_1.

Information-theoretic impossibility. *The original CNMC paper of [71] showed an information-theoretic impossibility result for 2-split-state CNMC. To aid the subsequent discussion, we present an outline of this result. The impossibility result considers a property of 2-split-state CNMC known as (perfect) "uniqueness." Informally, perfect uniqueness means that there do not exist triples* (x, y, z) *such that either (1)* $y \neq z \wedge \mathsf{D}(x, y) \neq \perp \wedge \mathsf{D}(x, z) \neq \perp$ *OR (2)* $x \neq y \wedge \mathsf{D}(x, z) \neq \perp \wedge \mathsf{D}(y, z) \neq \perp$*. First, a perfectly unique CNMC cannot be information-theoretically secure since, given* L*, the split-state tampering function can find the unique* R *such that* $\mathsf{D}(L, R) \neq \perp$ *and then tamper based on* $m = \mathsf{D}(L, R)$*. On the other hand, if the CNMC is not perfectly unique, then the following is an efficient attack (with non-uniform advice): Given a tuple* L'_1, L'_2, R' *such that* $\mathsf{D}(L'_1, R') \neq \perp$ *and* $\mathsf{D}(L'_2, R') \neq \perp$*, the adversary can learn* L *bit-by-bit by using the following tampering function in the* i*-th round:* f_L *does the following: If the* i*-th bit of* L *is equal to* 0*, replace* L *with* L'_1*. Otherwise, replace* L' *with* L'_2*.* f_R *always replaces* R *with* R'*. Now, in the* i*-th round, if the oracle returns* (L'_1, R')*, then the adversary learns that the* i*-th bit of* L *is equal to* 0*. If the oracle returns* (L'_2, R')*, then the adversary learns that the* i*-th bit of* L *is equal to* 1*. Once* L *is fully recovered, the adversary can tamper based on* $m = \mathsf{D}(L, R)$*.*

The computational setting. *The above shows that the CNMC setting is distinguished from other NMC settings, since information-theoretic (unconditional) security is impossible. Prior work has shown how to construct 2-split-state CNMC in the* CRS model *under the assumptions of collision-resistant hash functions and NIZK. On the other hand, CNMC's imply commitment schemes, which in turn imply*

OWF. It remains to determine where CNMC lies in terms of complexity assumptions and what are the minimal computational assumptions needed to achieve CNMC. As mentioned previously, a very recent work of Ostrovsky et al. [105] addressed minimizing computational assumptions under a relaxed definition of CNMC. See Section 2.3 for more details.

Black-box reductions. *In general, it is not feasible to unconditionally rule out the construction of a primitive G from a cryptographic assumption H, since unconditionally ruling it out is as hard as proving $P \neq NP$. Despite this, we can still show that the proof techniques we have at hand cannot be used to construct G from assumption H. In the literature, this is typically done by showing that there is no black-box reduction from primitive G to assumption H. In this work, what we mean by a black-box reduction is a reduction that accesses the* adversary *in an input/output fashion only. However, we allow non-black-box usage of the assumption H in both the construction and the proof (see Definition 7.2.3 for a formal definition tailored to CNMC). While there are some exceptions [21, 25], the vast majority of cryptographic reductions are black-box in the adversary.*

7.1.1 Our Results

We present upper and lower bounds for CNMC in the 2-split-state model. First, we show that with no CRS, single-bit CNMC in the 2-split-state model (with a black-box security proof) is impossible to construct from any falsifiable assumption.

Theorem 7.1.1 (Informal)**.** *There is no black-box reduction from a single-bit, 2-*

split-state, CNMC scheme $\Pi = (\mathsf{E}, \mathsf{D})$ *to any falsifiable assumption.*

On the other hand, in the CRS model, we show how to achieve single-bit CNMC in the 2-split-state model from injective one-way functions.

Theorem 7.1.2. *Assuming the existence of an injective one-way function family, there is a construction of a 2-split-state CNMC for encoding single bit, in the CRS model. Moreover, the corresponding reduction is black-box.*

Actually, we show a somewhat more general result: First, we define a (to the best of our knowledge) new type of commitment scheme called one-to-one commitment schemes in the CRS model. *Informally, these commitment schemes have the additional property that with all but negligible probability over* Σ *produced by CRS generation, for every string com, there is at most a* single *string d that will be accepted as a valid decommitment for com (See Definition 7.2.5 for a formal definition). We also define the notion of a 2-split-state CNM Randomness Encoder, which is the continuous analogue of the non-malleable randomness encoder recently introduced by [91] (See Definition 7.2.1). We then show the following:*

Theorem 7.1.3. *Assuming the existence of one-to-one commitment schemes in the CRS model, there is a construction of a 2-split-state CNM Randomness Encoder in the CRS model. Moreover, the corresponding reduction is black-box.*

One-to-one commitment schemes in the CRS model can be constructed from any injective one-way function family. Furthermore, we show (in Section 7.4.1) that 2-split-state CNM Randomness Encoders in the CRS model imply 2-split-state

CNMC for encoding single bit, in the CRS model. We therefore obtain Theorem 7.1.2 as a corollary. Moreover, CNMC with perfect *uniqueness in the CRS model implies one-to-one commitment schemes in the CRS model in a straightforward way.*

We leave open the question of constructing CNMC in the CRS model from (non-injective) one-way functions and/or showing a black-box separation between the two primitives. Finally, we extend the techniques from our single-bit construction above to achieve the following:

Theorem 7.1.4. *Assuming the existence of one-to-one commitment schemes in the CRS model, there is a construction of a multi-bit, 4-split-state CNMC in the CRS model. Moreover, the corresponding reduction is black-box.*

Are prior CNMC reductions "black-box"? *Prior CNMC reductions often proceed in a sequence of hybrids, where in the final hybrid, the description of the adversary is incorporated in the definition of a leakage function. It is then shown that the leakage-resilience properties of an underlying encoding imply that the view of the adversary is statistically close when the encoded message is set to m_0 or m_1. While this may seem like non-black-box usage of the adversary, we note that typically the leakage-resilience of the underlying encoding is information-theoretic. When converting a hybrid-style proof to a reduction, the reduction will choose one of the hybrid steps at random and use the fact that a distinguisher between some pair of consecutive hybrids implies an adversary breaking an underlying assumption. Therefore, reductions of the type discussed above are still black-box in the adversary, pairs of consecutive hybrids whose indistinguishability is implied by a* computational *assumption yield a* *reduction in which the adversary is used in a black-box manner.*

7.1.2 Technical Overview

Lower bound. *Recall that prior work has shown that if a CNMC is not perfectly unique, then there is an efficient attack (with non-uniform advice). Thus, it remains to show that there is no black-box reduction from a single-bit,* perfectly unique *CNMC scheme to any falsifiable assumption. We use the meta-reduction approach, which is to prove impossibility by showing that given only black-box access to the split-state adversary, $A = (A_L, A_R)$, the reduction cannot distinguish between the actual adversary and a* simulated *(efficient) adversary (which is possibly stateful). Since the view of the reduction is indistinguishable in the two cases, the reduction must also break the falsifiable assumption when interacting with the simulated adversary. But this in turn means that there is an efficient adversary (obtained by composing the reduction and the simulated adversary), which contradicts the underlying falsifiable assumption. Consider the following stateless, inefficient, split-state adversary $A = (A_L, A_R)$, which leverages the uniqueness property of the CNMC scheme: The real adversary, given L (resp. R), recovers the corresponding unique valid codeword (L, R) (if it exists) and decodes to get the bit b. If $b = 0$, the real adversary encodes a random bit b' using internal randomness that is tied to (L, R), and outputs the left/right side as appropriate. If $b = 1$ or there is no corresponding valid codeword, the real adversary outputs the left/right side of a random encoding of a random bit, b'' (generated using internal randomness that is tied to L or R respectively). The* *simulated adversary is stateful and keeps a table containing all the L and R values that it has seen. Whenever a L (resp. R) query is made, the simulated adversary first checks the table to see if a matching query to R (resp. L) such that $\mathsf{D}(L, R) \neq \perp$ was previously made. If not, the simulated adversary chooses a random encoding, (L', R'), of a random bit b', stores it in the table along with the L/R query that was made and returns either L' or R' as appropriate. If yes, the simulated adversary finds the corresponding R (resp. L) along with the pair (L', R') stored in the table. The simulated adversary then decodes (L, R) to find out b. If $b = 0$, the simulated adversary returns either L' or R' as appropriate. Otherwise, the simulated adversary returns the left/right side of an encoding of a random bit b''. The uniqueness property allows us to prove that the input/output behavior of the real adversary is identical to that of the simulated adversary. See Section 7.3 for additional details. For a discussion on why our impossibility result does not hold for the relaxed CNMC notion considered by [105], see Section 2.3.*

Upper bound. *For the upper bound, we construct a new object called a continuous non-malleable randomness encoder (see Definition 7.2.1), which is the continuous analogue of the non-malleable randomness encoder recently introduced by [91]. Informally, a continuous non-malleable randomness encoder is just a non-malleable code for randomly chosen messages. It is then straightforward to show that a continuous non-malleable randomness encoder implies a single-bit continuous non-malleable code (see Section 7.4.1).*

At a high level, the difficulty in proving continuous non-malleability arises *from the need of the security reduction to simulate the interactive tampering oracle, without knowing the message underlying the "challenge" encoding. The approach of prior work such as [71] was to include a NIZK Proof of Knowledge in each part of the codeword to allow the simulator to extract the second part of the encoding, given the first. This then allowed the simulator (with some additional leakage) to respond correctly to a tampering query, while knowing only one of the two split-states of the original encoding. In our setting, we cannot use NIZK, since our goal is to reduce the necessary complexity assumptions; therefore, we need a different extraction technique.*[1] *Our main idea is as follows: To respond to the i-th tampering query, we run the adversarial tampering function on random (simulated) codewords (L', R') that are consistent with the output seen thus far (denoted $\mathbf{out}_A^{i-1}$) and keep track of frequent outcomes (occurring with non-negligible probability) of the tampering function, $\widehat{L}, \widehat{R}$. I.e. S_L (resp. S_R) is the set of values of $\widehat{L}$ (resp. $\widehat{R}$) such that with non-negligible probability over choice of L' (resp. R'), it is the case that $\widehat{L} = f_L(L')$ (resp. $\widehat{R} = f_R(R')$). We then show that if the outcome of the tampering function applied to the actual "challenge" split-state L or R is not equal to one of these frequent outcomes (i.e. $f_L(L) \notin S_L$ or $f_R(R) \notin S_R$), then w.h.p. the decode function* D *outputs $\perp$. This will allow us to simulate the experiment with only a small amount of leakage (to determine which of the values in S_L/S_R should be outputted). Note that, while the sets S_L/S_R are small, and so only a few bits are needed to specify the outcome, conditioned on the outcome being in S_L/S_R, the CNMC experiment*

[1]Note that our extraction technique is inefficient. This is ok, since the goal of the extraction technique is simply to show that the view of the adversary can be simulated given a small amount of leakage on each of the two split-states. Then, information-theoretic properties of the encoding are used to show that the view of the adversary must be independent of the random encoded value.

runs for an unbounded *number of times, and so even outputting a small amount of information in each round can ultimately lead to unbounded leakage. To solve this problem, we also consider the* most frequent *outcome in the sets S_L/S_R. This is the value of $\widehat{L}$ (resp. $\widehat{R}$) that occurs with the highest probability when $f_L(L')$ (resp. $f_R(R')$) is applied to consistent L' (resp. R'). Note that if a value $\widehat{L}'$ (resp. $\widehat{R}'$) is* not *the most frequent value, then it occurs with probability at most $1/2$. We argue that, for each round i of the CNMC experiment, the probability that a value $\widehat{L}'$ (resp. $\widehat{R}'$) that is not the most frequent value is outputted by f_L (resp. f_R)* and *self-destruct does not occur is at most $1/2$. This allows us to bound, w.h.p., the number of times in the entire tampering experiment that the value outputted by f_L (resp. f_R) is not the most frequent value. Thus, when the value outputted by f_L (resp. f_R)* is *the most frequent value, the leakage function outputs* nothing*, since the most frequent value can be reconstructed from the given information. In contrast, if the value outputted by f_L (resp. f_R) is* not *the most frequent value, but is in the sets S_L/S_R, then it has a small description and, moreover, this event occurs a bounded number of times. Therefore, we can afford to leak this information up to some upperbounded number of rounds, while the total amount of leakage remains small relative to the length of the encoding. Looking ahead, our construction will use a two-source extractor, whose properties will guarantee that even given the leakage (which contains all the information needed to simulate the CNMC experiment), the decoded value remains uniform random.*

To show that if the outcome of the tampering function is not in S_L or S_R, then decode outputs $\perp$ w.h.p., we first use the "uniqueness" property, which says *that for every $\widehat{L} = f_L(L)$ (resp. $\widehat{R} = f_R(R)$), there is at most a single "match", $\widehat{R}'$ (resp. $\widehat{L}'$), such that $\mathsf{D}_\Sigma(\widehat{L}, \widehat{R}') \neq \perp$ (resp. $\mathsf{D}_\Sigma(\widehat{L}', \widehat{R}) \neq \perp$). Given the "uniqueness" property, it is sufficient to show that for every setting of $L, \mathbf{out}_A^{i-1}$*

$$\Pr[f_R(R) = \widehat{R}' \wedge \widehat{R}' \notin S_R \mid L \wedge \mathbf{out}_A^{i-1}] \leq \mathsf{negl}(n) \tag{7.1.1}$$

and that for every setting of $R \wedge \mathbf{out}_A^{i-1}$

$$\Pr[f_L(L) = \widehat{L}' \wedge \widehat{L}' \notin S_L \mid R \wedge \mathbf{out}_A^{i-1}] \leq \mathsf{negl}(n). \tag{7.1.2}$$

To prove the above, we first argue that for the "challenge" codeword, (L, R), the split-states L and R are conditionally independent, given $\mathbf{out}_A^{i-1}$ (assuming no $\perp$ has been outputted thus far) and an additional simulated part of the codeword. This means that the set of frequent outcomes S_L (resp. S_R) conditioned on $\mathbf{out}_A^{i-1}$ is the same as the set of frequent outcomes S_L (resp. S_R) conditioned on both *$\mathbf{out}_A^{i-1}$ and R (resp. L). So for any $\widehat{R} \notin S_R$,*

$$\Pr[f_R(R) = \widehat{R} \mid L \wedge \mathbf{out}_A^{i-1}] \leq \mathsf{negl}(n)$$

and for any $\widehat{L} \notin S_L$,

$$\Pr[f_L(L) = \widehat{L} \mid R \wedge \mathbf{out}_A^{i-1}] \leq \mathsf{negl}(n).$$

Since $\widehat{R}'$ (resp. $\widehat{L}'$) is simply a particular setting of $\widehat{R} \notin S_R$ (resp. $\widehat{L} \notin S_L$), we have *that (7.1.1) and (7.1.2) follow.*

For the above analysis, we need the encoding scheme to possess the following property: The L, R sides of the "challenge" codeword are conditionally independent given $\mathbf{out}_A^{i-1}$ (and an additional simulated part of the codeword), but any tampered split-state $f_L(L)$ or $f_R(R)$ created by the adversary has at most a single "match," $\widehat{R}'$ or $\widehat{L}'$.

To explain how we achieve this property, we briefly describe our construction. Our construction is based on a non-interactive, equivocal commitment scheme in the CRS model and a two-source (inner product) extractor. Informally, an equivocal commitment scheme is a commitment scheme with the normal binding and hiding properties, but for which there exists a simulator that can output simulated commitments which can be opened to both 0 *and* 1. *In the CRS model, the simulator also gets to sample a simulated CRS. Moreover, the CRS and commitments produced by the simulator are indistinguishable from real ones.*

To encode a random value m, random vectors c_L, c_R such that $\langle c_L, c_R \rangle = m$ are chosen. We generate a commitment com to $c_L || c_R$. The commitment scheme has the additional property that adversarially produced commitments are statistically binding (even if an equivocal commitment has been released) and have at most a single *valid decommitment string. The left (resp. right) split-state L (resp. R) consists of com and an opening of com to the bits of c_L (resp. c_R). The special properties of the commitment scheme guarantee the "perfect uniqueness" property of the code. In the security proof, we replace the statistically binding commitment com in the "challenge" codeword with an equivocal commitment. Thus, each split-state of the* *challenge encoding, L (resp. R), contains no information about c_R (resp. c_L). Moreover, assuming "$\perp$" is not yet outputted, the output received by the adversary in the experiment at the point that the i-th tampering function is submitted, denoted $\mathbf{out}_A^{i-1}$ is of the form $(f_L^1(L) = v_1, f_R^1(R) = w_1), \ldots, (f_L^{i-1}(L)) = v_{i-1}, f_R^{i-1}(R) = w_{i-1})$, where for $j \in [i-1]$, v_j is equal to the left value outputted in response to the j-th query and w_j is equal to the right value outputted in response to the j-th query. (note that v_j/w_j can be set to "same" if the tampering function leaves L/R unchanged). This allows us to argue that the distribution of $L \mid \mathbf{out}_A^{i-1}, R$ (resp. $R \mid \mathbf{out}_A^{i-1}, L$) is identical to the distribution of $L \mid \mathbf{out}_A^{i-1}$ (resp. $R \mid \mathbf{out}_A^{i-1}$) which implies that the left and right hand sides are conditionally independent given $\mathbf{out}_A^{i-1}$ and the equivocal commitment, as desired. See Section 7.4 for additional details.*

Extension to 4-state CNMC in CRS model from OWF. *To encode a message m we now generate random $(c_{L,1}, c_{R,1}, c_{L,2}, c_{R,2})$ conditioned on $\langle c_{L,1}, c_{R,1}\rangle + \langle c_{L,2}, c_{R,2}\rangle = m$ (where addition is over a finite field). Now, we generate a commitment com to $c_{L,1}||c_{R,1}||c_{L,2}||c_{R,2}$. Each of the four split states now consists of com and an opening of com to the bits of $c_{L,b}$ (resp. $c_{R,b}$). The analysis is similar to the previous case and requires the property that at each point in the experiment the distribution of $\langle c_{L,1}, c_{R,1}\rangle$ (resp. $\langle c_{L,2}, c_{R,2}\rangle$) is uniform random, conditioned on the output thus far. Our techniques are somewhat similar to those used in [60] in their construction of* 2*t-split-state continuously non-malleable codes from t-split-state one-way continuously non-malleable codes. See Section 7.5 for additional details.*

7.2 Preliminaries

In this section we will provide some more definitions and background required for our constructions and results presented in chapter 7.

7.2.1 Randomness Extractors

The following lemma is from [108].

Lemma 7.2.1 (Inner-Product Two-Source Extractor). *Let $\vec{X}, \vec{Y}, Z$ be correlated variables, where $\vec{X}, \vec{Y}$ have their support in $\{0,1\}^\ell = \mathbb{F}_{2^\lambda}^{\frac{\ell}{\lambda}}$ and $\lambda | \ell$, and are independent conditioned on Z. Let U_λ be uniform and independent on $\mathbb{F}_{2^\lambda}$. Then*

$$\Delta((Z, \langle \vec{X}, \vec{Y} \rangle), (Z, U_\lambda)) \leq 2^{-s}$$

for some $s \geq 1 + \frac{1}{2}(k_X + k_Y - \ell - \lambda)$, where $k_X := \widetilde{\mathbb{H}}_\infty(\vec{X}|Z)$, $k_Y := \widetilde{\mathbb{H}}_\infty(\vec{Y}|Z)$

7.2.2 Continuous Non-Malleable Randomness Encoder

The following definition is an adaptation of the notion of Non-Malleable Randomness Encoders [91] to the continuous setting.

Definition 7.2.1. *Let* $\mathsf{Code} = (\mathsf{CRSGen}, \mathsf{CNMREnc}, \mathsf{CNMRDec})$ *be such that* CRSGen *takes security parameter λ as input and outputs a string of length* $\mathsf{crs}_1 = \mathrm{poly}(\lambda)$ *as CRS.* $\mathsf{CNMREnc} : \{0,1\}^{\mathsf{crs}_1} \times \{0,1\}^r \to \{0,1\}^\lambda \times (\{0,1\}^{n_1}, \{0,1\}^{n_2})$ *is defined as* $\mathsf{CNMREnc}(r) = (\mathsf{CNMREnc}_{1,\mathsf{crs}}(r), \mathsf{CNMREnc}_{2,\mathsf{crs}}(r)) = (m, (x_0, x_1))$ *and* $\mathsf{CNMRDec} : \{0,1\}^{\mathsf{crs}_1} \times \{0,1\}^{n_1} \times \{0,1\}^{n_2} \to \{0,1\}^\lambda$.

We say that $(\mathsf{CRSGen}, \mathsf{CNMREnc}, \mathsf{CNMRDec})$ *is a* continuous non-malleable randomness encoder *with message space* $\{0,1\}^\lambda$ *and codeword space* $\{0,1\}^{n_1} \times \{0,1\}^{n_2}$*, for the distribution* $\mathcal{R}$ *on* $\{0,1\}^r$ *with respect to the 2-split-state family* $\mathcal{F}$ *if the following holds true:*

- *Correctness:*

$$\Pr_{r \leftarrow \mathcal{R}}[\mathsf{CNMRDec}_{\mathsf{crs}}(\mathsf{CNMREnc}_{2,\mathsf{crs}}(r)) = \mathsf{CNMREnc}_{1,\mathsf{crs}}(r)] = 1$$

- *Continuous Non-Malleability:*

$$(\mathsf{crs}, \mathsf{CNMREnc}_{1,\mathsf{crs}}(R), \mathsf{out}_{\mathsf{crs},\mathcal{A}}(R)) \approx_c (\mathsf{crs}, U_\lambda, \mathsf{out}_{\mathsf{crs},\mathcal{A}}(R))$$

where $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$*,* R *is a uniform random variable over* $\{0,1\}^r$*,* U_λ *is a uniform random variable over* $\{0,1\}^\lambda$ *and* $\mathsf{out}_{\mathsf{crs},\mathcal{A}}(R)$ *is defined as follows:*

$$\mathsf{out}_{\mathsf{crs},\mathcal{A}}(R) \leftarrow \mathcal{A}^{\mathcal{O}_{\mathsf{CNM}}(\mathsf{crs},(X_0,X_1),(\cdot,\cdot))} : (X_0, X_1) \leftarrow \mathsf{CNMREnc}_{2,\mathsf{crs}}(R)$$

where $\mathcal{O}_{\mathsf{CNM}}$ *runs with* $\mathsf{CNMRDec}$ *as decoding algorithm.*

Next, we present definitions related to falsifiable assumptions and black-box reductions, strong one-time signature schemes, and equivocal commitment scheme.

7.2.3 Falsifiable Assumptions and Black-Box Reductions

Definition 7.2.2. *A falsifiable assumption consists of* PPT *interactive challenger* $\mathcal{C}(1^\lambda)$ *that runs in time* $\text{poly}(\lambda)$ *and a constant* $0 \leq \delta < 1$. *The challenger* $\mathcal{C}$ *interacts with a machine* $\mathcal{A}$ *and may output special symbol* win. *If this occurs,* $\mathcal{A}$ *is said to win* $\mathcal{C}$. *For any adversary* $\mathcal{A}$, *the advantage of* $\mathcal{A}$ *over* $\mathcal{C}$ *is defined as:*

$$\text{Adv}_{\mathcal{A}}^{(\mathcal{C},\delta)} = |\Pr\left[\mathcal{A}(1^\lambda)\, wins\, \mathcal{C}(1^\lambda)\right] - \delta|,$$

where the probability is taken over the random coins of $\mathcal{A}$ *and* $\mathcal{C}$. *The assumption associated with the tuple* $(\mathcal{C}, \delta)$ *states that for every (non-uniform) adversary* $\mathcal{A}(1^\lambda)$ *running in time* $\text{poly}(\lambda)$,

$$\text{Adv}_{\mathcal{A}}^{(\mathcal{C},\delta)} = \mathsf{negl}(\lambda).$$

If the advantage of $\mathcal{A}$ *is non-negligible in* λ *then* $\mathcal{A}$ *is said to* break *the assumption.*

Definition 7.2.3. *Let* $\Pi = (\mathsf{E}, \mathsf{D})$ *be a split-state CNMC. We say that the non-malleability of* Π *can be proven via a* black-box reduction *to a falsifiable assumption, if there is an oracle access machine* $\mathcal{M}^{(\cdot)}$ *such that for every (possibly inefficient)* Π*-adversary* $\mathcal{P}^*$, *the machine* $\mathcal{M}^{\mathcal{P}^*}$ *runs in time* $\text{poly}(\lambda)$ *and breaks the assumption.*

7.2.4 (Strong) One-Time Signature Schemes

A digital signature scheme consists of a triple of ppt algorithms $(\mathsf{Gen}, \mathsf{Sign}, \mathsf{Verify})$ *such that:*

- Gen *takes the security parameter* 1^λ *as input and generates a pair of keys: a public verification key* vk*, and a secret signing key* $\textsc{sk}$*.*
- Sign *takes as input a secret key* $\textsc{sk}$ *and a message* m*, and generates a signature* σ*. We write this as* $\sigma \leftarrow \mathsf{Sign}_{\textsc{sk}}(m)$*.*
- Verify *takes as input a verification key* vk*, a message* m*, and a (purported) signature* σ *and outputs a single bit indicating acceptance or not.*

For correctness, we require that for all $(\mathsf{vk}, \textsc{sk})$ *output by* $\mathsf{Gen}(1^\lambda)$*, for all messages* m*, and for all* $\sigma \leftarrow \mathsf{Sign}_{\textsc{sk}}(m)$*, we have* $\mathsf{Verify}_{\mathsf{vk}}(m, \sigma) = 1$*.*

7.2.5 Equivocal Commitment Scheme

We start by defining the basic commitment schemes and then present the notion of equivocal bit-commitment schemes *introduced by Di Crescenzo et al. in [58].*

Definition 7.2.4 (Commitment Scheme)**.** *A (non-interactive) commitment scheme in the CRS model for the message space* $\mathcal{M}$*, is a triple* $(\mathsf{CRSGen}, \mathsf{Commit}, \mathsf{Open})$ *such that:*

- $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$ *generates the CRS.*

- *For all* $m \in \mathcal{M}$*,* $(com, d) \leftarrow \mathsf{Commit}_{\mathsf{crs}}(m)$ *is the* commitment/opening *pair for the message* m*. Specifically;* com *is the* commitment value *for* m*, and* d *is the* opening.

- $\mathsf{Open}_{\mathsf{crs}}(com, d) \to \tilde{m} \in \mathcal{M} \cup \{\perp\}$*, where* $\perp$ *is returned when* com *is not a valid commitment to any message.*

  The commitment scheme must satisfy the standard correctness requirement,

$$\forall \lambda \in \mathbb{N}, \forall m \in \mathcal{M} \text{ and } \mathsf{crs} \in \mathcal{CRS}, \ \Pr\left[\mathsf{Open}_{\mathsf{crs}}(\mathsf{Commit}_{\mathsf{crs}}(m)) = m\right] = 1$$

where, $\mathcal{CRS}$ *is the set of all possible valid CRS's generated by* $\mathsf{CRSGen}(1^\lambda)$ *and where the probability is taken over the randomness of* Commit*.*

The commitment scheme provides the following 2 security properties:

Hiding: *It is computationally hard for any adversary* $\mathcal{A}$ *to generate two messages* $m_0, m_1 \in \mathcal{M}$ *such that* $\mathcal{A}$ *can distinguish between their corresponding commitments. Formally, for any PPT adversary* $\mathcal{A} = (\mathcal{A}_1, \mathcal{A}_2)$ *it should hold that:*

$$\Pr\left[b = b' \middle| \begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda), (m_0, m_1, \alpha) \leftarrow \mathcal{A}_1(\mathsf{crs}), b \leftarrow \{0,1\}, \\ (com, d) \leftarrow \mathsf{Commit}_{\mathsf{crs}}(m_b), b' \leftarrow \mathcal{A}_2(com, \alpha) \end{array} \right] \leq \frac{1}{2} + \mathsf{negl}(\lambda)$$

Binding: *It is computationally hard for any adversary* $\mathcal{A}$ *to find a triple* (com, d, d') *such that* both (com, d) *and* (com, d') *are valid commitment/opening pairs for* *some $m, m' \in \mathcal{M}$ respectively, and $m \neq m'$. Formally, for any PPT adversary $\mathcal{A}$ it should hold that:*

$$\Pr\left[\begin{array}{c|c} m \neq m' \wedge & \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda), (com, d, d') \leftarrow \mathcal{A}(\mathsf{crs}), \\ & \\ m, m' \neq \perp & m \leftarrow \mathsf{Open}_{\mathsf{crs}}(com, d), m' \leftarrow \mathsf{Open}_{\mathsf{crs}}(com, d') \end{array}\right] \leq \mathsf{negl}(\lambda)$$

Definition 7.2.5 (One-to-One Commitment Scheme in the CRS Model)**.** *Let* $(\mathsf{CRSGen}, \mathsf{Commit}, \mathsf{Open})$ *be a bit-commitment scheme in CRS model. We say that* $(\mathsf{CRSGen}, \mathsf{Commit}, \mathsf{Open})$ *is a* one-to-one commitment scheme *if with all but negligible probability over $b \leftarrow \{0,1\}$,* $\mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda)$*,* $(com, d) \leftarrow \mathsf{Commit}_{\mathsf{crs}}(b)$*, $d' = d$ is the unique string such that* $\mathsf{Open}(com, d') \neq \perp$*.*

Definition 7.2.6 (Non-Interactive Equivocable Bit-Commitment Scheme)**.** *Let* $(\mathsf{CRSGen}, \mathsf{Commit}, \mathsf{Open})$ *be a bit-commitment scheme in CRS model. We say that* $(\mathsf{CRSGen}, \mathsf{Commit}, \mathsf{Open})$ *is a* non-interactive equivocable bit-commitment scheme *in the CRS model if there exists an efficient probabilistic algorithm* S_{Eq} *which on input 1^λ outputs a 4-tuple* $(\mathsf{crs}', com', d'_0, d'_1)$ *satisfying the following:*

- $\Pr[\mathsf{Open}_{\mathsf{crs}'}(com', d'_b) = b]$ *for $b \in \{0,1\}$.*
- *For $b \in \{0,1\}$, it holds that* $\mathsf{out}_{\mathsf{Commit}}(b) \approx_\varepsilon \mathsf{out}_{\mathsf{S}_{Eq}}(b)$ *where the random variables* $\mathsf{out}_{\mathsf{Commit}}(b)$ *and* $\mathsf{out}_{\mathsf{S}_{Eq}}(b)$ *are defined as follows:*

$$\left\{\begin{array}{c} \mathsf{crs} \leftarrow \mathsf{CRSGen}(1^\lambda);\ (com, d) \leftarrow \mathsf{Commit}_{\mathsf{crs}}(b); \\ \\ \mathsf{out}_{\mathsf{Commit}}(b) : (\mathsf{crs}, com, d) \end{array}\right\}$$

$$\approx \left\{ \begin{matrix} (\mathsf{crs}', com', d'_0, d'_1) \leftarrow \mathsf{S}_{Eq}(1^\lambda); \\ \\ \mathsf{out}_{\mathsf{S}_{Eq}}(b) : (\mathsf{crs}', com', d'_b) \end{matrix} \right\}$$

We now present variant of the commitment scheme presented by Naor in [101], specifically we present the same construction in CRS model. This is also presented in [58].

Let $\lambda > 0$ be an integer, let $G : \{0,1\}^\lambda \to \{0,1\}^{3\lambda}$ be a pseudo-random generator.

- $\mathsf{CRSGen}(1^\lambda)$*: Output a uniform random string* crs *of length 3λ.*

- $\mathsf{Commit}_{\mathsf{crs}}(b)$*: Choose uniform random seed $s \in \{0,1\}^\lambda$ and compute $t = G(s)$. If $b = 0$, set $com := t$. If $b = 1$, set $com := t \oplus \mathsf{crs}$. Output c. Output decommitment $d = s$.*

- $\mathsf{Open}_{\mathsf{crs}}(com, d)$*: If $com = G(d)$, then output 0. Else if, $com = G(d) \oplus \mathsf{crs}$, then output 1. Output $\perp$ otherwise.*

Claim 7.2.1.The scheme presented above is equivocal *commitment scheme.*

In order to prove claim 7.2.1 we need to show an efficient simulator S_{Eq} which outputs $(\mathsf{crs}', com', d'_0, d'_1)$ on input 1^λ. Following is the description of S_{Eq}: On input 1^λ, S_{Eq} chooses two uniform random seeds $s_0, s_1 \in \{0,1\}^\lambda$ and computes $u = G(s_0)$ and $v = G(s_1)$. Set $\mathsf{crs}' = u \oplus v$, $com' = u$, and for $b \in \{0,1\}$, set $d'_b = s_b$.

Clearly S_{Eq} can open both 0 and 1 by choosing d'_0 or d'_1 respectively. Moreover, for any algorithm distinguishing between real transcript and interaction with S_{Eq} we can present a distinguisher which breaks the security of G with same advantage.

This can be achieved by replacing the string v by the challenge string in the pseudo-random generator experiment.

7.2.6 One-to-one Equivocal Commitment

We present a modification of the above scheme that allows us to achieve an equivocal commitment scheme with the one-to-one property: for every statistically binding commitment, there is at most a single opening string that will be accepted by the receiver during the decommitment phase. As an underlying ingredient, we use any commitment scheme $\Pi = (\mathsf{CRSGen}_\Pi, \mathsf{Commit}_\Pi, \mathsf{Open}_\Pi)$ *(not necessarily equivocal) with the above property.*

Let $\lambda > 0$ *be an integer, let* $G_1 : \{0,1\}^{\lambda'} \to \{0,1\}^{3\lambda'}$ *and* $G_2 : \{0,1\}^{\lambda} \to \{0,1\}^{t\cdot\lambda'}$ *be pseudo-random generators.*

- $\mathsf{CRSGen}(1^\lambda)$*: Run* $\mathsf{CRSGen}_\Pi(1^\lambda)$ *to generate* crs_Π*. Output* $\mathsf{crs} = \mathsf{crs}_\Pi, \mathsf{crs}_1, \mathsf{crs}_2$ *where* $\mathsf{crs}_1, \mathsf{crs}_2$ *are uniform random strings of length* 3λ*.*

- $\mathsf{Commit}_{\mathsf{crs}}(b)$*: Choose uniform random seeds* $s_1, s_2 \in \{0,1\}^\lambda$ *and compute* $t_1 = G(s_1)$*,* $t_2 = G(s_2)$*. Choose* $\beta \in \{0,1\}$*. Set* $c^1 = t_1 \oplus (b \cdot \mathsf{crs}_1)$*. Set* $c^2 = t_2 \oplus (\beta \cdot \mathsf{crs}_2)$*. Generate* $(com_\beta, d_\beta) = \mathsf{Commit}_{\mathsf{crs}_\Pi}(s_1||s_2)$ *and* $(com_{1-\beta}, d_{1-\beta}) = \mathsf{Commit}_{\mathsf{crs}_\Pi}(0^{2n})$*. Output commitment* $com := (c^1, c^2, com_0, com_1)$*. Output decommitment information* $s := (s_1, s_2, d_\beta)$*.*

- $\mathsf{Open}_{\mathsf{crs}}(com, s)$*: Parse* $com = (c^1, c^2, com_0, com_1)$ *and* $s = s_1||s_2||d$*. If* $c^2 = G(s_2)$*, set* $\beta = 0$*. If* $c^2 = G(s_2) \oplus \mathsf{crs}_2$*, set* $\beta = 1$*. Run* $\mathsf{Open}_{\mathsf{crs}_\Pi}(com_\beta, d)$ *and check that it outputs* $s_1||s_2$*. Otherwise, output* $\perp$*. If* $c^1 = G(s_1)$*, output* 0*. If*

$c^1 = G(s_1) \oplus \mathsf{crs}_1$, output 1. Output $\perp$ otherwise.

Clearly, by the binding of the original commitment scheme and the unique string decommitment property of Π, the modified scheme has the unique string decommitment property.

To create equivocal commitments/openings one can do the following: Run $\mathsf{CRSGen}_\Pi(1^\lambda)$ to generate crs_Π. Choose uniform random seeds $s_1^0, s_1^1, s_2^0, s_2^1 \in \{0,1\}^\lambda$ and compute $t_1^0 = G(s_1^0)$, $t_2^0 = G(s_2^0)$, $t_1^1 = G(s_1^1)$, $t_2^1 = G(s_2^1)$. Choose $\beta \leftarrow \{0,1\}$ Generate $(com_\beta, d_\beta) = \mathsf{Commit}_{\mathsf{crs}_\Pi}(s_1^0||s_2^\beta)$ and $(com_{1-\beta}, d_{1-\beta}) = \mathsf{Commit}_{\mathsf{crs}_\Pi}(s_1^1||s_2^{1-\beta})$. Set $c^1 = t_1^0$. Set $c^2 = t_2^0$. Set $\mathsf{crs}_1 = c^1 \oplus t_1^1$. Set $\mathsf{crs}_2 = c^2 \oplus t_2^1$. Output commitment $com' := (c^1, c^2, com_0, com_1)$.

To open the commitment to a 0, output $(s_1^0||s_2^\beta||d_\beta)$, where d_β is the decommitment information for com_β.

To open the commitment to a 1, output $(s_1^1||s_2^{1-\beta}||d_{1-\beta})$, where $d_{1-\beta}$ is the decommitment information for $com_{1-\beta}$.

7.2.7 Equivocal Commitment (with extra properties) in the CRS model

Let $\Pi' = (\mathsf{Gen}'_{Com}, \mathsf{Com}', \mathsf{Open}', \mathsf{S}'_{Eq})$, be an equivocal, one-to-one ℓ-bit commitment scheme in the CRS model (given in Section 7.2.6). Let $(\mathsf{Gen}_{\mathsf{Sign}}, \mathsf{Sign}, \mathsf{Verify})$ be a strong, one-time signature scheme. We construct $\Pi = (\mathsf{Gen}_{Com}, \mathsf{Com}, \mathsf{Open}, \mathsf{S}_{Eq})$, which is an equivocal commitment scheme, with several additional properties that we describe at the end of the section and which will be useful for our constructions in

Sections 7.4 and 7.5.

Key generation Gen_{Com} ***is as follows:*** *On input security parameter 1^λ, run Gen'_{Com} $2t$ times to generate t pairs of CRS's $[(\Sigma^{0,i}_{Eq}, \Sigma^{1,i}_{Eq})]_{i\in[t]}$, where t is the length of the verification key vk output by $\mathsf{Gen}_{\mathsf{Sign}}$.*

Commitment Com ***is as follows:*** *To commit to a message m of length ℓ, generate a key pair $(\mathsf{vk}, \mathrm{SK}) \leftarrow \mathsf{Gen}_{\mathsf{Sign}}$. For $i \in [t]$, generate $(com_i, d_i) \leftarrow \mathsf{Com}'(\Sigma^{\mathsf{vk}_i,i}, m)$, where com_i is the commitment and d_i is the decommitment information. Generate $\sigma \leftarrow \mathsf{Sign}_{\mathrm{SK}}([com_i]_{i\in[t]})$. Output commitment $c = (\mathsf{vk}, [com_i]_{i\in[\ell]}, \sigma)$. A sender can decommit separately to any set of bits of the message m. Decommitment information for a set S of message bits consists of $d[S] = [d_{i,j}]_{i\in[t],j\in[S]}$, where $d_{i,j}$ is the decommitment information contained in d_i corresponding to the j-th bit.*

Decommitment Open ***w.r.t. a set*** S***:*** *Given a set S, a commitment com, and an opening $[d_{i,j}]_{i\in[t],j\in S}$,* Open *does the following: Parse commitment as $(\mathsf{vk}, [com_{i,j}]_{i\in[t],j\in[\ell]}, \sigma)$. (1) Check that $\mathsf{Verify}_{\mathsf{vk}}([com_{i,j}]_{i\in[t],j\in[\ell]}, \sigma) = 1$ (2) For $i \in [t]$, $j \in S$, check that $d_{i,j}$ is a valid decommitment for $com_{i,j}$ w.r.t. CRS $\Sigma^{\mathsf{vk}_i,i}$.*

Equivocal CRS generation and commitment S_{Eq} ***is as follows:*** *On input security parameter 1^λ, generate a key pair $(\mathsf{vk}, \mathrm{SK}) \leftarrow \mathsf{Gen}_{\mathsf{Sign}}$. Run S'_{Eq} t times to generate $[\Sigma^{\mathsf{vk}_i,i}]_{i\in[t]}$, equivocal commitments $[com_i]_{i\in[t]}$ and decommitments $[(d^0_{i,j}, d^1_{i,j})]_{i\in[t],j\in[\ell]}$. Run Gen'_{Com} t times to generate $[\Sigma^{1-\mathsf{vk}_i,i}]_{i\in t}$.*

Set $\Sigma_{Eq} := [(\Sigma_{Eq}^{0,i}, \Sigma_{Eq}^{1,i})]_{i \in [t]}$. *Compute* $\sigma \leftarrow \mathsf{Sign}_{\mathrm{SK}}([com_i]_{i\in[t]})$.

Output $(\mathsf{crs} = \Sigma_{Eq}, c = (\mathsf{vk}, [com_i]_{i\in[\ell]}, \sigma), d^0 = [d^0_{i,j}]_{i\in[t], j\in[\ell]}, d^1 = [d^1_{i,j}]_{i\in[t], j\in[\ell]})$.

Additional check ***functionality:*** *Given a* Σ *and commitments* $com = (\mathsf{vk}, [com_i]_{i\in[\ell]}, \sigma)$, $com' = (\mathsf{vk}', [com'_i]_{i\in[\ell]}, \sigma')$, $\mathsf{check}_\Sigma(com, com')$ *outputs* 1 *if* *(1)* $\mathsf{vk} = \mathsf{vk}'$*; (2)* $\mathsf{Verify}_{\mathsf{vk}}([com'_i]_{i\in[\ell]}, \sigma') = 1$.

Additional properties:

1. *With overwhelming probability over generation of* Σ*, for every set* $S \subseteq [\ell]$ *and every string com, there is at most a* single *string* $d[S]$ *such that* $\mathsf{Open}_\Sigma(S, com, d[S]) = 1$. *This property is achieved by using the equivocal, one-to-one, commitment scheme given in Section 7.2.6 as the underlying commitment scheme.*

2. *Given a pair* (Σ, com)*, a PPT adversary outputs* com' *such that* $com \neq com'$ *but* $\mathsf{check}_\Sigma(com, com') = 1$ *with negligible probability. This property follows from the security of the one-time signature scheme.*

3. *Given equivocal commitment* $(\Sigma_{Eq}, \overline{com})$*, for every string* com'*, if* $\mathsf{check}_{\Sigma_{Eq}}(\overline{com}, com') = 0$ *then (with overwhelming probability over generation of* Σ_{Eq}*)* com' *has at most one valid opening. Specifically, for every set* $S \subseteq [\ell]$*, there is at most a* single *string* $d[S]$ *such that* $\mathsf{Open}_{\Sigma_{Eq}}(S, com', d[S]) = 1$. *Again, this property is achieved by using the equivocal, one-to-one, commitment scheme given in Section 7.2.6 as the underlying commitment scheme.*

7.3 Impossibility of CNMC with no CRS

In this section we present Theorem 7.3.1, stating the impossibility of constructing CNMC without CRS.

Theorem 7.3.1. *There is no black-box reduction from a single-bit CNMC scheme* $\Pi = (\mathsf{E}, \mathsf{D})$ *to any falsifiable assumption, unless the assumption is false.*

We know from prior work that continuous NMC are impossible in the info-theoretic setting. Assume we have a construction of single-bit, continuous NMC from some falsifiable assumption with no CRS. We only allow black-box usage of the adversary in the reduction. However, the underlying assumption can be used in a non-black-box way in the construction/proof.

Preliminaries. *Given adversary* $A = (A_L, A_R)$*, we say that* A *has advantage* α *in the* simplified no-Σ CNMC game *against construction* $\Pi = (\mathsf{E}, \mathsf{D})$ *if:*

$$\Big|\Pr[\mathsf{D}(A_L(L), A_R(R)) \neq \perp \mid (L, R) \leftarrow \mathsf{E}(1^n, 0)] \\ - \Pr[\mathsf{D}(A_L(L), A_R(R)) \neq \perp \mid (L, R) \leftarrow \mathsf{E}(1^n, 1)]\Big| = \alpha,$$

Clearly, if $A = (A_L, A_R)$ *has non-negligible advantage in the* simplified no-Σ CNMC game, *it can be used to break the CNMC security of* $\Pi = (\mathsf{E}, \mathsf{D})$.

Definition 7.3.1. *A tuple* (x, y, z) *is* bad *relative to CNMC scheme* $\Pi = (\mathsf{E}, \mathsf{D})$ *if either:*

- $y \neq z \wedge \mathsf{D}(x, y) \neq \perp \wedge \mathsf{D}(x, z) \neq \perp$ *OR*

- $x \neq y \wedge \mathsf{D}(x, z) \neq \perp \wedge \mathsf{D}(y, z) \neq \perp$.

Definition 7.3.2. *A single-bit CNMC* $\Pi = (\mathsf{E}, \mathsf{D})$ *in the standard (no CRS model) is* perfectly unique *if there exist no bad tuples relative to* $\Pi = (\mathsf{E}, \mathsf{D})$.

We next present the following two lemmas, which, taken together, imply Theorem 7.3.1.

Lemma 7.3.1. *If a single-bit CNMC scheme* $\Pi = (\mathsf{E}, \mathsf{D})$ *is* not *perfectly unique then it is insecure.*

This is immediate, since if a bad tuple exists, it can be given to the adversary as non-uniform advice. Then the same attack from the literature (reviewed in the introduction) can be run.

Lemma 7.3.2. *There is no BB reduction from a single-bit CNMC scheme* $\Pi = (\mathsf{E}, \mathsf{D})$ *which is perfectly unique to any falsifiable assumption.*

The basic idea is that, given only black-box access to the split-state adversary, $A = (A_L, A_R)$*, the reduction cannot tell the difference between the actual adversary and a* simulated *adversary. The simulated adversary simply waits to get matching L and R queries from the reduction, decodes, and re-encodes a fresh value that is related to the decoded value. The challenges are that the L and R queries are not received simultaneously. In fact, there could be many queries interleaved between a L and R match. So the simulated adversary must return a value upon seeing the L or R half* before *seeing the other half and* before *knowing whether the encoded value is a* 0 *or a* 1*. Therefore, the simulated adversary does the following: It keeps a* *table containing all the L and R values that it has seen. Whenever a L or R query is made, the simulated adversary first checks the table to see if a matching query was previously made. If not, the simulated adversary chooses a random encoding, (L', R'), of a random bit b', stores it in the table along with the L/R query that was made and returns either L' or R' as appropriate. If yes, the simulated adversary finds the corresponding L/R along with the pair (L', R') stored in the table. The simulated adversary then decodes (L, R) to find out b. If $b = 0$, the simulated adversary returns either L' or R' as appropriate. Otherwise, the simulated adversary returns the left/right side of an encoding of a random bit b''. We prove that the view generated by the reduction interacting with this adversary is identical to the view of the reduction interacting with the following real adversary: The real adversary, given L or R, recovers the corresponding unique valid codeword (L, R) (if it exists) and decodes to get the bit b. If $b = 0$, the real adversary encodes a random bit $b' =$ $\mathsf{RO}_1(L||R)$ using randomness $r = \mathsf{RO}_2(L||R)$ (where $\mathsf{RO}_1, \mathsf{RO}_2$ are random oracles internal to the real adversary that are used to generate consistent randomness across invocations) and outputs the left/right side as appropriate. Otherwise (i.e. if the corresponding unique codeword does not exist or if $\mathsf{D}(L, R) = 1$), the real adversary outputs the left/right side of encoding of a random bit, $b'' = \mathsf{RO}_3(L)$ (or $b'' =$ $\mathsf{RO}_3(R)$) using randomness $r'' = \mathsf{RO}_4(L)$ (or $r'' = \mathsf{RO}_4(R)$) (where $\mathsf{RO}_3, \mathsf{RO}_4$ are random oracles internal to the real adversary that are used to generate consistent randomness across invocations). Note that since the CNMC is perfectly unique, the real adversary obtains non-negligible advantage of $1 - \mathsf{negl}(n)$ in the simplified no-Σ CNMC game.*

Proof. We will construct a *meta-reduction* as follows:

Consider the following inefficient, split state adversary $A = (A_L, A_R)$ with internal random oracles $\mathsf{RO}_1, \mathsf{RO}_2$, RO_3, and RO_4:

A_L: On input L, find the unique R such that $\mathsf{D}(L, R) \neq \perp$ (if it exists). Let $b := \mathsf{D}(L, R)$. If $b = 0$, encode $b' = \mathsf{RO}_1(L||R)$ using randomness $r = \mathsf{RO}_2(L||R)$ to obtain $(L', R') := \mathsf{E}(b'; r)$ and output L'. If such R does not exist or if $b = 1$, compute a random encoding of a random bit $b'' = \mathsf{RO}_3(L)$ using randomness $r'' = \mathsf{RO}_4(L)$ to obtain $(L'', R'') := \mathsf{E}(b'', r'')$ and output L''.

A_R: On input R, find the unique L such that $\mathsf{D}(L, R) \neq \perp$ (if it exists). Let $b := \mathsf{D}(L, R)$. If $b = 0$, encode $b' = \mathsf{RO}_1(L||R)$ using randomness $r = \mathsf{RO}_2(L||R)$ to obtain $(L', R') := \mathsf{E}(b'; r)$ and output R'. If such L does not exist or if $b = 1$, compute a random encoding of a random bit $b'' = \mathsf{RO}_3(R)$ using randomness $r'' = \mathsf{RO}_4(R)$ to obtain $(L'', R'') := \mathsf{E}(b'', r'')$ and output R''.

Clearly, A succeeds with advantage $1 - \mathsf{negl}(n)$ in the simplified no-Σ CNMC game.

The following adversary A' simulates the above efficiently: Let T be a table that records internal randomness. T is initialized to empty. A' is a stateful adversary that proceeds as follows:

1. On input L, check if the corresponding R such that $\mathsf{D}(L, R) \neq \perp$ has been queried. If yes, decode to get bit $b := \mathsf{D}(L, R)$. If $b = 0$, check the table T to recover (R, L', R'). Output L'. Otherwise, if $L \in T$ then output L''

corresponding to entry (L, L'', R''). If $L \notin T$, choose a random encoding of a random bit b'': $(L'', R'') \leftarrow \mathsf{E}(b'')$. Store (L, L'', R'') in T. and output L''.

2. On input R, check if the corresponding L such that $\mathsf{D}(L, R) \neq \perp$ has been queried. If yes, decode to get bit $b := \mathsf{D}(L, R)$. If $b = 0$, check the table T to recover (L, L', R'). Output R'. Otherwise, if $R \in T$ then output R'' corresponding to entry (R, L'', R''). If $R \notin T$, choose a random encoding of a random bit b'': $(L'', R'') \leftarrow \mathsf{E}(b'')$. Store (R, L'', R'') in T and output R''.

By properties of the random oracle, the view of the reduction **Red** when interacting with A versus A' are equivalent.

Since the reduction succeeds when interacting with Real adversary A with non-negligible probability p and since the view of the reduction is identical when interacting with A or A', **Red** interacting with A' must also succeed with non-negligible probability p. But **Red** composed with A' yields an efficient adversary, leading to an efficient adversary breaking the underlying falsifiable assumption, which is a contradiction. □

7.4 2-State CNMC for One-Bit Messages

In this section we prove the following theorem:

Theorem 7.4.1. *Assuming the existence of one-to-one commitment schemes in the CRS model, there is a construction of a 2-split-state CNM Randomness Encoder in the CRS model.*

The corollary is immediate, given the following transformation.

7.4.1 CNM Randomness Encoder to Single-Bit CNMC

Let $\Pi = (\mathsf{CRSGen}, \mathsf{E} = (\mathsf{E}_1, \mathsf{E}_2), \mathsf{D})$ *be a CNM Randomness Encoder. To construct a CNMC for a single bit from* Π*, we must show how to use* E *to encode a message* $b \in \{0,1\}$*. In the case of the CNM Randomness Encoder given in Section 7.4, this can be done by choosing random* $c_L, c_R \in \mathbb{F}_{2^\lambda}^{\frac{\ell}{\lambda}}$*, conditioned on the parity of* $\langle c_L, c_R \rangle$ *being equal to* b*. and then running* $\mathsf{E}(c_L || c_R || r_{enc} || r_L || r_R)$ *In general, one can run* $\mathsf{E}(r)$ *repeatedly until* $\mathsf{E}_2(r)$ *outputs a random message* m *with parity equal to* b*. (this will give an encode algorithm that runs in polynomial time with all but negligible probability).*

Now, it can be immediately seen that an adversary who breaks the security formulation of CNMC given in Definition 3.2.11 must also break the security of the CNM Randomness Encoder, given in Definition 7.2.1.

Corollary 7.4.1. *Assuming the existence of one-to-one commitment schemes in the CRS model, there is a construction of a single-bit, 2-split-state CNMC in the CRS model.*

Notation and parameters. λ *is security parameter and length of encoded randomness.* $\ell = \ell(\lambda) \in \Theta(\lambda^2)$ *and we assume for simplicity that* $\lambda | \ell$*. Sets* $S_L, S_R \subseteq [2\ell]$ *are defined as follows:* $S_L = [\ell], S_R = [2\ell] \setminus [\ell]$*.* $y_o = y_o(\ell) \in \Theta(\ell^{1/2})$*,* $y_t = y_t(\ell) \in \Theta(\ell^{1/2})$*.*

The construction of the 2-state CNM Randomness Encoder is given in Figure 7.1.

To prove Theorem 7.4.1, we show that the construction above is a secure CNM

Let $(\mathsf{CRSGen}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Open}, \mathsf{S}_{Eq})$ be the non-interactive, equivocal, one-to-one commitment in the CRS model given in Section 7.2.7.

$\mathsf{CRSGen}(1^\lambda)$: $\Sigma \leftarrow \mathsf{CRSGen}_{\mathsf{Com}}(1^\lambda)$. Output Σ.

$\mathsf{E}_\Sigma(c_L||c_R||r_{com})$:

1. Parse c_L, c_R as strings in $\mathbb{F}_{2^\lambda}^{\frac{\ell}{\lambda}}$.
2. $(com, d = d_1, \ldots, d_{2\ell}) \leftarrow \mathsf{Com}_\Sigma(c_L||c_R; r_{com})$
3. Let $d[S_L]$ (resp. $d[S_R]$) correspond to the decommitment of com to the bits corresponding to S_L (resp. S_R).
4. $\mathsf{E}_{2,\Sigma}$ outputs $L = (com, d[S_L])$; $R = (com, d[S_R])$. $\mathsf{E}_{1,\Sigma}$ outputs $\langle c_L, c_R \rangle$.

$\mathsf{D}_\Sigma(\widetilde{L}, \widetilde{R})$:

1. Parse $\widetilde{L} = (\widetilde{com}, \widetilde{d}[S_L])$, $\widetilde{R} = (\widetilde{com}', \widetilde{d}[S_R])$.
2. Check that $\widetilde{com} = \widetilde{com}'$.
3. Let $\widetilde{c}_L = \mathsf{Open}_\Sigma(S_L, \widetilde{com}, \widetilde{d}[S_L])$ and $\widetilde{c}_R = \mathsf{Open}_\Sigma(S_R, \widetilde{com}, \widetilde{d}[S_R])$. Check that $\widetilde{c}_L \neq \perp$ and $\widetilde{c}_R \neq \perp$.
4. If all the above checks pass, output $\langle \widetilde{c}_L, \widetilde{c}_R \rangle$. Otherwise, output $\perp$.

Figure 7.1: Construction of 2-State, Continuous, Non-Malleable Randomness Encoder.

Randomness Encoder, via the following sequence of hybrids.

Hybrid 0: *This is the "Real" security experiment.*

Hybrid 1: *The experiment is identical to Hybrid* 0 *except we modify the decode algorithm from* D_Σ *to* D^1_Σ *to abort if the tampered codeword submitted is different from the challenge codeword and the* check *function outputs* 1*. Specifically, let* $(L := (com, d[S_L]), R = (com, d[S_R]))$ *be the "challenge" codeword (i.e. the codeword generated by the security experiment).*

```
D^1_Σ(L̃, R̃):
    1. Parse $\widetilde{L} = (\widetilde{com}, \widetilde{d}[S_L])$, $\widetilde{R} = (\widetilde{com}', \widetilde{d}[S_R])$.
    2. If $\widetilde{L} \neq L$ and $\mathsf{check}_\Sigma(com, \widetilde{com}) = 1$ or $\widetilde{R} \neq R$ and $\mathsf{check}_\Sigma(com, \widetilde{com}') = 1$ then output $\perp$.
    3. Check that $\widetilde{com} = \widetilde{com}'$.
    4. Let $\widetilde{c}_L = \mathsf{Open}_\Sigma(S_L, \widetilde{com}, \widetilde{d}[S_L])$ and $\widetilde{c}_R = \mathsf{Open}_\Sigma(S_R, \widetilde{com}, \widetilde{d}[S_R])$. Check that $\widetilde{c}_L \neq \perp$ and $\widetilde{c}_R \neq \perp$.
    5. If all the above checks pass, output $\langle \widetilde{c}_L, \widetilde{c}_R \rangle$. Otherwise, output $\perp$.
```

Figure 7.2: Decode in Hybrid 1.

Hybrid 2: *The experiment is identical to Hybrid* 1*, except we switch to equivocal commitments in the codeword* (L, R) *that is given to the adversary. Specifically,* CRSGen *is replaced with* CRSGen^2 *and the challenge codeword is generated as shown in Figure 7.3.*

Hybrid 3: *The experiment is identical to Hybrid* 2*, except we modify* D^1 *to* D^3*, which aborts if the outcome of* $f^i_L(L)$ *or* $f^i_R(R)$ *is not a "likely value."*

$\underline{\mathsf{CRSGen}^2(1^\lambda)\text{: } (\Sigma_{Eq}, \overline{com}, d^0 = d^0_1 \ldots d^0_{2\ell}, d^1 = d^1_1 \ldots d^1_{2\ell}) \leftarrow \mathsf{S}_{Eq}(1^\lambda). \text{ Output } \Sigma_{Eq}.}$
Challenge codeword:

1. Sample c_L, c_R uniform randomly from $\mathbb{F}^{\frac{\ell}{\lambda}}_{2^\lambda}$.
2. $\underline{\text{Set } d[S_L] := [d_i^{c_L[i]}]_{i \in S_L}; \text{ Set } d[S_R] := [d_i^{c_R[i]}]_{i \in S_R};}$
3. Output $L = (\overline{com}, d[S_L])$; $R = (\overline{com}, d[S_R])$.

Figure 7.3: Gen and Challenge Codeword generation in Hybrid 2.

Specifically, given $(\Sigma_{Eq}, \overline{com}, d^0 = d^0_1 \ldots d^0_{2\ell}, d^1 = d^1_1 \ldots d^1_{2\ell})$ *and the adversary's current output* $\mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1}$*, we define the sets* $\mathcal{S}_L$*,* $\mathcal{S}_R$*,* $\mathcal{S}'_L$*,* $\mathcal{S}'_R$ *as follows:*

- $\mathcal{S}_L$ *contains all values of* $\widehat{L}'$ *that occur with probability at least* $\epsilon = 1/2^{y_o/3}$*, where values of* $\widehat{L}'$ *are sampled as follows: Sample* $\widehat{c}_L$ *conditioned on the output of the experiment in Hybrid 2 thus far being equal to* $\mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1}$*. Compute equivocal decommitment of* $\overline{com}$*:* $\widehat{d}[S_L] := [d_i^{\widehat{c}_L[i]}]_{i \in S_L}$*. Apply* f^i_L *to* $\widehat{L} = (\overline{com}, \widehat{d}[S_L])$ *to obtain* $\widehat{L}'$ *(or "same" if the output is* $\widehat{L}$ *itself).*

- $\mathcal{S}_R$ *contains all values of* $\widehat{R}'$ *that occur with probability at least* $\epsilon = 1/2^{y_o/3}$*, where values of* $\widehat{R}'$ *are sampled as follows: Sample* $\widehat{c}_R$ *conditioned on the output of the experiment in Hybrid 2 thus far being equal to* $\mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1}$*. Compute equivocal decommitment of* $\overline{com}$*:* $\widehat{d}[S_R] := [d_i^{\widehat{c}_R[i]}]_{i \in S_R}$*. Apply* f^i_R *to* $\widehat{R} = (\overline{com}, \widehat{d}[S_R])$ *to obtain* $\widehat{R}'$ *(or "same" if the output is* $\widehat{R}$ *itself).*

- *Let* $\mathcal{S}'_L \subseteq \mathcal{S}_L$ *be the set of* $\widehat{L}'$ *such that there is a "matching"* $\widehat{R}' \in \mathcal{S}_R$ *such that* $\mathsf{D}^1_{\Sigma_{Eq}}(\widehat{L}', \widehat{R}') \neq \perp$.

- *Let* $\mathcal{S}'_R \subseteq \mathcal{S}_R$ *be the set of* $\widehat{R}'$ *such that there is a "matching"* $\widehat{L}' \in \mathcal{S}_L$ *such that* $\mathsf{D}^1_{\Sigma_{Eq}}(\widehat{L}', \widehat{R}') \neq \perp$.

Note that the decode oracle is now stateful and depends on the current round of interaction, as well as the outputs returned in previous rounds. Specifically, note that the sets $\mathcal{S}'_L$, $\mathcal{S}'_R$ change in each round i, since the likely outputs depend on the tampering function (f^i_L, f^i_R) submitted by the adversary in round i, and are conditioned on the output $\mathbf{out}^{i-1}_A = \widehat{Out}^{i-1}_A$ seen by the adversary thus far in rounds $1, \ldots, i-1$.

$\mathsf{D}^3_{\Sigma_{Eq}}((f^i_L, f^i_R), \widetilde{L}, \widetilde{R})$:

1. $\underline{\text{Check that } \widetilde{L} \in \mathcal{S}'_L \text{ and that } \widetilde{R} \in \mathcal{S}'_R\text{. If not, output } \perp.}$
2. Parse $\widetilde{L} = (\widetilde{com}, \widetilde{d}[S_L])$, $\widetilde{R} = (\widetilde{com}', \widetilde{d}[S_R])$.
3. Check that $\widetilde{com} = \widetilde{com}'$.
4. Let $\widetilde{c}_L = \mathsf{Open}_\Sigma(S_L, \widetilde{com}, \widetilde{d}[S_L])$ and $\widetilde{c}_R = \mathsf{Open}_\Sigma(S_R, \widetilde{com}, \widetilde{d}[S_R])$. Check that $\widetilde{c}_L \neq \perp$ and $\widetilde{c}_R \neq \perp$.
5. If all the above checks pass, output $\langle \widetilde{c}_L, \widetilde{c}_R \rangle$. Otherwise, output $\perp$.

Figure 7.4: Decode in Hybrid 3.

Hybrid 4: *The experiment is identical to Hybrid 3, except we modify* D^3 *to* D^4 *which aborts if there are more than y_t number of queries f^i_L (resp. f^i_R) such that the outcome of $f^i_L(L)$ (resp. $f^i_R(R)$) is not the most "likely value". Specifically, at the beginning of the experiment, we initialize counters* $\mathsf{count}_L, \mathsf{count}_R$ *to* 0*. We also define L^* (resp. R^*) to be the element of $\mathcal{S}'_L$ (resp. $\mathcal{S}'_R$) that occurs most frequently. More precisely, we consider the sets*

$$\mathcal{L}^* := \mathsf{argmax}_{L' \in \mathcal{S}'_L} \Pr[f^i_L(\widehat{L}) = L' \mid \mathbf{out}^{i-1}_A = \widehat{Out}^{i-1}_A].$$

$$\mathcal{R}^* := \mathsf{argmax}_{R' \in \mathcal{S}'_R} \Pr[f^i_R(\widehat{R}) = R' \mid \mathbf{out}^{i-1}_A = \widehat{Out}^{i-1}_A].$$

Then L^* (resp. R^*) is defined to be the lexicographically first element in $\mathcal{L}^*$ (resp. $\mathcal{R}^*$).

```
$\mathsf{D}^4_{\Sigma_{Eq}}((f^i_L, f^i_R), \widetilde{L}, \widetilde{R})$:
    1. Check that $\widetilde{L} \in \mathcal{S}'_L$ and that $\widetilde{R} \in \mathcal{S}'_R$. If not, output $\perp$.
    2. $\underline{\text{If } \widetilde{L} \neq L^*, \text{ then set } \mathsf{count}_L := \mathsf{count}_L + 1.}$
    3. $\underline{\text{If } \widetilde{R} \neq R^*, \text{ then set } \mathsf{count}_R := \mathsf{count}_R + 1.}$
    4. $\underline{\text{If } \mathsf{count}_L > y_t \text{ or } \mathsf{count}_R > y_t, \text{ output } \perp.}$
    5. Parse $\widetilde{L} = (\widetilde{com}, \widetilde{d}[S_L])$, $\widetilde{R} = (\widetilde{com}', \widetilde{d}[S_R])$.
    6. Check that $\widetilde{com} = \widetilde{com}'$.
    7. Let $\widetilde{c}_L = \mathsf{Open}_\Sigma(S_L, \widetilde{com}, \widetilde{d}[S_L])$ and $\widetilde{c}_R = \mathsf{Open}_\Sigma(S_R, \widetilde{com}, \widetilde{d}[S_R])$. Check that $\widetilde{c}_L \neq \perp$ and $\widetilde{c}_R \neq \perp$.
    8. If all the above checks pass, output $\langle \widetilde{c}_L, \widetilde{c}_R \rangle$. Otherwise, output $\perp$.
```

Figure 7.5: Decode in Hybrid 4.

Claim 7.4.1. Hybrids 0 and 1 are computationally indistinguishable.

This follows from the additional properties of the equivocal commitment scheme given in Section 7.2.7.

Claim 7.4.2. Hybrids 1 and 2 are computationally indistinguishable.

This follows from the security of the equivocal commitment scheme.

Claim 7.4.3. Hybrids 2 and 3 are $\epsilon \cdot 2q$-close, where $\epsilon = 1/2^{y_o/3}$ and $y_o \in O(\ell^{1/2})$.

Proof. To prove indistinguishability of Hybrids 2 and 3, it is sufficient to show that for each $i \in [q]$, $\Pr[f^i_L(L) \notin \mathcal{S}'_L \wedge \mathsf{D}^1_{\Sigma_{Eq}}(f^i_L(L), f^i_R(R)) \neq \perp] \leq \epsilon$ and $\Pr[f^i_L(R) \notin \mathcal{S}'_R \wedge \mathsf{D}^1_{\Sigma_{Eq}}(f^i_L(L), f^i_R(R)) \neq \perp] \leq \epsilon$. The result then follows by a union bound over the q LHS and q RHS queries.

To bound the above, we in fact show something stronger: (1) for each $i \in [q]$, each value of $\mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1}$ (which does not contain a $\perp$ output) and each value of $R = \widehat{R}$,

$$\Pr[f_L^i(L) \notin \mathcal{S}'_L \wedge \mathsf{D}^1_{\Sigma_{Eq}}(f_L^i(L), f_R^i(R)) \neq \perp \mid R = \widehat{R} \wedge \mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1})] \leq \epsilon;$$

and (2) for each $i \in [q]$, each value of $\mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1}$ (which does not contain a $\perp$ output) and each value of $L = \widehat{L}$,

$$\Pr[f_R^i(R) \notin \mathcal{S}'_R \wedge \mathsf{D}^1_{\Sigma_{Eq}}(f_L^i(L), f_R^i(R)) \neq \perp \mid L = \widehat{L} \wedge \mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1})] \leq \epsilon.$$

We first fix $(\Sigma_{Eq}, \overline{com}, d^0 = d^0_1 \ldots d^0_{2\ell}, d^1 = d^1_1 \ldots d^1_{2\ell})$. Note that for fixed $\Sigma_{Eq}, \overline{com}, d^0 = d^0_1 \ldots d^0_{2\ell}, d^1 = d^1_1 \ldots d^1_{2\ell}$, there is a bijection ϕ_L (resp. ϕ_R) between c_L (resp. c_R) and $(\overline{com}, d[S_L])$ (where $d[S_L] := [d_i^{c_L[i]}]_{i \in S_L}$). Therefore the probability of a particular value of c_L (resp. c_R) occurring is equivalent to the probability of $L = \phi_L(c_L)$ (resp. $R = \phi_R(c_R)$) occurring. Additionally, Let ρ_L (resp. ρ_R) be the function that given $f_R^i(R)$ (resp. $f_L^i(L)$) returns the unique L' (resp. R') if it exists such that, $\mathsf{D}^1_{\Sigma_{Eq}}(L', f_R^i(R)) \neq \perp$ (resp. $\mathsf{D}^1_{\Sigma_{Eq}}(f_L^i(L), R') \neq \perp$). Note that L' (resp. R') is equal to "same" if and only if $f_R^i(R) =$ "same" (resp. $f_L^i(L) =$ "same"). To see why this is so, recall that in D^1, $\perp$ is outputted if $\widetilde{L} \neq L$ and $\mathsf{check}_\Sigma(com, \widetilde{com}) = 1$ or $\widetilde{R} \neq R$ and $\mathsf{check}_\Sigma(com, \widetilde{com}') = 1$. Now, if L' is equal to same, then it must be that $\mathsf{check}_\Sigma(com, \widetilde{com}) = 1$. Therefore, by the above, the only value of $f_R^i(R)$, for which $\perp$ will not be output is $f_R^i(R) =$ "$same''$. The same is true for the case that

$f_R^i(R) = \text{“}same''$.

We first show that for $i \in [q]$, c_L, c_R are conditionally independent given $\mathbf{out}_A^i = \widehat{Out}_A^i$. This follows from the fact that the information contained in $\widehat{Out}_A^i$ is of the form $(f_L^1(\phi_L(c_L)) = v_1, f_R^1(\phi_R(c_R)) = w_1), \ldots, (f_L^i(\phi_L(c_L)) = v_i, f_R^i(\phi_R(c_R)) = w_i)$, where for $j \in [i]$, v_j is equal to the L' value outputted in response to the j-th query and w_j is equal to the R' value outputted in response to the j-th query. (note that v_j/w_j can be set to “same” if the tampering function leaves L/R unchanged). Thus, the distribution of c_L, c_R conditioned on $(f_L^1(\phi_L(c_L)) = v_1, f_R^1(\phi_R(c_R)) = w_1), \ldots, (f_L^i(\phi_L(c_L)) = v_i, f_R^i(\phi_R(c_R)) = w_i)$ is equal to $(U_\ell \mid (f_L^1(\phi_L(U_\ell)) = v_1, \ldots, f_L^i(\phi_L(U_\ell)) = v_i)) \times (U_\ell \mid (f_R^1(\phi_R(U_\ell)) = w_1, \ldots, f_R^i(\phi_R(U_\ell)) = w_i))$. Moreover, due to the discussion above, L, R are also conditionally independent given $\mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1}$. Therefore, to show (1), we note that for every $(\widehat{L}, \widehat{R}, \widehat{Out}_A^{i-1})$, $\Pr[L = \widehat{L} \mid R = \widehat{R} \wedge \mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1})] = \Pr[L = \widehat{L} \mid \mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1})]$. So we have that for every fixed $R = \widehat{R}$ (for which $\Pr[R = \widehat{R} \wedge \mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1})] > 0$), and every $L' \notin \mathcal{S}'_L$, $\Pr[f^i(L) = L' \mid R = \widehat{R} \wedge \mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1})] \le \epsilon$. Therefore,

$$\begin{aligned}
&\Pr[f_L^i(L) \notin \mathcal{S}'_L \wedge \mathsf{D}^1_{\Sigma_{Eq}}(f_L^i(L), f_R^i(R)) \neq \perp \mid R = \widehat{R} \wedge \mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1})] \\
&\quad = \Pr[f_L^i(L) \notin \mathcal{S}'_L \wedge \big(f_L^i(L) = \rho_L(f_R^i(R))\big) \mid R = \widehat{R} \wedge \mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1})] \\
&\quad \le \epsilon.
\end{aligned}$$

The proof for (2) is analogous. $\square$

Claim 7.4.4. Hybrids 3 *and* 4 *are statistically indistinguishable.*

Proof. To prove indistinguishability of Hybrids 3 and 4, we must show that the probability that the event (1) $f_L^i(L)$ is not most frequent and $\mathsf{D}^3_{\Sigma_{Eq}}(f_L^i(L), f_R^i(R))$ $\neq \perp$ or event (2) $f_R^i(R)$ is not most frequent and $\mathsf{D}^3_{\Sigma_{Eq}}(f_L^i(L), f_R^i(R)) \neq \perp$ occurs more than y_t times in a single execution is at most $(1/2)^{y_t}$.

We first analyze the event (1). Recall that set $\mathcal{S}'_L$ contains values, L', that occur with probability p in some experiment. By "most frequent value" in $\mathcal{S}'_L$, we mean the value L' in $\mathcal{S}'_L$ with the maximum associated probability p. Note that if L' is not the most frequent value, the associated probability p is at most $1/2$, since otherwise, the probabilities will sum to more than 1. More precisely, if $f_L^i(L) = L'$ is not the most frequent query in $\mathcal{S}'_L$ then, by definition of the set $\mathcal{S}'_L$ and the above argument, $\Pr[f_L^i(\widehat{L}) = L' \mid \mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1}] \leq 1/2$. Recall that in the proof of the previous claim, we have shown that for $i \in \{0, \ldots, q\}$, L, R are conditionally independent given $\mathbf{out}_A^i$. Therefore, $\Pr[f_L^i(L) = L' \mid \mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1} \wedge R = \widehat{R}] \leq 1/2$. This implies that for every fixed $R = \widehat{R}$ (for which $\Pr[R = \widehat{R} \wedge \mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1}] > 0$),

$$
\begin{aligned}
&\Pr[f_L^i(L) \neq L^* \wedge \mathsf{D}^3_{\Sigma_{Eq}}(f_L^i(L), f_R^i(R)) \neq \perp \mid R = \widehat{R} \wedge \mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1})] \\
&\quad \leq \Pr[f_L^i(L) \neq L^* \wedge f_L^i(L) = \rho_L(f_R^i(R)) \mid R = \widehat{R} \wedge \mathbf{out}_A^{i-1} = \widehat{Out}_A^{i-1})] \\
&\quad \leq 1/2.
\end{aligned}
$$

We consider the number of adversarial queries such that both $f_L^i(L) = L'$ is not the most frequent value $(L^*) \in \mathcal{S}'_L$ *and* $\mathsf{D}^3_{\Sigma_{Eq}}(f_L^i(L), f_R^i(R)) \neq \perp$. (note that the total number of adversarial queries can be much higher). By the above argument, the probability that there are y_t number of rounds i such that both $f_L^i(L) = L'$

is not the most frequent value $(L^*) \in \mathcal{S}'_L$ *and* $\mathsf{D}^3_{\Sigma_{Eq}}(f^i_L(L), f^i_R(R)) \neq \perp$ is at most $(1/2)^{y_t} \in \mathsf{negl}(\lambda)$. Thus, we have concluded the proof for event (1). The proof for event (2) is analogous. □

We finally show the main technical claim of this section, which completes the proof of Theorem 7.4.1.

Claim 7.4.5. In Hybrid 4, the encoded randomness $\langle c_L, c_R \rangle$ is statistically close to uniform, given the view of the adversary.

Proof. Towards proving the claim, we consider the following leakage functions:

Leakage function on c_L: Fix $\Sigma_{Eq}, \overline{com}, d^0, d^1$, universal hash $h : \{0,1\}^\alpha \to \{0,1\}^{y_o} \in \mathcal{H}$ (where α is the length of a single split-state of the encoding) and adversary A. On input c_L, set output $\mathbf{out}_A$ to "" and $\mathbf{out}_L$ to "". Set $L = (\overline{com}, [d_i^{c_L[i]}]_{i\in[\ell]})$. Repeat the following in rounds $i = 1, 2, \ldots$:

1. Obtain the next tampering function (f_L, f_R) from adversary A. If A terminates then terminate with output $\mathbf{out}_L$.
2. Set $L' := f_L(L)$. If $L' \in \mathcal{S}'_L$, then:
   (a) Find the unique $\widehat{R}' \in \mathcal{S}'_R$ such that $\mathsf{D}^1_{\Sigma_{Eq}}(L', \widehat{R}') \neq \perp$. Return $(L', \widehat{R}')$ to the adversary. Set $\mathbf{out}_A = \mathbf{out}_A||(L', \widehat{R}')$.
   (b) If L' is not the most frequent output in $\mathcal{S}'_L$, set $\mathbf{out}_L := \mathbf{out}_L||(i||h(L'))$ If $|\mathbf{out}_L| > (\log(q)+y_o)\cdot y_t$ then terminate with output $\mathbf{out}_L := \mathbf{out}_L||(i||\perp)$.

3. If $L' \notin \mathcal{S}'_L$, output $\perp$ to the adversary and terminate with output $\mathbf{out}_L := \mathbf{out}_L||(i||\perp)$.

The leakage function for the RHS is analogous.

We now show that given $\mathbf{out}_L$ and $\mathbf{out}_R$ we can reconstruct the full output sequence for the adversary's view with probability $1 - \frac{2q}{\epsilon^2 \cdot 2^{y_O}} = 1 - \frac{2q}{2^{y_O/3}}$ in the following way:

Fix $\Sigma_{Eq}, \overline{com}, d^0 = d^0_1 \dots d^0_{2\ell}, d^1 = d^1_1 \dots d^1_{2\ell}$, universal hash $h \leftarrow \mathcal{H}$ and adversary A. Set output $\mathbf{out}_A$ to "" and $\mathbf{out}_L$ to "". Repeat the following in rounds $i = 1, 2, \dots, q$:

1. Obtain the next tampering function (f_L, f_R) from adversary A given its current view, $\mathbf{out}_A$.
2. If $(i, \perp) \in \mathbf{out}_L$ or $(i, \perp) \in \mathbf{out}_R$, set $\mathbf{out}_A = \mathbf{out}_A||\perp$ and abort.
3. If $(i, y) \in \mathbf{out}_L$, for some $y \neq \perp$, set $L' = \widehat{L}'$ such that $\widehat{L}' \in \mathcal{S}'_L$ and $h(\widehat{L}') = y$.
4. If $(i, \cdot) \notin \mathbf{out}_L$, set $L' = \widehat{L}'$ such that $\widehat{L}' \in \mathcal{S}'_L$ is the most frequent value.
5. If $(i, y) \in \mathbf{out}_R$, for some $y \neq \perp$, set $R' = \widehat{R}'$ such that $\widehat{R}' \in \mathcal{S}'_R$ and $h(\widehat{R}') = y$.
6. If $(i, \cdot) \notin \mathbf{out}_R$, set $R' = \widehat{R}'$ such that $\widehat{R}' \in \mathcal{S}'_R$ is the most frequent value.
7. If $L' =$ "same" and $R' =$ "same" output "same" and set $\mathbf{out}_A = \mathbf{out}_A||$"same".
8. Else if one of L', R' is "same" and not the other, set $\mathbf{out}_A = \mathbf{out}_A||\perp$ and abort.

9. Else Parse $L' := (com, d[S_L])$ and $R' := (com', d[S_R])$. If $com \neq com'$, set $\mathbf{out}_A = \mathbf{out}_A || \bot$ and abort.

10. Otherwise, set $\mathbf{out}_A = \mathbf{out}_A || (L', R')$.

It can be determined by inspection that the incorrect value is output only if in one of the at most $2q$ instances, there are two distinct values $\widehat{L}', \widehat{L}'' \in \mathcal{S}'_L$ or $\widehat{R}', \widehat{R}'' \in \mathcal{S}'_R$ such that $h(\widehat{L}') = h(\widehat{L}'')$ or $h(\widehat{R}') = h(\widehat{R}'')$. Due to universality of h and the fact that $|\mathcal{S}'_L| = |\mathcal{S}'_R| = 1/\epsilon$, this can occur with probability at most $\frac{2q}{\epsilon^2 \cdot 2^{y_o}}$, as claimed. [2]

Since $|\mathbf{out}_L| \leq (\log(q) + y_o) \cdot y_t \leq 2y_o \cdot y_t \leq c \cdot \ell$ for constant $c < 1$ and $|\mathbf{out}_R| \leq (\log(q) + y_o) \cdot y_t \leq 2y_o \cdot y_t \leq c \cdot \ell$ for constant $c < 1$, we can use the properties of the inner product extractor given in Lemma 7.2.1 to argue that $\langle c_L, c_R \rangle$ is statistically close to uniform random, given $\mathbf{out}_L, \mathbf{out}_R$. Moreover, since we have shown that the view of the adversary in the Hybrid 4 can be fully reconstructed given $\mathbf{out}_L, \mathbf{out}_R$, we have that, in the Hybrid 4, the encoded randomness $\langle c_L, c_R \rangle$ is statistically close to uniform, given the adversary's view in the CNMC experiment. $\square$

7.5 4-State CNMC for Multi-Bit Messages

In this section we prove the following theorem:

Theorem 7.5.1. *Assuming the existence of one-to-one commitment schemes in the CRS model, there is a construction of a multi-bit, 4-split-state CNMC in the CRS*

[2] Recall that $\mathcal{S}'_L \subseteq \mathcal{S}_L$, and $\mathcal{S}_L$ contains all the values of $\hat{L}'$ which occur with probability at least ϵ. Therefore $|\mathcal{S}_L| \leq 1/\epsilon$ (and thus $|\mathcal{S}'_L| \leq 1/\epsilon$), since otherwise the sum of the probabilities would exceed 1. A similar argument is true for $\mathcal{S}'_R$.

model.

Notation and parameters. *λ is security parameter and length of encoded message. $\ell = \ell(\lambda) \in \Theta(\lambda^2)$ and we assume for simplicity that $\lambda | \ell$. $k = 2\lambda$. Sets $S_{L,1}, S_{R,1}, S_{L,2}, S_{R,2} \subseteq [4\ell]$ are defined as follows: $S_{L,1} = [\ell], S_{R,1} = [2\ell] \setminus [\ell]$, $S_{L,2} = [3\ell] \setminus [2\ell], S_{R,2} = [4\ell] \setminus [3\ell]$. $y_o = y_o(\ell) \in \Theta(\ell^{1/2})$, $y_t = y_t(\ell) \in \Theta(\ell^{1/2})$.*

The construction of the multi-bit, 4-state CNMC is presented in Figure 7.6.

Let $(\mathsf{CRSGen}_{\mathsf{Com}}, \mathsf{Com}, \mathsf{Open}, \mathsf{S}_{Eq})$ be the non-interactive, equivocal, one-to-one commitment in the CRS model given in Section 7.2.7.

$\mathsf{CRSGen}'(1^\lambda)$: Run $\Sigma \leftarrow \mathsf{CRSGen}_{\mathsf{Com}}(1^\lambda)$. Output Σ.

$\mathsf{E}'_\Sigma(m)$ for $m \in \{0,1\}^\lambda$:

1. Choose at random $m^1, m^2 \in \mathbb{F}_{2^\lambda}$ such that $m^1 + m^2 = m$.
2. For $b \in \{1,2\}$, Choose $(c_{L,b}, c_{R,b})$ at random, from $\mathbb{F}_{2^\lambda}^{\frac{\ell}{\lambda}}$, conditioned on $\langle c_{L,b}, c_{R,b} \rangle = m^b$.
3. $(com, d = d_1, \ldots, d_{4\ell}) \leftarrow \mathsf{Com}_\Sigma([c_{L,b} || c_{R,b}]_{b\in\{1,2\}})$
4. For $b \in \{1,2\}$, let $d[S_{L,b}]$ (resp. $d[S_{R,b}]$) correspond to the decommitment of com to the bits corresponding to $S_{L,b}$ (resp. $S_{R,b}$).
5. Output $L_1 = (com, d[S_{L,1}])$; $R_1 = (com, d[S_{R,1}])$; $L_2 = (com, d[S_{L,2}])$; $R_2 = (com, d[S_{R,2}])$.

$\mathsf{D}'_\Sigma(\widetilde{L}_1, \widetilde{R}_1, \widetilde{L}_2, \widetilde{R}_2)$: //For simplicity of notation, we assume D' can take its inputs in any order.

1. For $b \in \{1,2\}$, parse $\widetilde{L}_b = (\widetilde{com}_b, d[S_{L,b}])$, $\widetilde{R}_b = (\widetilde{com}'_b, d[S_{R,b}])$.
2. Check that $\widetilde{com}_1 = \widetilde{com}'_1 = \widetilde{com}_2 = \widetilde{com}'_2$.
3. For $b \in \{1,2\}$, let $\widetilde{c}_{L,b} = \mathsf{Open}_\Sigma(S_{L,b}, \widetilde{com}, \widetilde{d}[S_{L,b}])$ and $\widetilde{c}_{R,b} = \mathsf{Open}_\Sigma(S_{R,b}, \widetilde{com}, \widetilde{d}[S_{R,b}])$. Check that $\widetilde{c}_{L,b} \neq \perp$ and $\widetilde{c}_{R,b} \neq \perp$.
4. For $b \in \{1,2\}$, compute $\widetilde{m}_b = \langle c_{L,b}, c_{R,b} \rangle$.
5. If all the above checks pass, output $\widetilde{m}_1 + \widetilde{m}_2$; otherwise, output $\perp$.

Figure 7.6: Construction of 4-state Continuous, Non-Malleable Code.

To prove Theorem 7.5.1, we show that the construction above is a secure multi-bit CNMC, via the following sequence of hybrids.

Hybrid 0: *This is the Experiment from Definition 3.2.11.*

Hybrids 1 *and* 2 *are analogous to the first and second hybrids in Section 7.4. We therefore give an abbreviated description.*

Hybrid 1: *The experiment is identical to Hybrid* 0 *except we modify the decode algorithm from* D'_{Σ} *to* D'^{1}_{Σ} *to abort if the tampered codeword* $(\widetilde{L}_1, \widetilde{R}_1, \widetilde{L}_2, \widetilde{R}_2)$ *is different from the challenge codeword* (L_1, R_1, L_2, R_2) *but the corresponding commitment is not statistically binding.*

Hybrid 2: *The experiment is identical to Hybrid* 1*, except we switch to equivocal commitments in* (L_1, R_1) *(resp.* (L_2, R_2)*) that is given to the adversary. We denote the corresponding CRS's, and equivocal commitment and decommitments by* $\Sigma_{Eq}, \overline{com}, d^0 = d^0_1 \ldots d^0_{4\ell}, d^1 = d^1_1 \ldots d^1_{4\ell}$.

We now define some terminology which will be needed for the next sequence of hybrids. Given $(\Sigma_{Eq}, \overline{com}, d^0 = d^0_1 \ldots d^0_{4\ell}, d^1 = d^1_1 \ldots d^1_{4\ell})$ *an output* $(\mathbf{out}^{i-1}_{A,1}, \mathbf{out}^{i-1}_{A,2})$*, for* $b \in \{1,2\}$*, we define the sets* $\mathcal{S}_{L,b}$*,* $\mathcal{S}_{R,b}$*,* $\mathcal{S}'_{L,b}$*,* $\mathcal{S}'_{R,b}$ *as follows:*

- $\mathcal{S}_{L,b}$ *contains all values of* $\widehat{L}'_b$ *that occur with probability at least* $\epsilon = 1/2^{y_o/3}$*, where values of* $\widehat{L}'_b$ *are sampled as follows: Sample* $\widehat{c}_{L,b}$ *conditioned on* $\mathbf{out}^{i-1}_{A,b}$*. Compute equivocal decommitment of* $\overline{com}$*:* $\widehat{d}[S_{L,b}] := [d_i^{\widehat{c}_{L,b}[i]}]_{i \in S_{L,b}}$*. Apply* $f^i_{L,b}$ *to* $\widehat{L}_b = (\overline{com}_b, \widehat{d}[S_{L,b}])$ *to obtain* $\widehat{L}'_b$ *(or "same" if the output is* $\widehat{L}_b$ *itself).*

- *$\mathcal{S}_{R,b}$ contains all values of $\widehat{R}'_b$ that occur with probability at least $\epsilon = 1/2^{y_o/3}$, where values of $\widehat{R}'_b$ are sampled as follows: Sample $\widehat{c}_{R,b}$ conditioned on $\mathbf{out}^{i-1}_{A,b}$. Compute equivocal decommitment of $\overline{com}$: $\widehat{d}[S_{R,b}] := [d_i^{\widehat{c}_{R,b}[i]}]_{i \in S_{R,b}}$. Apply $f^i_{R,b}$ to $\widehat{R}_b = (\overline{com}_b, \widehat{d}[S_{R,b}])$ to obtain $\widehat{R}'_b$ (or "same" if the output is $\widehat{R}_b$ itself).*

- *Let $\mathcal{S}'_{L,b} \subseteq \mathcal{S}_{L,b}$ be the set of $\widehat{L}'_b$ such that there is a "matching" $\widehat{R}'_b \in \mathcal{S}_{R,b}$ such that $\mathsf{D'}^1_{\Sigma_{Eq}}(\widehat{L}'_b, \widehat{R}'_b, \cdot, \cdot) \neq \perp$.*

- *Let $\mathcal{S}'_{R,b} \subseteq \mathcal{S}_{R,b}$ be the set of $\widehat{R}'_b$ such that there is a "matching" $\widehat{L}'_b \in \mathcal{S}_{L,b}$ such that $\mathsf{D'}^1_{\Sigma_{Eq}}(\widehat{L}'_b, \widehat{R}'_b, \cdot, \cdot) \neq \perp$.*

We also give two alternate decoding procedures in Figures 7.7 and 7.8.

$\mathsf{D'}^3_{\Sigma_{Eq}}([(f^i_{L,b}, f^i_{R,b}), \widetilde{L}_b, \widetilde{R}_b]_{b \in \{1,2\}})$:

1. $\underline{\text{For } b \in \{1,2\}, \text{ check that } \widetilde{L}_b \in \mathcal{S}'_{L,b} \text{ and that } \widetilde{R}_b \in \mathcal{S}'_{R,b}. \text{ If not, output } \perp.}$
2. For $b \in \{1,2\}$, parse $\widetilde{L}_b = (\widetilde{com}_b, \widetilde{d}[S_{L,b}])$, $\widetilde{R}_b = (\widetilde{com}'_b, \widetilde{d}[S_{R,b}])$.
3. Check that $\widetilde{com}_1 = \widetilde{com}'_1 = \widetilde{com}_2 = \widetilde{com}'_2$.
4. For $b \in \{1,2\}$, let $\widetilde{c}_{L,b} = \mathsf{Open}_\Sigma(S_{L,b}, \widetilde{com}, \widetilde{d}[S_{L,b}])$ and $\widetilde{c}_{R,b} = \mathsf{Open}_\Sigma(S_{R,b}, \widetilde{com}, \widetilde{d}[S_{R,b}])$. Check that $\widetilde{c}_{L,b} \neq \perp$ and $\widetilde{c}_{R,b} \neq \perp$.
5. For $b \in \{1,2\}$, compute $\widetilde{m}_b = \langle \widetilde{c}_{L,b}, \widetilde{c}_{R,b} \rangle$.
6. If all the above checks pass, output $\widetilde{m}_1 + \widetilde{m}_2$; otherwise, output $\perp$.

Figure 7.7: Algorithm $\mathsf{D'}^3_{\Sigma_{Eq}}$.

For the following decode algorithm, we assume that at the beginning of the experiment, for $b \in \{1,2\}$, counters $\mathsf{count}_{L,b}, \mathsf{count}_{R,b}$ are initialized to 0. We also define L^*_b (resp. R^*_b) to be the element of $\mathcal{S}'_{L,b}$ (resp. $\mathcal{S}'_{R,b}$) that occurs most frequently.

We next present a sequence of intermediate hybrids $H^2 = H^{2,0,b}, H^{2,1,a} \ldots,$ $H^{2,q,b} = H^3$, defined as follows:

```
$\mathsf{D'}^4_{\Sigma_{Eq}}([(f^i_{L,b}, f^i_{R,b}), \widetilde{L}_b, \widetilde{R}_b]_{b \in \{1,2\}})$:
    1. Check that $\widetilde{L} \in \mathcal{S}'_L$ and that $\widetilde{R} \in \mathcal{S}'_R$. If not, output $\perp$.
    2. $\underline{\text{For } b \in \{1,2\}, \text{ if } \widetilde{L}_b \neq L^*_b, \text{ then set } \mathsf{count}_{L,b} := \mathsf{count}_{L,b} + 1.}$
    3. $\underline{\text{For } b \in \{1,2\}, \text{ if } \widetilde{R}_b \neq R^*_b, \text{ then set } \mathsf{count}_{R,b} := \mathsf{count}_{R,b} + 1.}$
    4. $\underline{\text{For } b \in \{1,2\}, \text{ if } \mathsf{count}_{L,b} > y_t \text{ or } \mathsf{count}_{R,b} > y_t, \text{ output } \perp.}$
    5. For $b \in \{1,2\}$, parse $\widetilde{L}_b = (\widetilde{com}_b, \widetilde{d}[S_{L,b}])$, $\widetilde{R}_b = (\widetilde{com}'_b, \widetilde{d}[S_{R,b}])$.
    6. Check that $\widetilde{com}_1 = \widetilde{com}'_1 = \widetilde{com}_2 = \widetilde{com}'_2$.
    7. For $b \in \{1,2\}$, let $\widetilde{c}_{L,b} = \mathsf{Open}_\Sigma(S_{L,b}, \widetilde{com}, \widetilde{d}[S_{L,b}])$ and $\widetilde{c}_{R,b} = \mathsf{Open}_\Sigma(S_{R,b}, \widetilde{com}, \widetilde{d}[S_{R,b}])$. Check that $\widetilde{c}_{L,b} \neq \perp$ and $\widetilde{c}_{R,b} \neq \perp$.
    8. For $b \in \{1,2\}$, compute $\widetilde{m}_b = \langle \widetilde{c}_{L,b}, \widetilde{c}_{R,b} \rangle$.
    9. If all the above checks pass, output $\widetilde{m}_1 + \widetilde{m}_2$; otherwise, output $\perp$.
```

Figure 7.8: Algorithm $\mathsf{D'}^4_{\Sigma_{Eq}}$.

Hybrid $H^{2,i,a}$ for $i \in [q]$: *The experiment is identical to the previous hybrid, except we respond to the i-th query to the decoding oracle using* $\mathsf{D'}^3_{\Sigma_{Eq}}$.

Hybrid $H^{2,i,b}$ for $i \in [q]$: *The experiment is identical to the previous hybrid, except we respond to the i-th query to the decoding oracle using* $\mathsf{D'}^4_{\Sigma_{Eq}}$.

Claim 7.5.1. Hybrids 0 *and* 1 *are computationally indistinguishable.*

This follows from the security of the one-time signature scheme and "uniqueness of opening" property of the underlying commitment.

Claim 7.5.2. Hybrids 1 *and* 2 *are computationally indistinguishable.*

This follows from the security of the equivocal, non-malleable commitment scheme.

We say that an output pair $(\mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1}, \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2})$ *is* good *if the marginal distribution over m^1 is statistically 2^{-k}-close to uniform random conditioned on $(\mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1}, \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2})$ and the marginal distribution over m^2 is statistically 2^{-k}-close to uniform random conditioned on $(\mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1}, \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2})$.*

Claim 7.5.3. For $i \in \{0, \ldots, q\}$, in Hybrid $H^{2,i,b}$, the outcome $(\mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1}, \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2})$ is good *with probability $1 - 2^{-k}/q$.*

Proof. Towards proving the claim, we consider the following leakage functions:

Leakage function on $c_{L,b}$, for $b \in \{1,2\}$: Fix $\Sigma_{Eq}, \overline{com}, d^0 = d^0_1 \ldots d^0_{4\ell}, d^1 = d^1_1 \ldots d^1_{4\ell}$, universal hash $h : \{0,1\}^\alpha \to \{0,1\}^{y_o} \in \mathcal{H}$ (where α is the length of a single split-state of the encoding) and adversary A. On input c_L, set output $\mathbf{out}_{A,b}$ to "" and $\mathbf{out}_{L,b}$ to "", and set $L_b = (\overline{com}, [d_i^{c_{L,b}[i]}]_{i\in[\ell]})$, for $b \in \{1,2\}$. Repeat the following in rounds $i = 1, 2, \ldots$:

1. Obtain the next tampering function $[(f_{L,b}, f_{R,b})]_{b\in\{1,2\}}$ from adversary A. If A terminates then output $\mathbf{out}_{L,b}$.
2. Set $L'_b := f_{L,b}(L_b)$. If $L'_b \in \mathcal{S}'_{L,b}$, then:
   (a) Find the unique $\widehat{R}'_b \in \mathcal{S}'_{R,b}$ such that $\mathsf{D'}^1_{\Sigma_{Eq}}(L'_b, \widehat{R}'_b, \cdot, \cdot) \neq \perp$. Return $(L'_b, \widehat{R}'_b)$ to the adversary. Set $\mathbf{out}_{A,b} = \mathbf{out}_{A,b}||(L'_b, \widehat{R}'_b)$.
   (b) If L'_b is not the most frequent output in $\mathcal{S}'_{L,b}$, set $\mathbf{out}_{L,b} := \mathbf{out}_{L,b}||(i||h(L'_b))$ If $|\mathbf{out}_{L,b}| > (\log(q) + y_o) \cdot y_t$ then terminate with output $\mathbf{out}_{L,b} := \mathbf{out}_{L,b}||(i||\perp)$.

3. If $L'_b \notin \mathcal{S}'_{L,b}$, output $\perp$ to the adversary and terminate with output $\mathbf{out}_{L,b} := \mathbf{out}_{L,b}||(i||\perp)$.

The leakage function for $c_{R,b}$ is analogous.

We now show that given $\mathbf{out}_{L,1}$, $\mathbf{out}_{R,1}$, $\mathbf{out}_{L,2}$, $\mathbf{out}_{R,2}$ we can reconstruct the full output sequence for the adversary's view with probability $1 - \frac{4q}{\epsilon^2 \cdot 2^{y_O}} = 1 - \frac{4q}{2^{y_O/3}}$ in the following way:

Fix $\Sigma_{Eq}, \overline{com}$, universal hash $h \leftarrow \mathcal{H}$ and adversary A. Set output $\mathbf{out}_{A,1}$ to "" and $\mathbf{out}_{A,2}$ to "". Repeat the following in rounds $i = 1, 2, \ldots, q$:

1. Obtain the next tampering function $(f_{L,1}, f_{R,1}, f_{L,2}, f_{R,2})$ from adversary A given its current view, $\mathbf{out}_A = (\mathbf{out}_{A,1}, \mathbf{out}_{A,2})$.

2. If for $b \in \{1,2\}$, $(i, \perp) \in \mathbf{out}_{L,b}$ or $(i, \perp) \in \mathbf{out}_{R,b}$, then for $b \in \{1,2\}$, set $\mathbf{out}_{A,b} = \mathbf{out}_{A,b}||(i, \perp)$ and abort.

3. If for $b \in \{1,2\}$, $(i, y) \in \mathbf{out}_{L,b}$, for some $y \neq \perp$, set $L'_b = \widehat{L}'_b$ such that $\widehat{L}'_b \in \mathcal{S}'_{L,b}$ and $h(\widehat{L}'_b) = y$.

4. If for $b \in \{1,2\}$, $(i, \cdot) \notin \mathbf{out}_{L,b}$, set $L'_b = \widehat{L}'_b$ such that $\widehat{L}'_b \in \mathcal{S}'_{L,b}$ is the most frequent value.

5. If for $b \in \{1,2\}$, $(i, y) \in \mathbf{out}_{R,b}$, for some $y \neq \perp$, set $R'_b = \widehat{R}'_b$ such that $\widehat{R}'_b \in \mathcal{S}'_{R,b}$ and $h(\widehat{R}'_b) = y$.

6. If for $b \in \{1,2\}$, $(i, \cdot) \notin \mathbf{out}_{R,b}$, set $R'_b = \widehat{R}'_b$ such that $\widehat{R}'_b \in \mathcal{S}'_{R,b}$ is the most frequent value.

7. If for all $b \in \{1, 2\}$, $L'_b =$ "same" and $R'_b =$ "same" output "same" and for $b \in \{1, 2\}$, set $\mathbf{out}_{A,b} = \mathbf{out}_{A,b} || \text{"same"}$.

8. Else if at least one of $[L'_b, R'_b]_{b \in \{1,2\}}$ is "same" but not all, then for $b \in \{1, 2\}$, set $\mathbf{out}_{A,b} = \mathbf{out}_{A,b} || \perp$ and abort.

9. Else for $b \in \{1, 2\}$, parse $L'_b := (com_b, d[S_{L,b}])$ and $R'_b := (com'_b, d[S_{R_b}])$. Check that $com_1 = com'_1 = com_2 = com'_2$. If not, set $\mathbf{out}_{A,b} = \mathbf{out}_{A,b} || \perp$ and abort.

10. Otherwise, for $b \in \{1, 2\}$ set $\mathbf{out}_{A,b} = \mathbf{out}_{A,b} || (L'_b, R'_b)$.

It can be determined by inspection that the incorrect value is output only if in one of the at most q instances, for some $b \in \{1, 2\}$, there are two distinct values $\widehat{L}'_b, \widehat{L}''_b \in \mathcal{S}'_{L,b}$ or $\widehat{R}'_b, \widehat{R}''_b \in \mathcal{S}'_{R_b}$ such that $h(\widehat{L}'_b) = h(\widehat{L}''_b)$ or $h(\widehat{R}'_b) = h(\widehat{R}''_b)$. Due to universality of h and the fact that for $b \in \{1, 2\}$, $|\mathcal{S}'_{L,b}| = |\mathcal{S}'_{R,b}| \leq 1/\epsilon$, this can occur with probability at most $\frac{4q}{\epsilon^2 \cdot 2^{y_o}}$, as claimed.

Since for $b \in \{1, 2\}$, $|\mathbf{out}_{L,b}| \leq (\log(q) + y_o) \cdot y_t \leq 2y_o \cdot y_t \leq c \cdot \ell$, and $|\mathbf{out}_{R,b}| \leq (\log(q) + y_o) \cdot y_t \leq 2y_o \cdot y_t \leq c \cdot \ell$, for constant $c < 1$, we have that with probability $1 - 2^{-k}/q$ over choice of $(\mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1}, \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2})$, the min-entropy of $c_{L,b}$ (resp. $c_{R,b}$) conditioned on $(\mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1}, \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2})$ is at least $c' \cdot \ell$ for constant $c' < 1$. We can use the properties of the inner product extractor given in Lemma 7.2.1 to argue that $\langle c_{L,b}, c_{R,b} \rangle$ is statistically close to uniform random, given $\mathbf{out}_{L,1}$, $\mathbf{out}_{R,1}$, $\mathbf{out}_{L,2}$, $\mathbf{out}_{R,2}$. Moreover, since we have shown that the view of the adversary $\mathbf{out}^i_{A,1}, \mathbf{out}^i_{A,2}$ can be fully reconstructed given $\mathbf{out}_L, \mathbf{out}_R$, we have that $\langle c_{L,b}, c_{R,b} \rangle$ is statistically close to uniform, given the adversary's view in the CNMC experiment. □

Claim 7.5.4. For $i \in \{0, \dots, q-1\}$, Hybrids $H^{2,i,b}$ and $H^{2,i+1,a}$ are $4(\epsilon' + 2^{-k})$-close, where $\epsilon' = (1 + 2^{-\lambda})\epsilon$.

Proof. To prove indistinguishability of Hybrids 2 and 3, it is sufficient to show that for and $b \in \{1, 2\}$, $\Pr[f^i_{L,b}(L_b) \notin \mathcal{S}'_{L,b} \wedge \mathsf{D'}^1_{\Sigma_{Eq}}(f^i_{L,b}(L_b), f^i_{R,b}(R_b), \cdot, \cdot) \neq \bot] \leq \epsilon'$ and $\Pr[f^i_{R,b}(R_b) \notin \mathcal{S}'_{R,b} \wedge \mathsf{D'}^1_{\Sigma_{Eq}}(f^i_{L,b}(L), f^i_{R,b}(R_b), \cdot, \cdot) \neq \bot] \leq \epsilon'$. The result then follows by a union bound.

Given Claim 7.5.3, to bound the above it is sufficient to show: (1) for $b \in \{1, 2\}$, each *good* pair $\mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1}$ (which does not contain $\bot$), $\mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2}$ (which does not contain $\bot$), and each value of $R_b = \widehat{R}_b$,

$$\Pr\left[\begin{array}{c|c} f^i_{L,b}(L_b) \notin \mathcal{S}'_{L,b} \wedge & R_b = \widehat{R}_b \wedge \mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1} \wedge \\ \mathsf{D'}^1_{\Sigma_{Eq}}(f^i_{L,b}(L_b), f^i_{R,b}(R_b), \cdot, \cdot) \neq \bot & \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2} \end{array}\right] \leq \epsilon';$$

and (2) for $b \in \{1, 2\}$, each *good* pair $\mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1}$ (which does not contain $\bot$), $\mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2}$ (which does not contain $\bot$), and each value of $L_b = \widehat{L}_b$,

$$\Pr\left[\begin{array}{c|c} f^i_{R,b}(R_b) \notin \mathcal{S}'_{R,b} \wedge & L_b = \widehat{L}_b \wedge \mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1} \\ \mathsf{D'}^1_{\Sigma_{Eq}}(f^i_{L,b}(L_b), f^i_{R,b}(R_b), \cdot, \cdot) \neq \bot & \wedge \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2} \end{array}\right] \leq \epsilon'.$$

We first fix $(\Sigma_{Eq}, \overline{com}, d^0 = d^0_1 \dots d^0_{4\ell}, d^1 = d^1_1 \dots d^1_{4\ell})$.

Note that fixed $\Sigma_{Eq}, \overline{com}, d^0 = d^0_1 \dots d^0_{4\ell}, d^1 = d^1_1 \dots d^1_{4\ell}$ and $b \in \{1, 2\}$, there is a bijection $\phi_{L,b}$ (resp. $\phi_{R,b}$) between $c_{L,b}$ (resp. $c_{R,b}$) and $(\overline{com}, d[S_{L,b}])$, where $d[S_{L,b}] = [d_i^{c_{L,b}[i]}]_{i \in [\ell]}$. Therefore the probability of a particular value of $c_{L,b}$ (resp. $c_{R,b}$) occurring is equivalent to the probability of $L_b = \phi_{L,b}(c_{L,b})$ (resp. $R_b = \phi_{R,b}(c_{R,b})$).

Additionally, Let $\rho_{L,b}$ (resp. $\rho_{R,b}$) be the function that given $f^i_{R,b}(R_b)$ (resp. $f^i_{L,b}(L_b)$) returns the unique L'_b (resp. R'_b) if it exists such that, $\mathsf{D'}^1_{\Sigma_{Eq}}(L'_b, f^i_{R,b}(R_b), \cdot, \cdot) \neq \perp$ (resp. $\mathsf{D'}^1_{\Sigma_{Eq}}(f^i_{L,b}(L_b), R'_b, \cdot, \cdot) \neq \perp$). Note that L'_b (resp. R'_b) is equal to "same" if and only if $f^i_{R,b}(R_b) =$ "same" (resp. $f^i_{L,b}(L_b) =$ "same").

Now, note that for $b = 1$ and every $(\widehat{c}_{L,1}, \widehat{c}_{R,1})$ and every *good* pair $\widehat{Out}_{A,1}, \widehat{Out}_{A,2}$):

$$\begin{aligned}
&\Pr[c_{L,1} = \widehat{c}_{L,1} \mid c_{R,1} = \widehat{c}_{R,1} \wedge \mathbf{out}^{i-1}_{A,1} = \widehat{Out}_{A,1}, \mathbf{out}^{i-1}_{A,2} = \widehat{Out}_{A,2}] \\
&= \sum_{\widehat{c}_{L,2}, \widehat{c}_{R,2}} \Big(\Pr[c_{L,2} = \widehat{c}_{L,2}, c_{R,2} = \widehat{c}_{R,2} \mid \mathbf{out}^{i-1}_{A,2} = \widehat{Out}_{A,2}] \\
&\qquad \cdot \Pr[c_{L,1} = \widehat{c}_{L,1}, c_{R,1} = \widehat{c}_{R,1} \mid \mathbf{out}^{i-1}_{A,1} = \widehat{Out}_{A,1} \wedge \langle c_{L,2}, c_{R,2}\rangle = \langle \widehat{c}_{L,2}, \widehat{c}_{R,2}\rangle] \Big) \\
&= \sum_{m^2} \Pr[\langle c_{L,2}, c_{R,2}\rangle = m^2 \mid \mathbf{out}^{i-1}_{A,2} = \widehat{Out}_{A,2}] \\
&\qquad \cdot \Pr[c_{L,1} = \widehat{c}_{L,1}, c_{R,1} = \widehat{c}_{R,1} \mid \mathbf{out}^{i-1}_{A,1} = \widehat{Out}_{A,1} \wedge m^2] \\
&\in \sum_{\widehat{m}^2} (2^{-\lambda} \pm 2^{-k}) \cdot \Pr[c_{L,1} = \widehat{c}_{L,1} \mid c_{R,1} = \widehat{c}_{R,1} \wedge \mathbf{out}^{i-1}_{A,1} = \widehat{Out}_{A,1} \wedge m^2 = \widehat{m}^2] \qquad (7.5.1) \\
&= (2^{-\lambda} \pm 2^{-k}) \sum_{m^2} \Pr[c_{L,1} = \widehat{c}_{L,1} \mid c_{R,1} = \widehat{c}_{R,1} \wedge \mathbf{out}^{i-1}_{A,1} = \widehat{Out}_{A,1} \wedge m^2 = \widehat{m}^2] \\
&= (1 \pm 2^{\lambda-k}) \cdot \Pr[c_{L,1} = \widehat{c}_{L,1} \mid \wedge \mathbf{out}^{i-1}_{A,1} = \widehat{Out}_{A,1}] \\
&= (1 \pm 2^{-\lambda}) \cdot \Pr[c_{L,1} = \widehat{c}_{L,1} \mid c_{R,1} \wedge \mathbf{out}^{i-1}_{A,1} = \widehat{Out}_{A,1}],
\end{aligned}$$

where (7.5.1) follows from Claim 7.5.3. An analogous statement holds for $b = 2$.

Morever, by the same reasoning as in the proof of Claim 7.4.3 (where we showed conditional independence of $c_{L,b}, c_{R,b}$) we have that for every $(\widehat{c}_{L,b}, \widehat{c}_{R,b}, \widehat{Out}_{A,b})$:

$$\Pr[c_{L,b} = \widehat{c}_{L,b} \mid c_{R,b} \wedge \mathbf{out}^{i-1}_{A,b} = \widehat{Out}_{A,b}] = \Pr[c_{L,b} = \widehat{c}_{L,b} \mid \mathbf{out}^{i-1}_{A,b} = \widehat{Out}_{A,b}].$$

Therefore, for every every $(\widehat{c}_{L,1}, \widehat{c}_{R,1})$ and every *good* pair $\widehat{Out}_{A,1}, \widehat{Out}_{A,2}$):

$$\Pr\left[L_b = \widehat{L}_b \;\middle|\; \begin{array}{c} R_b = \widehat{R}_b \wedge \mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1} \\ \wedge \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2} \end{array} \right] \in (1 \pm 2^{-\lambda}) \Pr[L_b = \widehat{L}_b \mid \mathbf{out}^{i-1}_{A,b} = \widehat{Out}^{i-1}_{A,b}].$$

So for every $R_b = \widehat{R}_b$, every *good* pair $\widehat{Out}_{A,1}, \widehat{Out}_{A,2}$) and every $L'_b \notin \mathcal{S}'_{L,b}$:

$$\Pr\left[f^i_{L,b}(L_b) = L'_b \;\middle|\; \begin{array}{c} R_b = \widehat{R}_b \wedge \mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1} \wedge \\ \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2} \end{array} \right] \leq (1 + 2^{-\lambda})\epsilon \leq \epsilon'.$$

Therefore,

$$\begin{aligned} &\Pr\left[\begin{array}{c} f^i_{L,b}(L_b) \notin \mathcal{S}'_{L,b} \wedge \\ \mathsf{D'}^1_{\Sigma_{Eq}}(f^i_{L,b}(L_b), f^i_{R,b}(R_b), \cdot, \cdot) \neq \perp \end{array} \;\middle|\; \begin{array}{c} R_b = \widehat{R}_b \wedge \mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1} \wedge \\ \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2} \end{array} \right] \\ &\quad = \Pr\left[\begin{array}{c} f^i_{L,b}(L_b) \notin \mathcal{S}'_{L,b} \wedge \\ f^i_{L,b}(L_b) = \rho_L(f^i_{R,b}(R_b)) \end{array} \;\middle|\; \begin{array}{c} R_b = \widehat{R}_b \wedge \mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1} \wedge \\ \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2} \end{array} \right] \\ &\quad \leq \epsilon'. \end{aligned}$$

The proof for (2) is analogous. □

Claim 7.5.5. For $i \in \{1, \ldots, q\}$, Hybrids $H^{2,i,a}$ and $H^{2,i,b}$ are $4(\epsilon' + 2^{-k})$-close, where

$\epsilon' = 2 \cdot \epsilon$.

Proof. To prove indistinguishability, we must show that for $i \in \{1, \dots, q\}$, $b \in \{1, 2\}$ the probability that the event (1) $f^i_{L,b}(L_b)$ is not most frequent and $\mathsf{D'}^1_{\Sigma_{Eq}}(f^i_{L,b}(L_b), f^i_{R,b}(R_b), \cdot, \cdot) \neq \perp$ occurs more than y_t times in $H^{2,i,a}$ is at most $((1 + 2^{-\lambda})/2)^{y_t} + 2^{-k}$ and the probability that the event (2) $f^i_{R,b}(R_b)$ is not most frequent and $\mathsf{D'}^1_{\Sigma_{Eq}}(f^i_{L,b}(L), f^i_{R,b}(R_b), \cdot, \cdot) \neq \perp$ occurs more than y_t times in $H^{2,i,a}$ is at most $((1 + 2^{-\lambda})/2)^{y_t} + 2^{-k}$.

We first analyze the event (1). If $f^i_{L,b}(L_b) = L'_b$ is not the most frequent query in $\mathcal{S}'_{L,b}$ then, by definition,

$$\Pr[f^i_{L,b}(\widehat{L}_b) = L'_b \mid \mathbf{out}^{i-1}_A = \widehat{Out}^{i-1}_A] \leq 1/2. \tag{7.5.2}$$

Recall that, by the arguments in the proof of the previous Claim, in $H^{2,i-1,b}$ (and hence also $H^{2,i,a}$), for *good* pairs $(\mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1} \wedge \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2})$:

$$\Pr\left[f^i_{L,b}(L_b) = L'_b \,\middle|\, \begin{array}{c} R_b = \widehat{R}_b \wedge \mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1} \wedge \\ \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2} \end{array} \right]$$
$$\in (1 \pm 2^{-\lambda}) \Pr[f^i_{L,b}(L_b) = L'_b \mid \mathbf{out}^{i-1}_{A,b} = \widehat{Out}^{i-1}_{A,b}].$$

Combining with (7.5.2):

$$\Pr\left[f^i_{L,b}(L_b) = L'_b \,\middle|\, \begin{array}{c} R_b = \widehat{R}_b \wedge \mathbf{out}^{i-1}_{A,1} = \widehat{Out}^{i-1}_{A,1} \wedge \\ \mathbf{out}^{i-1}_{A,2} = \widehat{Out}^{i-1}_{A,2} \end{array} \right] \leq (1 + 2^{-\lambda})/2.$$

This implies that for every fixed $R_b = \widehat{R}_b$ (for which $\Pr[R_b = \widehat{R}_b \wedge \mathbf{out}_{A,1}^{i-1} = \widehat{Out}_{A,1}^{i-1} \wedge \mathbf{out}_{A,2}^{i-1} = \widehat{Out}_{A,2}^{i-1}] > 0$),

$$\begin{aligned}
&\Pr\left[\begin{array}{c} f_{L,b}^{i}(L_b) \neq L_b^{*} \wedge \\ \mathsf{D}'(f_{L,b}^{i}(L_b), f_{R,b}^{i}(R_b)) \neq \perp \end{array} \middle| \begin{array}{c} R_b = \widehat{R}_b \wedge \mathbf{out}_{A,1}^{i-1} = \widehat{Out}_{A,1}^{i-1} \wedge \\ \mathbf{out}_{A,2}^{i-1} = \widehat{Out}_{A,2}^{i-1} \end{array}\right] \\
&\quad = \Pr\left[\begin{array}{c} f_{L,b}^{i}(L_b) \neq L_b^{*} \wedge \\ f_{L,b}^{i}(L) = \rho_L(f_{R,b}^{i}(R_b)) \end{array} \middle| \begin{array}{c} R_b = \widehat{R}_b \wedge \mathbf{out}_{A,1}^{i-1} = \widehat{Out}_{A,1}^{i-1} \wedge \\ \mathbf{out}_{A,2}^{i-1} = \widehat{Out}_{A,2}^{i-1} \end{array}\right] \\
&\quad \leq (1 + 2^{-\lambda})/2.
\end{aligned}$$

The probability that this occurs y_t times for y_t distinct values of $j \leq [i]$, where all outcomes of $\mathbf{out}_{A,1}^{j} = \widehat{Out}_{A,1}^{j} \wedge \mathbf{out}_{A,2}^{j} = \widehat{Out}_{A,2}^{j}$ are *good* for all j is at most $((1+2^{-\lambda})/2)^{y_t}$. Since by Claim 7.5.3, with probability $1-2^{-k}$, all outcomes $\mathbf{out}_{A,1}^{j} = \widehat{Out}_{A,1}^{j} \wedge \mathbf{out}_{A,2}^{j} = \widehat{Out}_{A,2}^{j}$ are *good* for all $j \in [q]$, the upperbound for event (1) follows.

The proof for event (2) is analogous. □

Claim 7.5.6. In Hybrid 3, for all (even inefficient) distinguishers D, it holds that

$$\Pr[D(\mathsf{out}_{\mathcal{A}}^{b}) = b] \leq 1/2 + O(2^{-\lambda}).$$

Proof. We first compute

$$\begin{aligned}
O = &\sum_{m'^2} \Pr[m^2 = m'^2 \wedge m^1 = m_0 + m'^2 \mid \mathbf{out}_{A,1}, \mathbf{out}_{A,2}] \\
&+ \sum_{m'^2} \Pr[m^2 = m'^2 \wedge m^1 = m_1 + m'^2 \mid \mathbf{out}_{A,1}, \mathbf{out}_{A,2}] \\
= &\sum_{m'^2} \Pr[m^2 = m'^2 \mid \mathbf{out}_{A,1}, \mathbf{out}_{A,2}] \cdot \Pr[m^1 = m_0 + m'^2 \mid \mathbf{out}_{A,1}, \mathbf{out}_{A,2}] \\
&+ \sum_{m'^2} \Pr[m^2 = m'^2 \mid \mathbf{out}_{A,1}, \mathbf{out}_{A,2}] \cdot \Pr[m^1 = m_1 + m'^2 \mid \mathbf{out}_{A,1}, \mathbf{out}_{A,2}] \\
\geq &(2^{-\lambda} - 2^{-k}) \cdot (\sum_{m'^2} \Pr[m^1 = m_0 + m'^2 \mid \mathbf{out}_{A,1}, \mathbf{out}_{A,2}] \\
&\qquad + \sum_{m'^2} \Pr[m^1 = m_1 + m'^2 \mid \mathbf{out}_{A,1}, \mathbf{out}_{A,2}]) \\
= &2 \cdot (2^{-\lambda} - 2^{-k}) \\
= &2 \cdot 2^{-\lambda} - 2 \cdot 2^{-k}.
\end{aligned}$$

where the first inequality follows from Claim 7.5.3.

So

$$\begin{aligned}
&\Pr[\text{ message is } m_b \mid \mathbf{out}_{A,1}, \mathbf{out}_{A,2}] \\
&\quad = \frac{\sum_{m'_2} \Pr[m^2 = m'^2 \mid \mathbf{out}_{A,1}, \mathbf{out}_{A,2}] \cdot \Pr[m^1 = m_b + m^2 \mid \mathbf{out}_{A,1}, \mathbf{out}_{A,2}]}{O} \\
&\quad \le (2^{-\lambda} + 2^{-k}) \cdot \frac{\sum_{m'_2} \Pr[m^1 = m_b + m^2 \mid \mathbf{out}_{A,1}, \mathbf{out}_{A,2}]}{O} \\
&\quad = \frac{2^{-\lambda} + 2^{-k}}{O} \\
&\quad = \frac{2^{-\lambda} + 2^{-k}}{2 \cdot 2^{-\lambda} - 2 \cdot 2^{-k}} \\
&\quad \le \frac{2^{-\lambda} + 3 \cdot 2^{-k}}{2 \cdot 2^{-\lambda}} \\
&\quad = 1/2 + \frac{3 \cdot 2^{-k}}{2 \cdot 2^{-\lambda}} \\
&\quad = 1/2 + O(2^{-\lambda}).
\end{aligned}$$

$\square$

Chapter 8: Conclusion and Future Directions

In this dissertation, we have shown following contributions to improve the applicability of non-malleable codes.

- *We explore a novel connection between tampering classes of non-malleable codes and well studied complexity classes. Exploring these connections to complexity theory has made it possible to construct explicit, efficient non-malleable codes for broader tampering classes. Specifically,*
  - *We construct first explicit, efficient non-malleable codes with information theoretic security for the tampering functions which can be modeled as bounded-depth, bounded fan-in circuits. This tampering class includes* NC^0.
  - *We present a general framework to construct non-malleable codes from average-case hardness properties for various complexity classes which are known/believed to be strictly weaker from the class of all polynomial time algorithms (*P*).*
  - *We construct first explicit, efficient non-malleable codes for* AC^0*, decision tress of depth* n^{ε} *for constant* $0 < \varepsilon < 1$ *in with computational security*

in CRS model. We also construct information theoretically secure, non-malleable codes for streaming, space-bounded tampering functions.

This line of research has sparked interest in the community and led to follow-up work focusing on construction of non-malleable codes for well studied complexity classes.

- *We present lower and upper bounds on the optimal locality (efficiency), which is number of codeword blocks required to be accessed in order to decode/update the codeword, of locally decodable and updatable non-malleable codes which allow random access (i.e. entire codeword need not be read to access just a single block) to the codeword blocks. Specifically, we show that LDUNMC allowing rewind attacks cannot have constant locality. We also improve the locality of previous construction to match the lower bound.*

- *We show that continuous non-malleable codes, providing security against stronger attacks which tamper the codeword continually, cannot be constructed from any falsifiable assumption without CRS for 2-split-state tampering with black-box reduction security proofs (such reductions have only input/output access to the adversary breaking the underlying assumption). Prior work already showed 2-split-state CNMC cannot be constructed with information theoretic security. We then construct CNMC for 2-split-state tampering from injective one-way functions in CRS model. We also construct CNMC for multi-bit messages for 4-split-state tampering functions from injective one-way functions in CRS model.*

8.1 Future Directions

An important future direction would be to find practical applications of non-malleable codes to achieve tamper resilience. The Rowhammer attack (introduced by [112]) can be a potential threat to be considered. Since this attack tampers with the memory but does not interfere with the computation, which is precisely the threat model considered by non-malleable codes.

Recently, Cojocar et.al [47], studied the effectiveness of error-correcting codes against the rowhammer attack (introduced by [112]). They [47], tried to inject undetectable, silent corruptions in ECC memory by combining single bit flips caused by rowhammer attack and then caused 2 *more bits to flip. It is an interesting research direction to explore whether the type of tampering induced by Rowhammer is captured by local tampering functions. Therefore, studying the effectiveness of non-malleable codes against such types of attacks can be an interesting step forward in improving the applicability of non-malleable codes.*

Bibliography

[1] *Abe, M., Groth, J., Ohkubo, M.: Separating short structure-preserving signatures from non-interactive assumptions. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. Lecture Notes in Computer Science, vol. 7073, pp. 628–646. Springer, Heidelberg, Germany, Seoul, South Korea (Dec 4–8, 2011)*

[2] *Aggarwal, D., Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Optimal computational split-state non-malleable codes. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A: 13th Theory of Cryptography Conference, Part II. Lecture Notes in Computer Science, vol. 9563, pp. 393–417. Springer, Heidelberg, Germany, Tel Aviv, Israel (Jan 10–13, 2016)*

[3] *Aggarwal, D., Dodis, Y., Kazana, T., Obremski, M.: Non-malleable reductions and applications. In: Servedio, R.A., Rubinfeld, R. (eds.) 47th Annual ACM Symposium on Theory of Computing. pp. 459–468. ACM Press, Portland, OR, USA (Jun 14–17, 2015)*

[4] *Aggarwal, D., Dodis, Y., Lovett, S.: Non-malleable codes from additive combinatorics. In: Shmoys, D.B. (ed.) 46th Annual ACM Symposium on Theory of Computing. pp. 774–783. ACM Press, New York, NY, USA (May 31 – Jun 3, 2014)*

[5] *Aggarwal, D., Döttling, N., Nielsen, J.B., Obremski, M., Purwanto, E.: Continuous non-malleable codes in the 8-split-state model. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019, Part I. Lecture Notes in Computer Science, vol. 11476, pp. 531–561. Springer, Heidelberg, Germany, Darmstadt, Germany (May 19–23, 2019)*

[6] *Aggarwal, D., Dziembowski, S., Kazana, T., Obremski, M.: Leakage-resilient non-malleable codes. Cryptology ePrint Archive, Report 2014/807 (2014), http://eprint.iacr.org/2014/807*

[7] Aggarwal, D., Dziembowski, S., Kazana, T., Obremski, M.: Leakage-resilient non-malleable codes. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015: 12th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 9014, pp. 398–426. Springer, Heidelberg, Germany, Warsaw, Poland (Mar 23–25, 2015)

[8] Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Explicit non-malleable codes against bit-wise tampering and permutations. In: Gennaro, R., Robshaw, M.J.B. (eds.) Advances in Cryptology – CRYPTO 2015, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 538–557. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2015)

[9] Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: Explicit non-malleable codes against bit-wise tampering and permutations. In: Gennaro, R., Robshaw, M. (eds.) Advances in Cryptology - CRYPTO 2015 - 35th Annual Cryptology Conference, Santa Barbara, CA, USA, August 16-20, 2015, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9215, pp. 538–557. Springer (2015), http://dx.doi.org/10.1007/978-3-662-47989-6_26

[10] Agrawal, S., Gupta, D., Maji, H.K., Pandey, O., Prabhakaran, M.: A rate-optimizing compiler for non-malleable codes against bit-wise tampering and permutations. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015: 12th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 9014, pp. 375–397. Springer, Heidelberg, Germany, Warsaw, Poland (Mar 23–25, 2015)

[11] Ajtai, M.: σ_1^1-formulae on finite structures. Annals of Pure and Applied Logic 24, 607–620 (1983)

[12] Applebaum, B.: Cryptography in Constant Parallel Time. Information Security and Cryptography, Springer (2014), http://dx.doi.org/10.1007/978-3-642-17367-7

[13] Applebaum, B., Barak, B., Wigderson, A.: Public-key cryptography from different assumptions. In: Schulman, L.J. (ed.) 42nd Annual ACM Symposium on Theory of Computing. pp. 171–180. ACM Press, Cambridge, MA, USA (Jun 5–8, 2010)

[14] Arora, S., Barak, B.: Computational complexity: a modern approach. Cambridge University Press (2009)

[15] Ball, M., Dachman-Soled, D., Guo, S., Malkin, T., Tan, L.Y.: Non-malleable codes for small-depth circuits. In: Thorup, M. (ed.) 59th Annual Symposium on Foundations of Computer Science. pp. 826–837. IEEE Computer Society Press, Paris, France (Oct 7–9, 2018)

[16] *Ball, M., Dachman-Soled, D., Kulkarni, M., Lin, H., Malkin, T.: Non-malleable codes against bounded polynomial time tampering. In: Ishai, Y., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2019, Part I. Lecture Notes in Computer Science, vol. 11476, pp. 501–530. Springer, Heidelberg, Germany, Darmstadt, Germany (May 19–23, 2019)*

[17] *Ball, M., Dachman-Soled, D., Kulkarni, M., Malkin, T.: Non-malleable codes for bounded depth, bounded fan-in circuits. In: Fischlin, M., Coron, J.S. (eds.) Advances in Cryptology – EUROCRYPT 2016, Part II. Lecture Notes in Computer Science, vol. 9666, pp. 881–908. Springer, Heidelberg, Germany, Vienna, Austria (May 8–12, 2016)*

[18] *Ball, M., Dachman-Soled, D., Kulkarni, M., Malkin, T.: Non-malleable codes from average-case hardness:* AC^0*, decision trees, and streaming space-bounded tampering. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018, Part III. Lecture Notes in Computer Science, vol. 10822, pp. 618–650. Springer, Heidelberg, Germany, Tel Aviv, Israel (Apr 29 – May 3, 2018)*

[19] *Ball, M., Guo, S., Wichs, D.: Non-malleable codes for decision trees. IACR Cryptology ePrint Archive 2019, 379 (2019)*

[20] *Ball, M., Rosen, A., Sabin, M., Vasudevan, P.N.: Average-case fine-grained hardness. In: Hatami, H., McKenzie, P., King, V. (eds.) Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017. pp. 483–496. ACM (2017), http://doi.acm.org/10.1145/3055399.3055466*

[21] *Barak, B.: How to go beyond the black-box simulation barrier. In: 42nd Annual Symposium on Foundations of Computer Science. pp. 106–115. IEEE Computer Society Press, Las Vegas, NV, USA (Oct 14–17, 2001)*

[22] *Barak, B., Mahmoody-Ghidary, M.: Merkle puzzles are optimal - an* $O(n^2)$*-query attack on any key exchange from a random oracle. In: Halevi, S. (ed.) Advances in Cryptology – CRYPTO 2009. Lecture Notes in Computer Science, vol. 5677, pp. 374–390. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 16–20, 2009)*

[23] *Ben-Or, M., Goldwasser, S., Wigderson, A.: Completeness theorems for non-cryptographic fault-tolerant distributed computation (extended abstract). In: 20th Annual ACM Symposium on Theory of Computing. pp. 1–10. ACM Press, Chicago, IL, USA (May 2–4, 1988)*

[24] *Biham, E., Shamir, A.: Differential fault analysis of secret key cryptosystems. In: Kaliski Jr., B.S. (ed.) Advances in Cryptology – CRYPTO'97. Lecture Notes in Computer Science, vol. 1294, pp. 513–525. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 1997)*

[25] Bitansky, N., Paneth, O.: From the impossibility of obfuscation to a new non-black-box simulation technique. In: 53rd Annual Symposium on Foundations of Computer Science. pp. 223–232. IEEE Computer Society Press, New Brunswick, NJ, USA (Oct 20–23, 2012)

[26] Bogdanov, A., Lee, C.H.: Homomorphic evaluation requires depth. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A: 13th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 9562, pp. 365–371. Springer, Heidelberg, Germany, Tel Aviv, Israel (Jan 10–13, 2016)

[27] Boneh, D., DeMillo, R.A., Lipton, R.J.: On the importance of eliminating errors in cryptographic computations. Journal of Cryptology 14(2), 101–119 (Mar 2001)

[28] Brumley, D., Boneh, D.: Remote timing attacks are practical. Computer Networks 48(5), 701–716 (2005)

[29] Chabanne, H., Cohen, G.D., Patey, A.: Secure network coding and non-malleable codes: Protection against linear tampering. In: Proceedings of the 2012 IEEE International Symposium on Information Theory, ISIT 2012, Cambridge, MA, USA, July 1-6, 2012. pp. 2546–2550. IEEE (2012), http://dx.doi.org/10.1109/ISIT.2012.6283976

[30] Chandran, N., Goyal, V., Mukherjee, P., Pandey, O., Upadhyay, J.: Block-wise non-malleable codes. Cryptology ePrint Archive, Report 2015/129 (2015), http://eprint.iacr.org/2015/129

[31] Chandran, N., Goyal, V., Mukherjee, P., Pandey, O., Upadhyay, J.: Block-wise non-malleable codes. In: Chatzigiannakis, I., Mitzenmacher, M., Rabani, Y., Sangiorgi, D. (eds.) ICALP 2016: 43rd International Colloquium on Automata, Languages and Programming. LIPIcs, vol. 55, pp. 31:1–31:14. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, Rome, Italy (Jul 11–15, 2016)

[32] Chandran, N., Kanukurthi, B., Ostrovsky, R.: Locally updatable and locally decodable codes. In: Lindell, Y. (ed.) TCC 2014: 11th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 8349, pp. 489–514. Springer, Heidelberg, Germany, San Diego, CA, USA (Feb 24–26, 2014)

[33] Chandran, N., Kanukurthi, B., Raghuraman, S.: Information-theoretic local non-malleable codes and their applications. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A: 13th Theory of Cryptography Conference, Part II. Lecture Notes in Computer Science, vol. 9563, pp. 367–392. Springer, Heidelberg, Germany, Tel Aviv, Israel (Jan 10–13, 2016)

[34] Chattopadhyay, E., Goyal, V., Li, X.: Non-malleable extractors and codes, with their many tampered extensions. In: Wichs, D., Mansour, Y. (eds.) 48th Annual ACM Symposium on Theory of Computing. pp. 285–298. ACM Press, Cambridge, MA, USA (Jun 18–21, 2016)

[35] Chattopadhyay, E., Kanukurthi, B., Obbattu, S.L.B., Sekar, S.: Privacy amplification from non-malleable codes. Cryptology ePrint Archive, Report 2018/293 (2018), https://eprint.iacr.org/2018/293

[36] Chattopadhyay, E., Li, X.: Non-malleable codes and extractors for small-depth circuits, and affine functions. In: Hatami, H., McKenzie, P., King, V. (eds.) 49th Annual ACM Symposium on Theory of Computing. pp. 1171–1184. ACM Press, Montreal, QC, Canada (Jun 19–23, 2017)

[37] Chattopadhyay, E., Li, X.: Non-malleable extractors and codes for composition of tampering, interleaved tampering and more. Cryptology ePrint Archive, Report 2018/1069 (2018), https://eprint.iacr.org/2018/1069

[38] Chattopadhyay, E., Li, X.: Non-malleable extractors and codes in the interleaved split-state model and more. arXiv preprint arXiv:1804.05228 (2018)

[39] Chattopadhyay, E., Zuckerman, D.: Non-malleable codes against constant split-state tampering. In: 55th Annual Symposium on Foundations of Computer Science. pp. 306–315. IEEE Computer Society Press, Philadelphia, PA, USA (Oct 18–21, 2014)

[40] Chee, Y.M., Feng, T., Ling, S., Wang, H., Zhang, L.F.: Query-efficient locally decodable codes of subexponential length. computational complexity 22(1), 159–189 (2013), http://dx.doi.org/10.1007/s00037-011-0017-1

[41] Chen, R., Santhanam, R., Srinivasan, S.: Average-case lower bounds and satisfiability algorithms for small threshold circuits. In: Raz, R. (ed.) 31st Conference on Computational Complexity, CCC 2016, May 29 to June 1, 2016, Tokyo, Japan. LIPIcs, vol. 50, pp. 1:1–1:35. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2016), https://doi.org/10.4230/LIPIcs.CCC.2016.1

[42] Cheraghchi, M., Guruswami, V.: Capacity of non-malleable codes. In: Naor, M. (ed.) ITCS 2014: 5th Conference on Innovations in Theoretical Computer Science. pp. 155–168. Association for Computing Machinery, Princeton, NJ, USA (Jan 12–14, 2014)

[43] Cheraghchi, M., Guruswami, V.: Non-malleable coding against bit-wise and split-state tampering. In: Lindell, Y. (ed.) TCC 2014: 11th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 8349, pp. 440–464. Springer, Heidelberg, Germany, San Diego, CA, USA (Feb 24–26, 2014)

[44] Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: Black-box construction of a non-malleable encryption scheme from any semantically secure one. In: Canetti, R. (ed.) TCC 2008: 5th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 4948, pp. 427–444. Springer, Heidelberg, Germany, San Francisco, CA, USA (Mar 19–21, 2008)

[45] Choi, S.G., Dachman-Soled, D., Malkin, T., Wee, H.: A note on improved, black-box constructions of non-malleable encryption from semantically-secure encryption. Manuscript (2015)

[46] Choi, S.G., Kiayias, A., Malkin, T.: BiTR: Built-in tamper resilience. In: Lee, D.H., Wang, X. (eds.) Advances in Cryptology – ASIACRYPT 2011. Lecture Notes in Computer Science, vol. 7073, pp. 740–758. Springer, Heidelberg, Germany, Seoul, South Korea (Dec 4–8, 2011)

[47] Cojocar, L., Razavi, K., Giuffrida, C., Bos, H.: Exploiting correcting codes: On the effectiveness of ecc memory against rowhammer attacks. In: 2019 2019 IEEE Symposium on Security and Privacy (SP). vol. 1, pp. 279–295. IEEE Computer Society, Los Alamitos, CA, USA (may 2019), https://doi.ieeecomputersociety.org/10.1109/SP.2019.00089

[48] Coretti, S., Dodis, Y., Tackmann, B., Venturi, D.: Non-malleable encryption: Simpler, shorter, stronger. Cryptology ePrint Archive, Report 2015/772 (2015), http://eprint.iacr.org/2015/772

[49] Coretti, S., Dodis, Y., Tackmann, B., Venturi, D.: Non-malleable encryption: Simpler, shorter, stronger. In: Kushilevitz, E., Malkin, T. (eds.) TCC 2016-A: 13th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 9562, pp. 306–335. Springer, Heidelberg, Germany, Tel Aviv, Israel (Jan 10–13, 2016)

[50] Coretti, S., Faonio, A., Venturi, D.: Rate-optimizing compilers for continuously non-malleable codes. Cryptology ePrint Archive, Report 2019/055 (2019), https://eprint.iacr.org/2019/055

[51] Coron, J.S.: Security proof for partial-domain hash signature schemes. In: Yung, M. (ed.) Advances in Cryptology – CRYPTO 2002. Lecture Notes in Computer Science, vol. 2442, pp. 613–626. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2002)

[52] Dachman-Soled, D., Kulkarni, M.: Upper and lower bounds for continuous non-malleable codes. In: Lin, D., Sako, K. (eds.) PKC 2019: 22nd International Conference on Theory and Practice of Public Key Cryptography, Part I. Lecture Notes in Computer Science, vol. 11442, pp. 519–548. Springer, Heidelberg, Germany, Beijing, China (Apr 14–17, 2019)

[53] Dachman-Soled, D., Kulkarni, M., Shahverdi, A.: Tight upper and lower bounds for leakage-resilient, locally decodable and updatable non-malleable codes. In: Fehr, S. (ed.) PKC 2017: 20th International Conference on Theory and Practice of Public Key Cryptography, Part I. Lecture Notes in Computer Science, vol. 10174, pp. 310–332. Springer, Heidelberg, Germany, Amsterdam, The Netherlands (Mar 28–31, 2017)

[54] Dachman-Soled, D., Liu, F.H., Shi, E., Zhou, H.S.: Locally decodable and updatable non-malleable codes and their applications. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015: 12th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 9014, pp. 427–450. Springer, Heidelberg, Germany, Warsaw, Poland (Mar 23–25, 2015)

[55] De Santis, A., Di Crescenzo, G., Ostrovsky, R., Persiano, G., Sahai, A.: Robust non-interactive zero knowledge. In: Kilian, J. (ed.) Advances in Cryptology – CRYPTO 2001. Lecture Notes in Computer Science, vol. 2139, pp. 566–598. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2001)

[56] De Wolf, R.: A brief introduction to fourier analysis on the boolean cube. Theory of Computing, Graduate Surveys 1, 1–20 (2008)

[57] Decatur, S.E., Goldreich, O., Ron, D.: Computational sample complexity. SIAM Journal on Computing 29(3), 854–879 (2000)

[58] Di Crescenzo, G., Ishai, Y., Ostrovsky, R.: Non-interactive and non-malleable commitment. In: 30th Annual ACM Symposium on Theory of Computing. pp. 141–150. ACM Press, Dallas, TX, USA (May 23–26, 1998)

[59] Dolev, D., Dwork, C., Naor, M.: Nonmalleable cryptography. SIAM Journal on Computing 30(2), 391–437 (2000)

[60] Döttling, N., Nielsen, J.B., Obremski, M.: Information theoretic continuously non-malleable codes in the constant split-state model. Cryptology ePrint Archive, Report 2017/357 (2017), `http://eprint.iacr.org/2017/357`

[61] Drucker, A.: New limits to classical and quantum instance compression. SIAM Journal on Computing 44(5), 1443–1479 (2015)

[62] Dubrov, B., Ishai, Y.: On the randomness complexity of efficient sampling. In: Kleinberg, J.M. (ed.) 38th Annual ACM Symposium on Theory of Computing. pp. 711–720. ACM Press, Seattle, WA, USA (May 21–23, 2006)

[63] Dwork, C., Naor, M., Reingold, O.: Immunizing encryption schemes from decryption errors. In: Cachin, C., Camenisch, J. (eds.) Advances in Cryptology – EUROCRYPT 2004. Lecture Notes in Computer Science, vol. 3027, pp. 342–360. Springer, Heidelberg, Germany, Interlaken, Switzerland (May 2–6, 2004)

[64] Dziembowski, S., Kazana, T., Obremski, M.: Non-malleable codes from two-source extractors. In: Canetti, R., Garay, J.A. (eds.) Advances in Cryptology – CRYPTO 2013, Part II. Lecture Notes in Computer Science, vol. 8043, pp. 239–257. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 18–22, 2013)

[65] Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. In: Yao, A.C.C. (ed.) ICS 2010: 1st Innovations in Computer Science. pp. 434–452. Tsinghua University Press, Tsinghua University, Beijing, China (Jan 5–7, 2010)

[66] Dziembowski, S., Pietrzak, K., Wichs, D.: Non-malleable codes. J. ACM 65(4), 20:1–20:32 (Apr 2018), `http://doi.acm.org/10.1145/3178432`*, extended abstract appeared in Innovations in Computer Science (ICS) 2010*

[67] Efremenko, K.: 3-query locally decodable codes of subexponential length. In: Mitzenmacher, M. (ed.) 41st Annual ACM Symposium on Theory of Computing. pp. 39–44. ACM Press, Bethesda, MD, USA (May 31 – Jun 2, 2009)

[68] Erds, P., Rado, R.: Intersection theorems for systems of sets. Journal of the London Mathematical Society s1-35(1), 85–90 (1960), `http://dx.doi.org/10.1112/jlms/s1-35.1.85`

[69] Faonio, A., Nielsen, J.B., Simkin, M., Venturi, D.: Continuously non-malleable codes with split-state refresh. In: Preneel, B., Vercauteren, F. (eds.) ACNS 18: 16th International Conference on Applied Cryptography and Network Security. Lecture Notes in Computer Science, vol. 10892, pp. 121–139. Springer, Heidelberg, Germany, Leuven, Belgium (Jul 2–4, 2018)

[70] Faust, S., Hostáková, K., Mukherjee, P., Venturi, D.: Non-malleable codes for space-bounded tampering. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology – CRYPTO 2017, Part II. Lecture Notes in Computer Science, vol. 10402, pp. 95–126. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 2017)

[71] Faust, S., Mukherjee, P., Nielsen, J.B., Venturi, D.: Continuous non-malleable codes. In: Lindell, Y. (ed.) TCC 2014: 11th Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 8349, pp. 465–488. Springer, Heidelberg, Germany, San Diego, CA, USA (Feb 24–26, 2014)

[72] Faust, S., Mukherjee, P., Nielsen, J.B., Venturi, D.: A tamper and leakage resilient von neumann architecture. In: Katz, J. (ed.) PKC 2015: 18th International Conference on Theory and Practice of Public Key Cryptography. Lecture Notes in Computer Science, vol. 9020, pp. 579–603. Springer, Heidelberg, Germany, Gaithersburg, MD, USA (Mar 30 – Apr 1, 2015)

[73] Faust, S., Mukherjee, P., Venturi, D., Wichs, D.: Efficient non-malleable codes and key-derivation for poly-size tampering circuits. In: Nguyen, P.Q., Oswald, E. (eds.) Advances in Cryptology – EUROCRYPT 2014. Lecture Notes in Computer Science, vol. 8441, pp. 111–128. Springer, Heidelberg, Germany, Copenhagen, Denmark (May 11–15, 2014)

[74] Fischlin, M., Schröder, D.: On the impossibility of three-move blind signature schemes. In: Gilbert, H. (ed.) Advances in Cryptology – EUROCRYPT 2010.

Lecture Notes in Computer Science, vol. 6110, pp. 197–215. Springer, Heidelberg, Germany, French Riviera (May 30 – Jun 3, 2010)

[75] Fuchsbauer, G., Konstantinov, M., Pietrzak, K., Rao, V.: Adaptive security of constrained PRFs. In: Sarkar, P., Iwata, T. (eds.) Advances in Cryptology – ASIACRYPT 2014, Part II. Lecture Notes in Computer Science, vol. 8874, pp. 82–101. Springer, Heidelberg, Germany, Kaoshiung, Taiwan, R.O.C. (Dec 7–11, 2014)

[76] Garg, S., Bhaskar, R., Lokam, S.V.: Improved bounds on security reductions for discrete log based signatures. In: Wagner, D. (ed.) Advances in Cryptology – CRYPTO 2008. Lecture Notes in Computer Science, vol. 5157, pp. 93–107. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2008)

[77] Gennaro, R., Lysyanskaya, A., Malkin, T., Micali, S., Rabin, T.: Algorithmic tamper-proof (ATP) security: Theoretical foundations for security against hardware tampering. In: Naor, M. (ed.) TCC 2004: 1st Theory of Cryptography Conference. Lecture Notes in Computer Science, vol. 2951, pp. 258–277. Springer, Heidelberg, Germany, Cambridge, MA, USA (Feb 19–21, 2004)

[78] Gentry, C., Wichs, D.: Separating succinct non-interactive arguments from all falsifiable assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd Annual ACM Symposium on Theory of Computing. pp. 99–108. ACM Press, San Jose, CA, USA (Jun 6–8, 2011)

[79] Gertner, Y., Kannan, S., Malkin, T., Reingold, O., Viswanathan, M.: The relationship between public key encryption and oblivious transfer. In: 41st Annual Symposium on Foundations of Computer Science. pp. 325–335. IEEE Computer Society Press, Redondo Beach, CA, USA (Nov 12–14, 2000)

[80] Groth, J., Maller, M.: Snarky signatures: Minimal signatures of knowledge from simulation-extractable snarks. In: Katz, J., Shacham, H. (eds.) Advances in Cryptology - CRYPTO 2017 - 37th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 20-24, 2017, Proceedings, Part II. Lecture Notes in Computer Science, vol. 10402, pp. 581–612. Springer (2017), https://doi.org/10.1007/978-3-319-63715-0_20

[81] Guo, A., Kopparty, S., Sudan, M.: New affine-invariant codes from lifting. In: Kleinberg, R.D. (ed.) ITCS 2013: 4th Innovations in Theoretical Computer Science. pp. 529–540. Association for Computing Machinery, Berkeley, CA, USA (Jan 9–12, 2013)

[82] Håstad, J.: Computational limitations of small-depth circuits (1987)

[83] Håstad, J.: On the correlation of parity and small-depth circuits. SIAM Journal on Computing 43(5), 1699–1708 (2014)

[84] Hemenway, B., Ostrovsky, R., Wootters, M.: Local correctability of expander codes. In: Fomin, F.V., Freivalds, R., Kwiatkowska, M.Z., Peleg, D. (eds.) ICALP 2013: 40th International Colloquium on Automata, Languages and Programming, Part I. Lecture Notes in Computer Science, vol. 7965, pp. 540–551. Springer, Heidelberg, Germany, Riga, Latvia (Jul 8–12, 2013)

[85] Impagliazzo, R., Matthews, W., Paturi, R.: A satisfiability algorithm for ac 0. In: Proceedings of the twenty-third annual ACM-SIAM symposium on Discrete Algorithms. pp. 961–972. Society for Industrial and Applied Mathematics (2012)

[86] Impagliazzo, R., Rudich, S.: Limits on the provable consequences of one-way permutations. In: 21st Annual ACM Symposium on Theory of Computing. pp. 44–61. ACM Press, Seattle, WA, USA (May 15–17, 1989)

[87] Ishai, Y., Prabhakaran, M., Sahai, A., Wagner, D.: Private circuits II: Keeping secrets in tamperable circuits. In: Vaudenay, S. (ed.) Advances in Cryptology – EUROCRYPT 2006. Lecture Notes in Computer Science, vol. 4004, pp. 308–327. Springer, Heidelberg, Germany, St. Petersburg, Russia (May 28 – Jun 1, 2006)

[88] Ishai, Y., Sahai, A., Wagner, D.: Private circuits: Securing hardware against probing attacks. In: Boneh, D. (ed.) Advances in Cryptology – CRYPTO 2003. Lecture Notes in Computer Science, vol. 2729, pp. 463–481. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 17–21, 2003)

[89] Jafargholi, Z., Wichs, D.: Tamper detection and continuous non-malleable codes. In: Dodis, Y., Nielsen, J.B. (eds.) TCC 2015: 12th Theory of Cryptography Conference, Part I. Lecture Notes in Computer Science, vol. 9014, pp. 451–480. Springer, Heidelberg, Germany, Warsaw, Poland (Mar 23–25, 2015)

[90] Kanukurthi, B., Obbattu, S.L.B., Sekar, S.: Four-state non-malleable codes with explicit constant rate. In: Kalai, Y., Reyzin, L. (eds.) TCC 2017: 15th Theory of Cryptography Conference, Part II. Lecture Notes in Computer Science, vol. 10678, pp. 344–375. Springer, Heidelberg, Germany, Baltimore, MD, USA (Nov 12–15, 2017)

[91] Kanukurthi, B., Obbattu, S.L.B., Sekar, S.: Non-malleable randomness encoders and their applications. In: Nielsen, J.B., Rijmen, V. (eds.) Advances in Cryptology – EUROCRYPT 2018, Part III. Lecture Notes in Computer Science, vol. 10822, pp. 589–617. Springer, Heidelberg, Germany, Tel Aviv, Israel (Apr 29 – May 3, 2018)

[92] Katz, J., Trevisan, L.: On the efficiency of local decoding procedures for error-correcting codes. In: 32nd Annual ACM Symposium on Theory of Computing. pp. 80–86. ACM Press, Portland, OR, USA (May 21–23, 2000)

[93] *Kiayias, A., Liu, F.H., Tselekounis, Y.: Practical non-malleable codes from l-more extractable hash functions. In: Weippl, E.R., Katzenbeisser, S., Kruegel, C., Myers, A.C., Halevi, S. (eds.) ACM CCS 2016: 23rd Conference on Computer and Communications Security. pp. 1317–1328. ACM Press, Vienna, Austria (Oct 24–28, 2016)*

[94] *Kiayias, A., Liu, F.H., Tselekounis, Y.: Non-malleable codes for partial functions with manipulation detection. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018, Part III. Lecture Notes in Computer Science, vol. 10993, pp. 577–607. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2018)*

[95] *Kocher, P.C., Jaffe, J., Jun, B.: Differential power analysis. In: Wiener, M.J. (ed.) Advances in Cryptology – CRYPTO'99. Lecture Notes in Computer Science, vol. 1666, pp. 388–397. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 15–19, 1999)*

[96] *Kopparty, S., Saraf, S., Yekhanin, S.: High-rate codes with sublinear-time decoding. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd Annual ACM Symposium on Theory of Computing. pp. 167–176. ACM Press, San Jose, CA, USA (Jun 6–8, 2011)*

[97] *Li, X.: Improved non-malleable extractors, non-malleable codes and independent source extractors. In: Hatami, H., McKenzie, P., King, V. (eds.) 49th Annual ACM Symposium on Theory of Computing. pp. 1144–1156. ACM Press, Montreal, QC, Canada (Jun 19–23, 2017)*

[98] *Li, X.: Non-malleable extractors and non-malleable codes: Partially optimal constructions. Cryptology ePrint Archive, Report 2018/353 (2018), https://eprint.iacr.org/2018/353*

[99] *Lindell, Y.: A simpler construction of cca2-secure public-key encryption under general assumptions. In: Biham, E. (ed.) Advances in Cryptology – EUROCRYPT 2003. Lecture Notes in Computer Science, vol. 2656, pp. 241–254. Springer, Heidelberg, Germany, Warsaw, Poland (May 4–8, 2003)*

[100] *Liu, F.H., Lysyanskaya, A.: Tamper and leakage resilience in the split-state model. In: Safavi-Naini, R., Canetti, R. (eds.) Advances in Cryptology – CRYPTO 2012. Lecture Notes in Computer Science, vol. 7417, pp. 517–532. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2012)*

[101] *Naor, M.: Bit commitment using pseudo-randomness. In: Brassard, G. (ed.) Advances in Cryptology – CRYPTO'89. Lecture Notes in Computer Science, vol. 435, pp. 128–136. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 20–24, 1990)*

[102] Naor, M., Yung, M.: Public-key cryptosystems provably secure against chosen ciphertext attacks. In: 22nd Annual ACM Symposium on Theory of Computing. pp. 427–437. ACM Press, Baltimore, MD, USA (May 14–16, 1990)

[103] Nisan, N.: Pseudorandom generators for space-bounded computation. Combinatorica 12(4), 449–461 (1992)

[104] Oliveira, I.C., Santhanam, R.: Pseudodeterministic constructions in subexponential time. In: Hatami, H., McKenzie, P., King, V. (eds.) Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing, STOC 2017, Montreal, QC, Canada, June 19-23, 2017. pp. 665–677. ACM (2017), http://doi.acm.org/10.1145/3055399.3055500

[105] Ostrovsky, R., Persiano, G., Venturi, D., Visconti, I.: Continuously non-malleable codes in the split-state model from minimal assumptions. In: Shacham, H., Boldyreva, A. (eds.) Advances in Cryptology – CRYPTO 2018, Part III. Lecture Notes in Computer Science, vol. 10993, pp. 608–639. Springer, Heidelberg, Germany, Santa Barbara, CA, USA (Aug 19–23, 2018)

[106] Paillier, P., Vergnaud, D.: Discrete-log-based signatures may not be equivalent to discrete log. In: Roy, B.K. (ed.) Advances in Cryptology – ASIACRYPT 2005. Lecture Notes in Computer Science, vol. 3788, pp. 1–20. Springer, Heidelberg, Germany, Chennai, India (Dec 4–8, 2005)

[107] Pass, R.: Limits of provable security from standard assumptions. In: Fortnow, L., Vadhan, S.P. (eds.) 43rd Annual ACM Symposium on Theory of Computing. pp. 109–118. ACM Press, San Jose, CA, USA (Jun 6–8, 2011)

[108] Rao, A.: An exposition of bourgains 2-source extractor. In: Electronic Colloquium on Computational Complexity (ECCC). vol. 14 (2007)

[109] Rasmussen, P.M.R., Sahai, A.: Expander graphs are non-malleable codes. Cryptology ePrint Archive, Report 2018/929 (2018), https://eprint.iacr.org/2018/929

[110] Raz, R.: Fast learning requires good memory: A time-space lower bound for parity learning. CoRR abs/1602.05161 (2016), http://arxiv.org/abs/1602.05161

[111] Sahai, A.: Non-malleable non-interactive zero knowledge and adaptive chosen-ciphertext security. In: 40th Annual Symposium on Foundations of Computer Science. pp. 543–553. IEEE Computer Society Press, New York, NY, USA (Oct 17–19, 1999)

[112] Seaborn, M., Dullien, T.: Exploiting the dram rowhammer bug to gain kernel privileges

[113] Seurin, Y.: On the exact security of Schnorr-type signatures in the random oracle model. In: Pointcheval, D., Johansson, T. (eds.) Advances in Cryptology – EUROCRYPT 2012. Lecture Notes in Computer Science, vol. 7237, pp. 554–571. Springer, Heidelberg, Germany, Cambridge, UK (Apr 15–19, 2012)

[114] Simon, D.R.: Finding collisions on a one-way street: Can secure hash functions be based on general assumptions? In: Nyberg, K. (ed.) Advances in Cryptology – EUROCRYPT'98. Lecture Notes in Computer Science, vol. 1403, pp. 334–345. Springer, Heidelberg, Germany, Espoo, Finland (May 31 – Jun 4, 1998)

[115] Skorobogatov, S.P., Anderson, R.J.: Optical fault induction attacks. In: Kaliski Jr., B.S., Koç, Çetin Kaya., Paar, C. (eds.) Cryptographic Hardware and Embedded Systems – CHES 2002. Lecture Notes in Computer Science, vol. 2523, pp. 2–12. Springer, Heidelberg, Germany, Redwood Shores, CA, USA (Aug 13–15, 2003)

[116] Tal, A.: Properties and applications of boolean function composition. In: Kleinberg, R.D. (ed.) ITCS 2013: 4th Innovations in Theoretical Computer Science. pp. 441–454. Association for Computing Machinery, Berkeley, CA, USA (Jan 9–12, 2013)

[117] Tal, A.: Tight bounds on the fourier spectrum of AC0. In: O'Donnell, R. (ed.) 32nd Computational Complexity Conference, CCC 2017, July 6-9, 2017, Riga, Latvia. LIPIcs, vol. 79, pp. 15:1–15:31. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2017), https://doi.org/10.4230/LIPIcs.CCC.2017.15

[118] Viola, E.: Extractors for circuit sources. SIAM J. Comput. 43(2), 655–672 (2014), http://dx.doi.org/10.1137/11085983X

[119] Yekhanin, S.: Towards 3-query locally decodable codes of subexponential length. J. ACM 55(1), 1:1–1:16 (Feb 2008), http://doi.acm.org/10.1145/1326554.1326555

[120] Yekhanin, S.: Locally decodable codes: A brief survey. In: Chee, Y.M., Guo, Z., Ling, S., Shao, F., Tang, Y., Wang, H., Xing, C. (eds.) Coding and Cryptology: Third International Workshop, IWCC 2011, Qingdao, China, May 30-June 3, 2011. Proceedings, pp. 273–282. Springer Berlin Heidelberg, Berlin, Heidelberg (2011), http://dx.doi.org/10.1007/978-3-642-20901-7_18